Just A Rather Very Intelligent System (J.A.R.V.I.S) - Real Time Animated Head Assistant

Matthew Wilmot
mcwilmot@mit.edu

Danielle Knutson
daniknut@mit.edu

A. Motivation

In this project, we set out to build a real-time, stylized animated assistant in the form of a customizable floating 3D head that talks and moves its face in sync with LLM-generated speech. The core problem is how to take text responses from a large language model, convert them to speech, and then drive believable facial animation with mouth, jaw, eyes, eyelids, and facial distortions that reflect both the phonetic content and the intended emotion. This tackles the broader challenge of making LLMs feel embodied and expressive, rather than just text boxes, by turning them into characters that can speak and react visually in real time.

Our goal was to implement an end-to-end pipeline that takes a user prompt, calls an external LLM (ChatGPT) to generate a response, and then speaks that response while animating a 3D head in real time. At a minimum, we aimed to build a neutral head mesh with a set of morph targets for visemes and emotions, combine them in an animation controller driven by phoneme and emotion labels, and render the result using GLOO. We also hoped to explore how well a finite set of discrete visemes and emotion presets can approximate natural speech and expression, and to understand the practical issues that arise when synchronizing audio, text, and animation in real time.

Overall, in this project, we implemented a real-time 3D conversational assistant: a floating, stylized head rendered with GLOO that speaks LLM-generated responses while its face animates in sync. The assistant receives text from a ChatGPT API call, passes it through a text-to-speech and grapheme-to-phoneme pipeline, and uses the resulting phoneme sequence to drive a set of viseme morph targets on the head mesh. A separate, discrete emotion label (neutral, happy, sad, angry, excited, energetic, gloomy) modulates additional morph targets for the eyes, eyebrows, and overall facial pose. All graphics and animation are implemented in GLOO, so that the final system runs in real time and can be demonstrated by speaking to the assistant and watching its facial motion react to the generated speech and emotion.

B. Related Work

This project combines several topics from 6.4400 into a single, real-time interactive system. All visual elements are implemented using the GLOO framework, leveraging rasterization, shading, and animation to render a talking head that responds in real time to user input. The assistant uses morph targets and SSD-style facial deformations, similar to assignment 2, to realize viseme and emotion poses, while the renderer handles lighting and materials, ensuring the head remains visually coherent and expressive across different viewing conditions. We had to modify the Phong shader to work for the skybox and for reflections, similar to assignments 4 and 5. Our project draws directly on prior work in lip synchronization and speech-driven facial animation. In particular, we follow Edwards et al. in using a discrete viseme model that maps phonemes to a small set of animator-friendly mouth shapes, and we draw inspiration from Xing et al., who represent speech-driven facial motion as a finite set of discrete motion primitives [3].

Edwards et al., "An Animator-Centric Viseme Model for Expressive Lip Synchronization," propose an animator-centric viseme model that groups phonemes into a manageable number of visually distinguishable mouth shapes [1]. Their grouping reduces the full phoneme inventory to a finite set of visemes, which is easier to animate and blend in practice. In the project, we adopt this idea of a discrete viseme set and use it as the basis for defining lip and jaw morph targets that are activated according to the phoneme timeline.

Xing et al., "Speech-Driven 3D Facial Animation with Discrete Motion Prior," cast speech-driven facial animation into a codebook of discrete motion primitives, so that complex motion is built from a finite library of learned facial movements. This motivates my choice to represent emotion not as a continuous space but as a small set of discrete facial presets (e.g., happy, sad, angry) that can be combined with visemes. We mirror their emphasis on discrete, reusable motion units by designing emotion morph targets that can be mixed in alongside viseme morphs to shape the overall facial expression [3].

C. Approach

C.1. System Overview

The complete system consists of two major subsystems:

1. *Graphics & Animation Engine (C++ / GLOO):* Responsible for loading the 3D head mesh, blending the vertex targets, rendering the scene, and coordinating head, eye, and facial motion.

2. *LLM-Driven and TTS-Driven Speech Pipeline*

(Python): Generates emotionally-conditioned text, synthesizes speech audio, and produces phoneme timing for animation. These python files are called from inside the C++ system.

C.2. Head Mesh, Phoneme Basis, and Blendshape Architecture

A neutral 3D head mesh is loaded along with a JSON file containing phoneme-specific vertex displacements. We generated this from blender using an asset from Free3D. The system then stores a dictionary of active phoneme blend weights which allows multiple facial components (mouth, brows, blink, etc.) to combine simultaneously.

More specifically, given the neutral vertices V_{neutral} and per-phoneme vertex targets V_i, our code computes the final deformed mesh V iin each frame as:

$$V = V_{\text{neutral}} + \sum_i \alpha_i \big(V_i - V_{\text{neutral}}\big),$$

where α_i are the active phoneme weights.

C.3. Rendering Pipeline and Cube Map Setup

The code then initializes a real-time scene containing:

- A cubemap skybox providing ambient environmental lighting.
- Ambient, point, and directional lights.
- A Phong shader for both the skybox and the head mesh.
- An ArcBall camera for interactive scene manipulation.

This establishes a visually coherent environment before facial animation and speech synchronization are applied. We also adjusted the transparency of the mesh and use cube mapping (mapping the cube to the face itself) to simulate a glass type material [2].

(1) LLM-Based Text + Emotion Generation

Firstly, a dialog box on screen collects the user's text input prompt and sends it to an LLM using a structured JSON schema requesting:

- the final spoken utterance, and
- one emotional label from the set {neutral, happy, sad, angry, excited, energetic, gloomy}.

There is also another dialog box that allows for direct repetition of text that the user specifies. Both are processed in the same way.

(2) Emotion-Conditioned TTS

Secondly, the python scripts synthesize the speech audio with emotional adjustments to speaking rate and volume. The output is a WAV file consumed by the C++

(3) Phoneme Extraction and Alignment

Thirdly, using CMUdict, the TTS stage converts text into ARPAbet phonemes, maps them onto a reduced set of blendshape categories (the ones that the model supports), and exports a JSON alignment file containing phoneme labels and timestamps.

C.4. Real-Time Phoneme and Facial Animation

After all that, during audio playback, the code continuously:

1. Computes the current phoneme segment based on elapsed audio time.
2. Ramps blend weights in and out (we had to do this to avoid popping artifacts).
3. Resumes idle behavior once the utterance finishes.

The system also includes a blinking subsystem and micro eye-dart behavior to improve perceived realism when the thing is idle.

C.5. Procedural Head Motion and Speaking Micro-Motions

Finally, to avoid robotic rigidity, head movements are generated through procedural quaternion-based animation segments. Each speaking motion follows a structured cycle:

1. Return briefly to the neutral head orientation.
2. Perform a small tilt or down-left/down-right glance.
3. Hold the pose for a natural dwell time.
4. Return smoothly to neutral.
5. Pause before permitting another micro-motion.

D. Results and Discussion

Qualitatively, the system produces convincing lip-sync and recognizable emotions. Even though it is imperfect and rough, we still find that the mouth generally moves in step with the audio and vowels and longer consonants surprisingly look especially natural. However, pauses are not treated very well and very rapid phonemes sometimes blur into neighboring shapes. Nevertheless, the mead movement outside of and during speech actually compensate for these

pitfalls as the change in eye contact added that much needed element of realism. Informally, viewers even reported that it "felt like the head was actually speaking," with emotions that were easy to distinguish.

In terms of performance though, once the audio and phoneme timings are available, the GLOO application runs comfortably in real time. Morph blending, deformation, and shading fit within a single frame, and the head animates smoothly at interactive frame rates. The main latency comes from upstream LLM and TTS calls rather than from rendering. Despite all this however, several limitations remain. Firstly, the discrete viseme model does not capture advanced coarticulation and strong emotions or movements can occasionally look quite under-exaggerated when combined with extreme mouth shapes. Furthermore, the lack of detailed skin textures and global hair motion keeps the character in a more stylized regime. Though we were going for that initially, it would have been great to at least have that option.

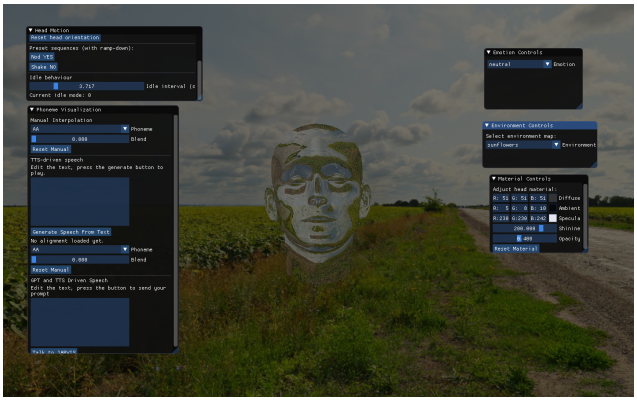

Figure 1. Basic GUI Layout

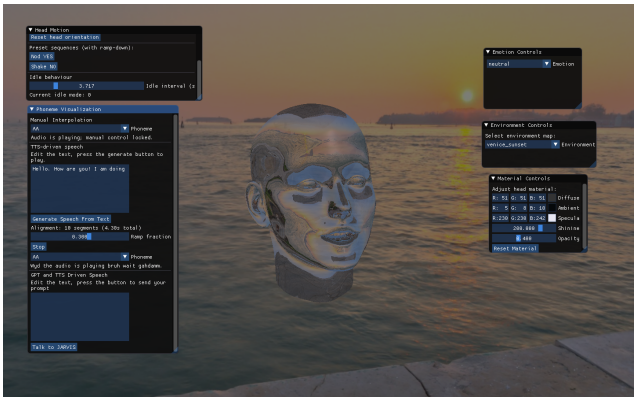

Figure 2. Phoneme Morphing during Speech

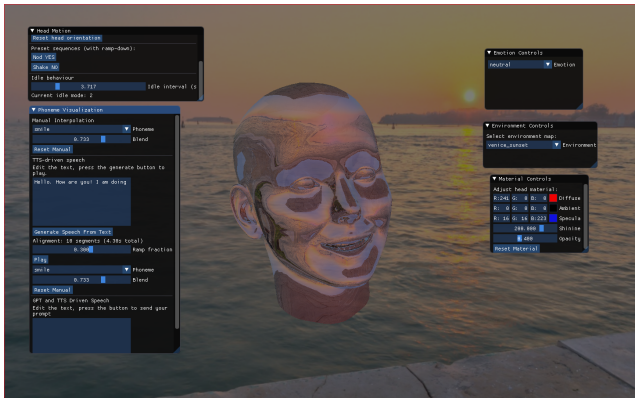

Figure 3. Customization of Colour + Idle Animation

E. Conclusion and Next Steps

In this project, we successfully implemented an end-to-end real-time 3D conversational assistant. On the graphics side, we modeled a neutral head mesh with SSD-style deformations and a library of morph targets for visemes and human motion, blended them per-frame in an animation controller, and shaded the result with Phong lighting, reflections, and a customizable skybox. On the systems side, we wired together the ChatGPT API, a text-to-speech and grapheme-to-phoneme pipeline, and a timing-driven animation loop that maps phoneme timestamps to viseme weights and layers on discrete emotion presets.

In the future, we can implement an emotion control slider, so that people can play around with having the head say their response or certain words with different emotions. Another extension could be skin texture mapping, so people can customize their heads. We were unable to implement this feature because there were no free skin texture resources, and creating them ourselves would have taken too long. Regarding the head shading, someone could implement a true glass-like material that refracts light instead of mimicking it artificially using Cube maps as we have. Finally, on the LLM side, there could be additional implementation of interaction patterns, such as back-and-forth multi-turn conversations and memory, to enhance the realism of the interaction. We wanted to do this but we were limited by money constraints as the API costs would have been too high. As an extension of the rendered head though, there could be integration with VR/head-tracked stereo rendering to give it a floating, omnipotent J.A.R.V.I.S. feel.

References

[1] Patrick Edwards, Chris Landreth, Eugene Fiume, and Karan Singh. Jali: An animator-centric viseme model for expressive lip synchronization. *ACM Transactions on Graphics (TOG)*, 35(4):127:1–127:11, 2016. 1

[2] pnlmon. 460 free cubemap converted from https://hdrihaven.com/. https://devforum.roblox.com/t/460-free-cubemap-converted-from-httpshdrihavencom/1040110, 2021. Roblox Developer Forum – Community Resources. 2

[3] Jun Xing, Menglei Xia, Yajing Zhang, Xiaodong Cun, Jingjing Wang, and Tien-Tsin Wong. Codetalker: Speech-driven 3d facial animation with discrete motion prior. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 12780–12790, 2023. 1