Final Project Write-Up: Animated ASCII Art with Music Synchronization

Introduction

"A picture is worth a thousand words"

In this project, I took this quite literally, building not just an image but an entire video using only text. My broader motivation was to explore how lightweight, low-bandwidth visuals could support educational access in rural areas. An expanded version of this project could deliver engaging learning materials to children without the need for expensive hardware or high-speed internet.

Building on my midterm project, I focused on integrating audio into animated ASCII art. ASCII animations are essentially matrix manipulations, where each character represents part of an image. My goal was to sync beats or rhythm changes with visual effects within the ASCII display, simulating the experience of a video using only text and sound. By combining playfulness with purpose, I wanted to show that computational creativity can be both expressive and impactful.

Project Implementation Walkthrough

- The main driver of the animation is the function `animateAsciiShow.m`, which calls a sequence of modular animation effects (scroll, dance, wave, transition) in one figure window.
- ASCII art was generated using `img2txt.m`, which converts an image into a matrix of characters and saves it as a `.txt file`.
- Each animation effect is handled in its own file.
- To integrate music, I used matlab's `audioreader` and `audioplayer functions`. This allowed me to loop audio and stop it explicitly when the animation ends.
- In addition to the functions I created for my midterm project, I created `scrollOneStep` to enable frame-by-frame control of scroll animations, which allowed audio to loop cleanly during scrolling.
- I also wrote a function `fadeInOutAudio` to apply fade-in and fade-out effects to each audio track for smoother transitions.
- Between animations, I added transition sounds (e.g. `bell-transition.mp3`) that play before the next animation starts. These are hardcoded but well-timed.

- All transitions and their sounds are combined and sequenced inside `animateAsciiShow.m`

Discussion

This project was a bit more dynamic than my midterm project. It pushed me to think beyond modularity and automation, challenging me to also consider timing, synchronization, and the emotional pacing of visuals. While my midterm emphasized building reusable components, this final project focused more on achieving a cohesive flow. I designed the transitions and sounds to feel natural together, like different scenes in a video.

Along the way, I encountered challenges with audio playback inside functions—specifically that audio objects would terminate when the function ended. I solved this by integrating the `audioplayer` object into the main `animateAsciiShow` function, rather than managing audio separately inside individual animation files. Another major challenge was determining how long each animation should run. I initially hoped to dynamically adjust animation timing based on the audio track, but that proved too complex to implement within the project timeline. As a solution, I hardcoded the animation lengths to match the specific tracks I used.

Next Steps

In the future, I would like to build a system where animation timing adapts automatically to different audio tracks, and where visual effects are triggered dynamically by changes in the audio itself. This would make the system even more flexible and responsive.

Conclusion

Ultimately, this project started as a fun experiment and I thoroughly enjoyed working on it. Even though the idea seems simple, it made me think more seriously about real-world challenges, like how to make learning tools more accessible in low-resource settings. There's still a lot more that could be done to expand and improve what I built, but this project showed me how small ideas—like pairing basic animations with sound—can spark important conversations about access and education.

Resources Used

- MATLAB documentation (for audio and string manipulation)
- ChatGPT (for brainstorming and debugging)
- mixkit.co and pixabay (source of royalty-free sound effects and background music)

Relevance to Class Concepts

- Matrix manipulations: ASCII art is built from 2D character arrays, and each animation is a transformation on that matrix (scrolling = row shift, wave = offset by sine).

- Modularity : Each animation is self-contained and can be reused or reordered.

Folder Navigation to run the full animation:

1. Open `lab01_free.m`, which contains the call to `animateAsciiShow()`.
2. This function loads and runs all animations and audio from within the folder.
3. ASCII image `.txt` files are auto-generated using `img2txt.m`, and are read in by `setupAsciiDisplay.m`.
4. All `.wav` and `.mp3` audio files in the folder are used for individual animations or transitions.
5. All relevant animation logic is inside files like `scroll.m`, `dance.m`, `waveEffect.m`, `transition.m`, and `ZoomOut.m`.
6. To visualize the final output, simply run `lab01_free.m` in MATLAB and ensure the figure window is active.