Musically Composed Visuals-Design

Fourier2 Team

Spring 2023

1 Team

Justin Choi, Ryan Gaffney, Ian Lips, Matt Dim

2 Goal

Musically-Composed-Visuals Parallelizing audio transformation into different classifications regarding image generation and using graphics to display our results (in the form of fractals or other image processing). There are already ways of breaking music into components through fourier. These components would split songs into components such as basslines, vocals, accompaniment, etc.

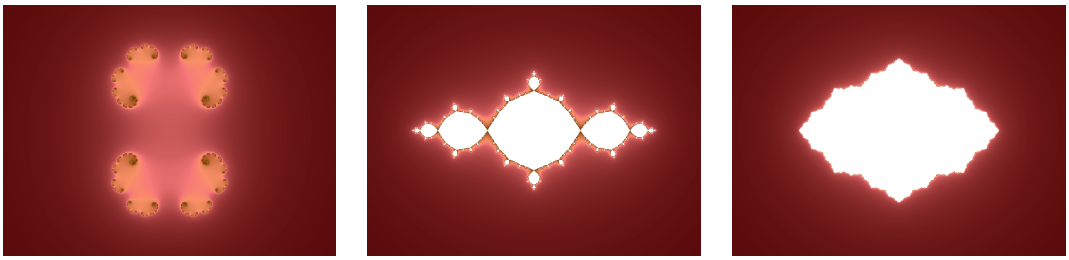

Figure 1: Polynomial Julia Set images

3 Relevance

OpenGL, a GPU library implementation talked about in the GPU part of the course, which will be used to animate our signal processed data to the screen. We will also be using both OpenMP and Pthreads for process speedup splitting up relevant tasks and data. This will be

4 Methods

We plan to use both OpenMP, Pthreads, and CUDA's managed memory to parallelize the processing of .mp3 and image generation. The data will be read from the src folder through native C I/O functions. We will then work on data and task parallelism in relevant sections of the signal processing functions. We expect that fourier transforms and audio compression will parallelize nicely because both of these functions are loops through data of the audio file. This means that we can have threads process different parts of a song at various points. Finally, we will use these processed signals and generate a mapping of audio to images. These images will be generated using OpenGL and we plan to use CUDA for parallelism. We will use the maximum kernels available on a machine for image generation of specific frames of a song. We hope to have an animation by the end that can be displayed per timestep. We will be performing testing during development in an incremental manner.

5 Possible Roadblocks

The scale of the project is definitely a big roadblock to consider, as our plan is to generate a picture for virtually every point of a song, which could include up to thousands of images to process and generate. Through parallelism this will help with the massive amounts of gpu computation however we have never worked on a scale this large and may need to pivot to a smaller style of deliverable. For example instead of creating thousands of images for each point of a song, we can instead use the entirety of a song and its elements/components for one or a set of images.

6 Draft Deliverable

For our draft, our submission will consist of an instance of a song that will be broken down into its components with all those components being parallelized and displayed into a single image.

Significant progress would consist of the serial implementation being finished where we can display an image from a section of a song (or from any information from the song). Excellent progress would be the implementation of threading and basic analysis on speedup with thorough testing. We would also include a serial version while getting openGL and cuda to communicate using the documentation from OpenGL Interoperability.

7 Final Deliverable

Alongside our code and report, we will also be submitting an audio file and the images processed by that audio file. The audio file will be sent in its base form (most likely an MP3) and if we have time we can compare the different kinds of images created depending on the current parameters. We may even be able to show the images through an animated style by having each individual part of a song its own image.

References

[1] N. Corporation, "OpenGL interoperability," *NVIDIA Documentation Hub*, 2023. [Online]. Available: `https://docs.nvidia.com/cuda/cuda-runtime-api/group__CUDART__OPENGL.html`. [Accessed: 03-Mar-2023].

[2] K. Kolsha, "Kolsha/STM32-audio-visualizer: Audio visualizer based on stm32 and Extension Board.," GitHub, 2018. [Online]. Available: `https://github.com/Kolsha/STM32-AUDIO-VISUALIZER`. [Accessed: 03-Mar-2023].

[3] C. Williams, "Online fractal generator," UsefulJS, 2014. [Online]. Available: `http://usefuljs.net/fractals/`. [Accessed: 03-Mar-2023].

[4] C. Zelga, A. Ulug, and J. Gilbert, "ASENAULUG/music-visualizer: A music visualizer which displays the real time fast Fourier transform (FFT) of music on an RGB led matrix," GitHub, 2019. [Online]. Available: `https://github.com/asenaulug/music-visualizer`. [Accessed: 03-Mar-2023].