Procedural sound visualization

For the course: TNM084 Procedural Methods for Images
Examiner: Ingemar Ragnemalm
Author: Ruben Bromée

2023–01–13

1 Aim

The aim of this project was to create a procedural visualization for sound that reacts on the frequency data and the amplitude data of the sound.

The features that were going to be included was a procedural visualization of sound using flow noise [1]. This visualization would be in two dimensions. The features that were optional in the project was a three dimensional visualization that would displace geometry depending on frequency and amplitude data of the sound.

2 Gradient noise

Gradient noise is a type of procedural noise that produces the color values of the fragments in an image by using randomly angled and sized gradients. One form of gradient noise is Perlin Noise [2].

Gradient noise is created by placing an uniform grid of randomly angled and sized gradients on an image. Each point in the image is evaluated by using the dot product between the point and the nearest gradients, as seen in Figure 1. The result of all the dot products for the nearest gradients are interpolated using the smoothstep function [3] shown in Equation 1. The interpolated dot products are shown in Figure 2. Depending on number of dimensions the number of gradients per box will vary. The number of gradients will be 2^n where n is the number of dimensions.

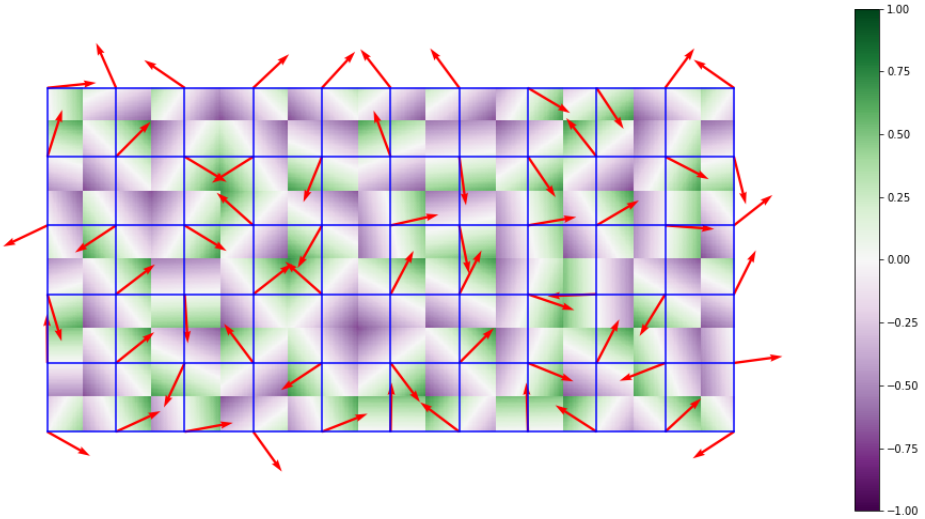

Figure 1: The dot products for each point with its single nearest gradient. The dot products with the other three gradients are not shown. The image is taken from [4]. The image is under a share-alike license [5]. No changes have been made to the image.

$$f(x) = \begin{cases} 0 & x \leq 0 \\ 3x^2 + 2x^3 & 0 \leq x \leq 1 \\ 1 & 1 \leq x \end{cases} \tag{1}$$

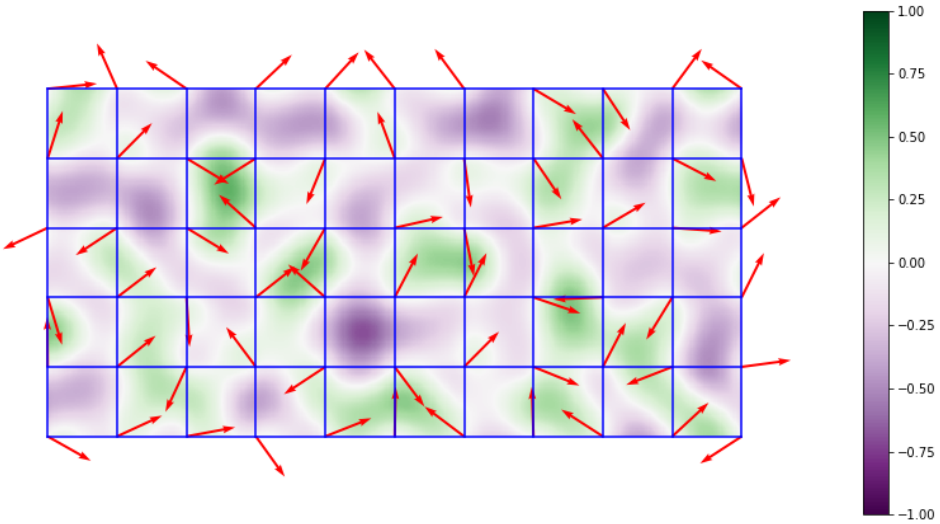

Figure 2: The dot products interpolated using the smoothstep function. The image is taken from [4]. The image is under a share-alike license [5]. No changes have been made to the image.

2.1 Multi octave gradient noise

To produce a more interesting noise multiple layers of the gradient noise with double the grid resolution and half the amplitude are combined. Changing the grid resolution of the gradient noise is also referred to as changing the frequency of the gradient noise. These additional layers with double the frequency and half the amplitude are referred to as octaves.

2.2 Flow noise

Flow noise is essentially using gradient noise and rotating the gradients. When the gradients are rotated a cyclical swirl will be created in the gradient noise. The gradients of each octave in a multi octave gradient noise will be rotated.

3 Method

The visualization was created using Python [6]. Librosa [7], which is a sound analysis library for Python, was used for sound analysis. PyOpenGL [8], which is Python bindings for OpenGL was used for rendering. Pygame [9] was used to play the sound while performing the rendering.

The sound was first loaded and split up into sample windows. Each window about 12 ms long. For each sample window the average amplitude of the sound was calculated and a short time fourier transform [3] was used to calculate the average of the five most dominating frequencies in a sample window. After this had been performed the results were normalized to floating point values between 0 and 1. These calculations were done on initialization and then two arrays were created that contained the normalized frequency and amplitude values for each frame. In each frame of the rendering the normalized frequency and amplitude value from the created arrays were sent to the fragment shader in PyOpenGL to affect the rendering of that specific frame. In each frame both the frequency and amplitude were used to affect the visualization.

A multi octave flow noise was used for this visualization. On top of having a constant rotation amount based on the time the flow noise was affected by varying the rotation amount of the gradients. Higher amplitude and frequency values would rotate the gradients faster, lower amplitude and frequency values would rotate the gradients slower. Since there may be considerable variations in the amplitude and frequency values it was not guaranteed that the gradients would rotate in the same direction all the time. They may oscillate because of these variations.

In addition to the flow noise visualization a radial sine wave based visualization was also used. This visualization was based on each points distance to the center of the window, frequency of the sound and the amplitude of the sound. The frequency and the amplitude of the sound affected the frequency of the sine wave visualization.

For both the radial sine wave visualization and the flow noise visualization the colors also vary depending on sound frequency and amplitude.

The final implementation of the flow noise that was chosen had an additional step. The absolute value of the gradient noise for each octave was used. This created the cloud like appearance that can be seen in Figure 3.

4 Result

The final implementation of the visualizer can be found on GitHub [10]. The final appearance of the visualizer can be seen in Figure 3, 4, 5 and 6.

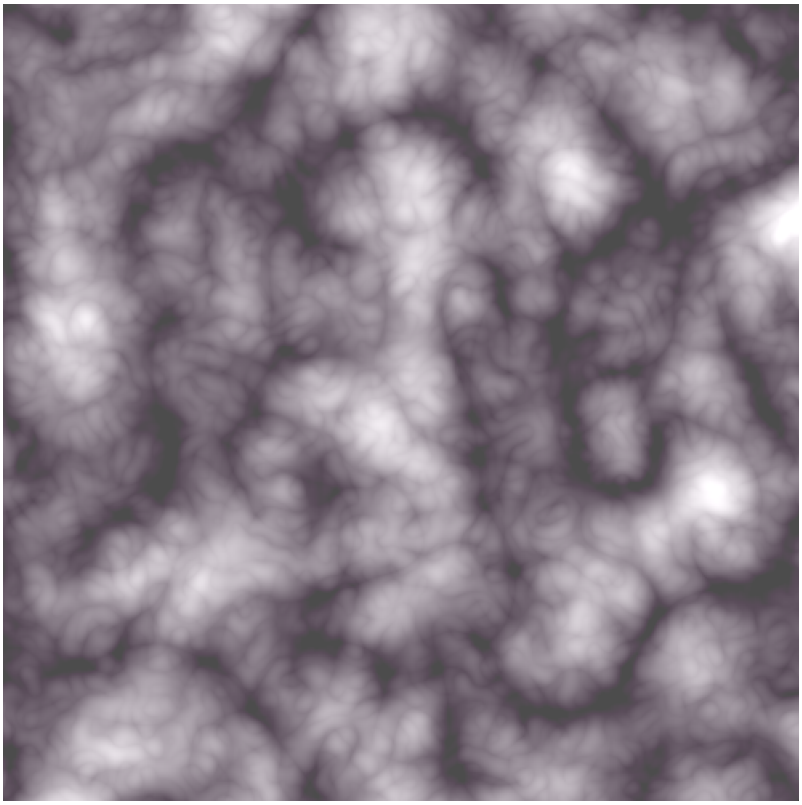

Figure 3: The flow noise used in the visualization.

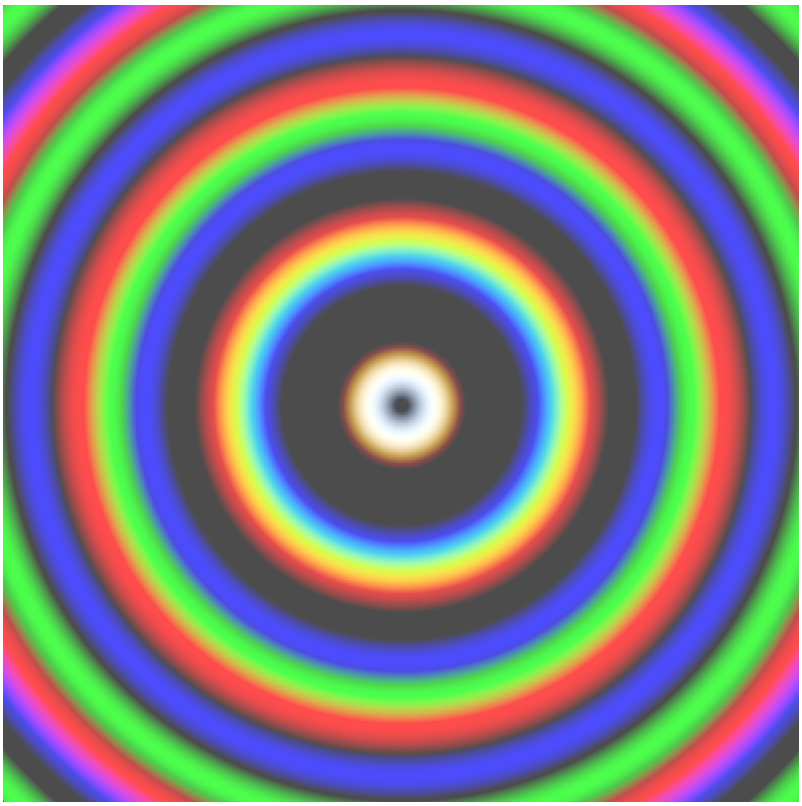

Figure 4: The radial sine wave used in the visualization.

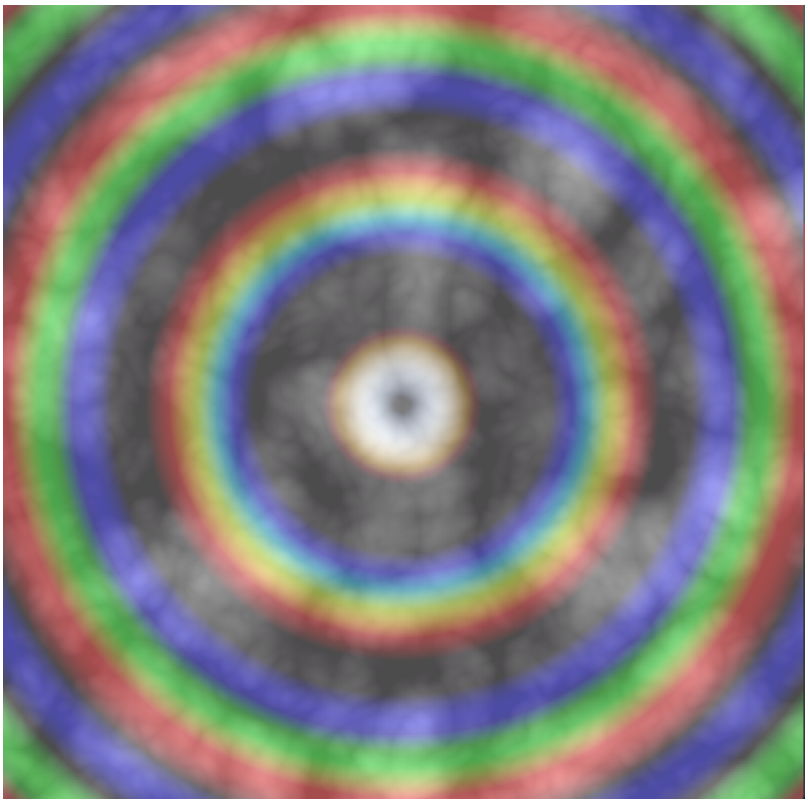

Figure 5: The flow noise visualization mixed with the radial sine wave visualization for a frame with lower amplitude and dominating frequencies.

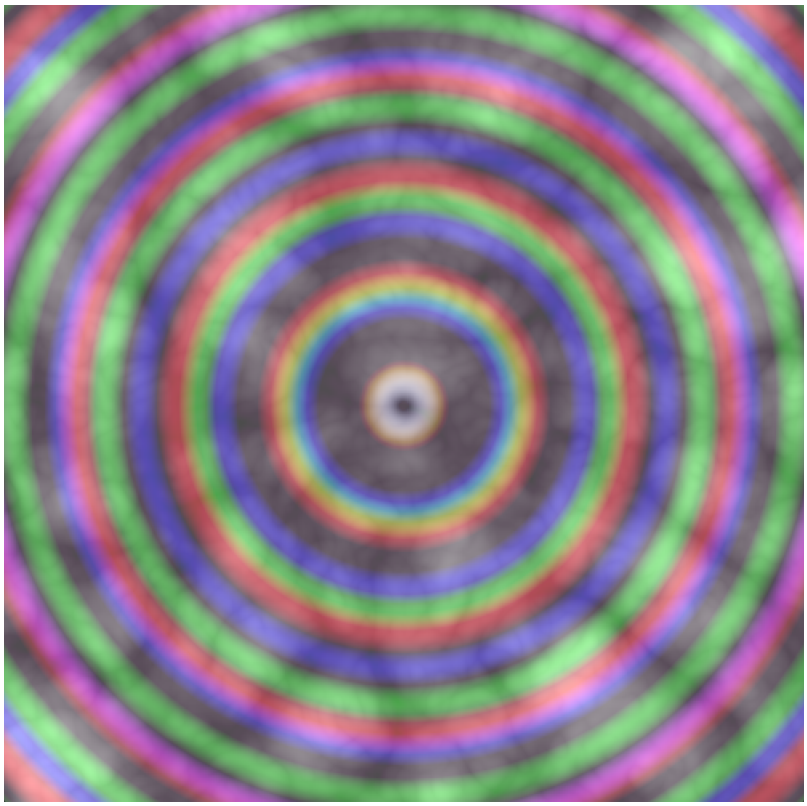

Figure 6: The flow noise visualization mixed with the radial sine wave visualization for a frame with higher amplitude and dominating frequencies.

5 Discussion

One of the problems with the visualization was the jaggedness in the animation. One solution that was attempted to reduce this was to take the average of the last five normalized amplitude and frequency values. The animation is still jagged and this is something that should be improved.

One challenge with this project was the way the sound should be visualized. Having the radial sine wave visualization gives a clear visualization of the frequency and amplitude variations in the sound. Thinner lines means a larger amplitude and higher frequency, thicker lines means a lower amplitude and frequency. This visualization is similar to the appearance of a speaker which is fitting for a sound visualization.

The performance of the application is not ideal and it could probably be rewritten in a more efficiently in a lower level language such as C++. Python was chosen to minimize the amount of time spent on the audio analysis part of the project. Since Python has libraries dedicated to audio analysis such as librosa it seemed fitting to use Python.

The audio analysis part could have been extended. Instead of taking the average of the five most dominating frequencies in each frame, the energy in a number of pre-defined frequency bands could be decided. The visualization could then vary on a number of frequency variables instead of just one. The visualization could also have separate visualizations for frequency and amplitude. Right now it is a bit difficult to discern if it is the amplitude or the frequency affecting the visualization.

References

[1] D. S. Ebert, F. K. Musgrave, D. Peachey, K. Perlin, and S. Worley, *Texturing & Modeling: A Procedural Approach*. Morgan Kaufmann, 2003.

[2] K. Perlin, "An image synthesizer," *ACM Siggraph Computer Graphics*, vol. 19, no. 3, pp. 287–296, 1985.

[3] "Smoothstep." `https://en.wikipedia.org/wiki/Smoothstep`. Retrieved: 2023-01-10.

[4] "Perlin noise." `https://en.wikipedia.org/wiki/Perlin_noise`. Retrieved: 2023-01-10.

[5] "Attribution-sharealike 4.0 international (cc by-sa 4.0)." `https://creativecommons.org/licenses/by-sa/4.0/deed.en`. Retrieved: 2023-01-10.

[6] "Python." `https://www.python.org/`. Retrieved: 2023-01-10.

[7] "librosa." `https://librosa.org/doc/latest/index.html#`. Retrieved: 2023-01-10.

[8] "Pyopengl." `https://pyopengl.sourceforge.net/`. Retrieved: 2023-01-10.

[9] "Pygame." `https://pypi.org/project/pygame/`. Retrieved: 2023-01-10.

[10] "TNM084-Project." `https://github.com/Rubenbromee/TNM084-Project`. Retrieved: 2023-01-10.