National Technical University of Ukraine
"Ihor Sikorsky Kyiv Polytechnic Institute"
Institute of Atomic and Thermal Energy
Department of digital technologies in energy

METHODS OF SYNTHESIS OF VIRTUAL REALITY

Calculation and graphics work

Executed by:
Syshylnykov Kyrylo
TR-23mp

2023 year

Chapter describing the task

In this calculation and graphic work, the task consists in the following:

- reuse the code from practical assignment #2;
- implement sound source rotation around the geometrical center of the surface patch by means of tangible interface (the surface stays still this time and the sound source moves). Reproduce your favorite song in mp3/ogg format having the spatial position of the sound source controlled by the user;
- visualize position of the sound source with a sphere;
- add a sound filter (use BiquadFilterNode interface) per variant (Variant 19 - Shelf filter of high frequencies). Add checkbox element that enables or disables the filter. Set filter parameters according to your taste.

Theory

The Web Audio API is a powerful JavaScript API that enables audio processing and manipulation in web applications. It provides a wide range of audio processing functionalities, allowing developers to create interactive and dynamic audio experiences directly in the browser.

One theory related to the Web Audio API is the concept of real-time audio synthesis. Real-time audio synthesis involves generating sound in real-time by manipulating audio signals and parameters. The Web Audio API supports this theory by providing a comprehensive set of audio nodes and audio processing methods.

Audio synthesis can be achieved by creating a network of audio nodes and connecting them together to form a signal flow. Each audio node represents a specific audio processing operation, such as oscillators for generating waveforms, filters for modifying frequency content, and gain nodes for controlling volume.

By connecting these nodes in a specific configuration, developers can create complex sound synthesis systems. For example, they can create a synthesizer by connecting an oscillator to a filter and then routing the output to a gain node. The oscillator generates the basic waveform, the filter modifies the frequency spectrum, and the gain node controls the overall volume.

Another aspect of this theory involves the concept of modulation. Modulation refers to the process of dynamically altering audio parameters over time. The Web Audio API supports modulation through audio parameters that can be connected to audio nodes. For example, an oscillator's frequency can be modulated by another oscillator to create vibrato effects.

In addition to synthesis and modulation, the Web Audio API theory also encompasses audio effects processing. Developers can apply various audio effects to audio signals in real-time. This includes reverberation, delay, distortion, and more. These effects can be achieved by connecting audio nodes that implement the desired processing algorithms.

A high-frequency shelf filter is a type of equalization (EQ) filter that affects frequencies above a specified cutoff frequency. It allows to boost or attenuate the amplitude of high-frequency content in an audio signal. It is used to adjust the amplitude (volume) of high-frequency content in an audio signal.

The term "shelf" refers to the shape of the filter's magnitude response curve, which resembles a shelf. Below the cutoff frequency, the response is typically flat, meaning it has no effect on frequencies in that range. However, above the cutoff frequency, the filter gradually boosts or attenuates the amplitude of the high-frequency content.

Shelving filters, including high-frequency shelf filters, are based on the concept of a magnitude response curve. This curve represents how the filter affects the amplitude of different frequencies in the audio signal. In the case of a shelf filter, the curve has a flat response below the cutoff frequency and a gradual change in gain above the cutoff.

High-frequency shelf filters are commonly used in audio production and sound engineering for various purposes: Tonal balance, De-essing, Frequency shaping, Mastering.

By manipulating the cutoff frequency and gain settings of a high-frequency shelf filter, audio engineers and producers can fine-tune the tonal characteristics of a sound or mix, ultimately achieving the desired sonic result.

To implement a high-frequency shelf filter, the Web Audio API provides the BiquadFilterNode, which is a versatile filter node capable of implementing different types of filters, including shelving filters.

The BiquadFilterNode is controlled using the following parameters:

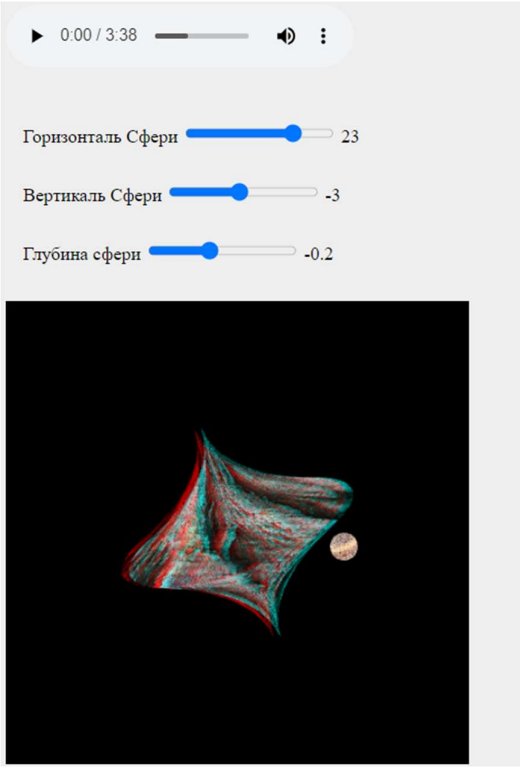
Type: Determines the type of filter. For a high-frequency shelf filter, set the type to "highshelf."

Cutoff frequency: Specifies the frequency above which the filter affects the signal. In the case of a high-frequency shelf filter, it is the frequency at which the shelving effect begins.

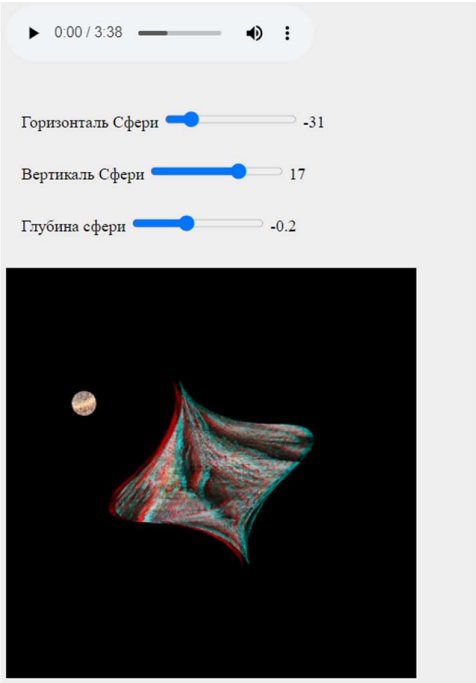
Gain: Controls the amount of gain applied to frequencies above the cutoff frequency. A positive gain boosts the high-frequency content, while a negative gain attenuates it.

To set the cutoff frequency and gain values, typically need to convert them to values suitable for the BiquadFilterNode. This involves normalizing the cutoff frequency to the range between 0 and 1, where 1 represents the Nyquist frequency (half of the sample rate), and setting the gain value in decibels (dB).

Chapter of user's instruction with screenshots



The work uses numerical input elements, with the help of which you can visually and easily change the coordinates of the sphere relative to the figure.

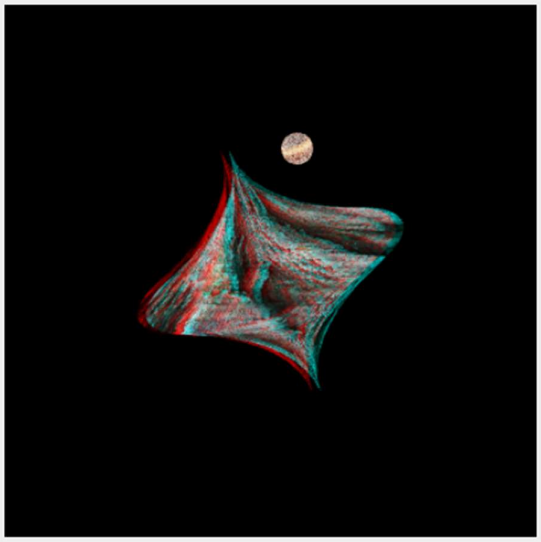


With the help of numerical coordinates, the position of the sphere along the abscissa, ordinate and applicate axes is changed.

Горизонталь Сфери 5

Вертикаль Сфери 23

Глубина сфери 0.5



Source Code

The code for writing a shelf high-pass filter:

```
xPositionInput.addEventListener("input", () => {
   draw();
 });

 yPositionInput.addEventListener("input", () => {
   draw();
 });

 zPositionInput.addEventListener("input", () => {
   draw();
 });

 spaceball = new TrackballRotator(canvas, draw, 0);

 if ("DeviceOrientationEvent" in window) {
   window.addEventListener("deviceorientation", handleOrientation);
 } else {
   console.log("Device orientation not supported");
 }

 audio = document.querySelector(".audio");
 audio.addEventListener("pause", () => {
   audioElement.resume();
 });

 audio.addEventListener("play", () => {
   if (!audioElement) {
     audioElement = new (window.AudioContext || window.webkitAudioContext)();
     audioSource = audioElement.createMediaElementSource(audio);
     audioPanner = audioElement.createPanner();
     highShelfFilter = audioElement.createBiquadFilter();
     audioPanner.panningModel = "HRTF";
     audioPanner.distanceModel = "linear";
     highShelfFilter.type = "highshelf";
     highShelfFilter.frequency.value = 8000;

     audioSource.connect(audioPanner);
     audioPanner.connect(highShelfFilter);
     highShelfFilter.connect(audioElement.destination);
     audioElement.resume();
   }
 });
```

```
  let filter = document.querySelector(".filterCheckbox");


  filter.addEventListener("change", function () {
    if (filter.checked) {
      audioPanner.disconnect();
      audioPanner.connect(highShelfFilter);
      highShelfFilter.connect(audioElement.destination);
    } else {
      audioPanner.disconnect();
      audioPanner.connect(audioElement.destination);
    }
  });

  audio.play();

  rerender();
}
```