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.

In the context of the Web Audio API, a high-pass filter can be implemented using the available audio nodes and methods provided by the API. The theory behind this implementation involves understanding the principles of filtering and how to manipulate audio signals to achieve the desired effect.

A high-pass filter typically consists of a combination of filtering techniques, such as a series of delay elements and feedback loops. The specific implementation may vary depending on the desired characteristics and complexity of the filter. However, the basic idea is to allow higher frequencies to pass through while reducing or eliminating lower frequencies.

In the Web Audio API, a high-pass filter can be implemented using the BiquadFilterNode, which represents a second-order filter with customizable frequency and Q values. The BiquadFilterNode implements various filter types, including high-pass, low-pass, and band-pass filters.

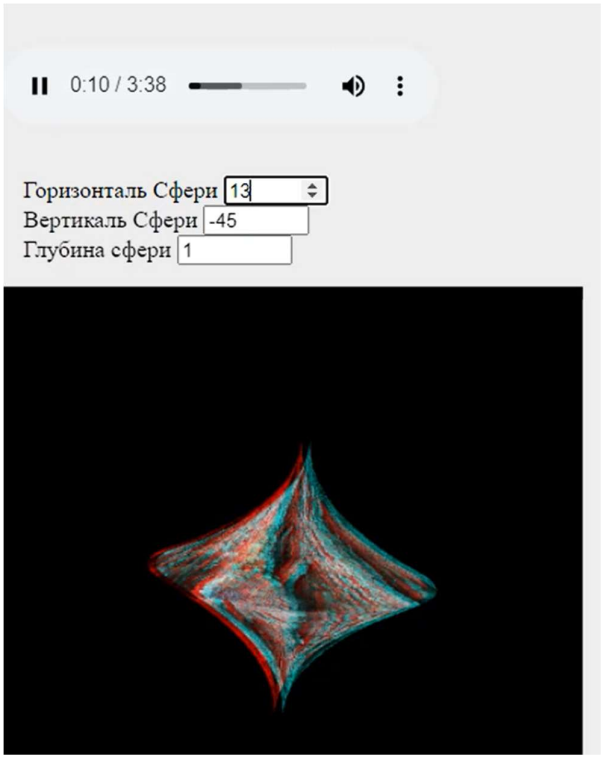
To create a high-pass filter, you would set the type of the BiquadFilterNode to 'highpass' and adjust the frequency parameter to determine the cutoff frequency below which the audio signal starts to attenuate. Higher frequencies above the cutoff frequency would pass through unaffected.

The Q value, also known as the quality factor, controls the steepness of the filter's roll-off curve. Higher Q values result in a narrower band of frequencies affected by the filter, while lower Q values produce a broader range of attenuation.

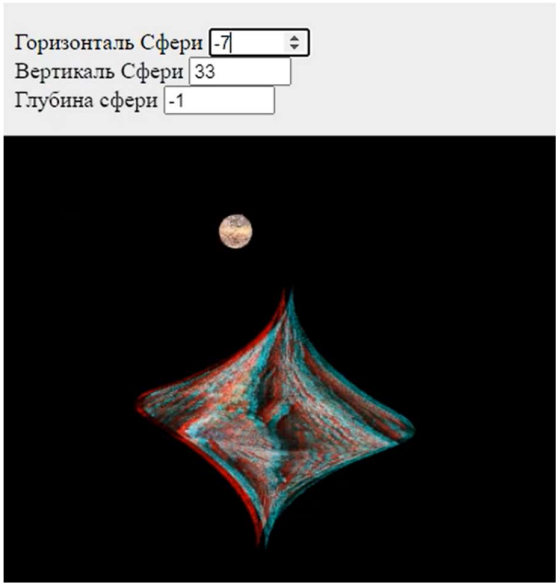
Once the BiquadFilterNode is created and configured, it can be connected to the audio graph by connecting its input and output to other audio nodes as needed. This allows you to integrate the high-pass filter into the overall audio processing pipeline.

By applying a high-pass filter to an audio signal in real-time using the Web Audio API, you can effectively remove or reduce low-frequency content, emphasizing higher frequencies and potentially achieving a desired audio effect or tonal balance.

Chapter of user's instruction with screenshots

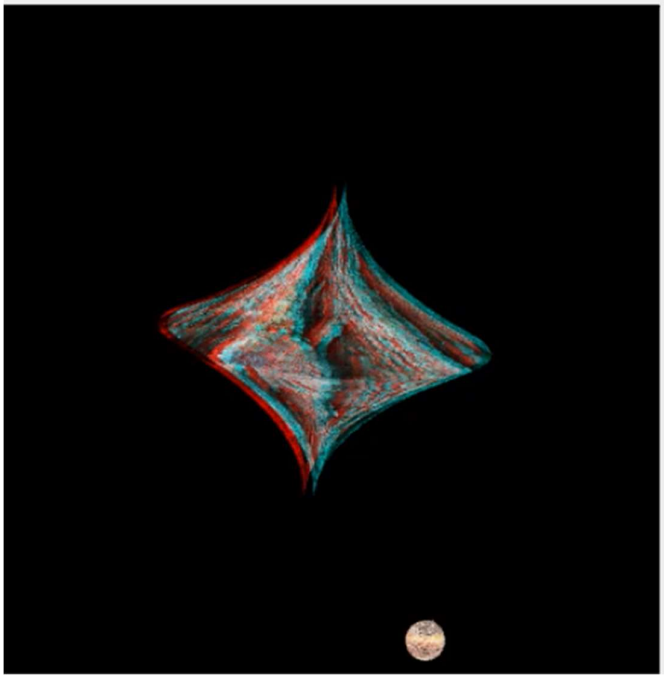


The work uses numerical input elements, with the help of which you can visually and easily change the coordinates of the sphere relative to the figure.



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

Горизонталь Сфери 15
Вертикаль Сфери -45
Глубина сфери -0,7



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();
}
```