# Scrolling Through Sound

## Scrolling as a method of interaction with audio on the web

Ehsan Ziya
Independent Researcher
London, UK
ehsan.ziya@gmail.com

## ABSTRACT

This paper aims to investigate the creative uses of scrolling as an interaction method for navigating through sound and music. Mainly focused on the use of granular synthesis, the paper explores the interaction model and technical challenges and presents a prototype as proof of concept to demonstrate a case in which scrolling can be used to create an immersive and interactive audio experience on the web.

Link to prototype: `zya.github.io/scrollsound`

## Categories and Subject Descriptors

H.5.5 [**Information Interfaces and Presentation**]: Sound and Music Computing; H.5.2 [**Information Interfaces and Presentation**]: User Interfaces

## General Terms

Experimentation, Design

## Keywords

Web Audio API,Granular Synthesis, JavaScript,Generative Music, Interactive Audio,Scrolling

## 1. INTRODUCTION

Audio, despite its relatively long history on the web, has been fairly static until recently. From the early `<bg-sound>`[1] to the more recent `<audio>`[2] element, audiences are used to listening passively to audio in a linear and static fashion, while having a limited amount of control over it. They could only play, pause or stop the audio or change the volume in some cases. The introduction of Web Audio API[3] opened up a breadth of new possibilities for changing this. Developers can now synthesise, manipulate, process and analyse audio directly in the browser, in real-time and without the need for third-party plugins. This has lead to creation of a series of interesting tools and demos. From full-fledged music production environments and synthesisers such as SoundTrap[4] , to games and audio visualisation experiments such as Weird Kids[5] by Cabibbo. Although the idea of interactive digital audio is not a new concept, ubiquity of browsers across platforms and the web's reach makes the web a powerful publishing platform for any interactive content including new forms of digital audio and music. However, the unique power of web audio is revealed when it is used in conjunction with other web technologies. Apps can leverage these technologies to create immersive, interactive and collaborative experiences. Integration with Canvas API[6], WebGL[7] and CSS animations[8] can lead to creation of audio-visual experiences. Standards such as WebRTC [9]and Web Sockets[10] provide good opportunities for development of collaborative environments and connected experiences. In addition to the mentioned standards, the web offers a set of APIs for handling user interaction with support for different types of user input such as keyboard, mouse, multi-touch and scrolling.

The interaction method we will focus on in this paper is scrolling. Despite its recent popularity in the visual design community with trending techniques such as parallax scrolling[11], this basic interaction tool has been so far overlooked by the audio community. Scrolling is browser's native method for navigating through content. Traditionally, it is used to scroll through text and images, from top to bottom or left to right. But It can also be used creatively to add depth and narrative to interactive experiences on the web. As mentioned before, the technique widely known as parallax scrolling is a solid example of this. Web designers have been using it as a way to take audiences through stories. The technique enhances user engagement by dynamically changing the visual properties of the page with scrolling events and revealing new elements as the user moves through the page. When it comes to audio, scrolling has the potential to be used for development of immersive and non-linear audio-visual experiences. By allowing users to move through sound smoothly, we can introduce an element of narrative and exploration to audio-visual experiences. This paper aims to investigate the techniques in which scrolling can be used as an interaction method for navigating through sound. Mainly focused on the use of granular synthesis, we present a prototype as a proof of concept which will be discussed in the following sections.

## 2. USECASES

### 2.1 Interactive Music Experiences

Since the introduction of Web Audio API in 2011, a series of experiments and demos have been developed, exploring the field of interactive music. Apps such as Flora Drift[12] have


*WAC '15* January 26 - 28 2015, Paris, France

experimented with generative and dynamic music where experiments such as George And Jonathan III[13] have focused on audio visualisation. But not a lot of experimentations have been done with manipulation of time and progression. The demos mentioned above either follow a traditional, linear time or do not have a notion of time and progression. For example a song is played from the beginning to the end and the user can see real-time visualisation of musical notes. Or the user is put in a timeless space where she can change different characteristics of the music by interacting with visual objects.

The method proposed in this paper aims to add a layer of narration and progression to interactive music using scrolling as the main user input. Where the user has the ability to go back and forth in a piece of music without interruption while being able to interact with visual objects to affect the sounds. Familiarity of users with the act of scrolling is a great advantage in creating interactive experiences since users will not need to learn new and complex interfaces and interaction models. By using this method, we can still leverage the dynamic nature of generative music while keeping the general sense of narrative and progression in music.

Scroll Sound[14] was developed as a proof of concept exploring the various ways in which scrolling can be used in addition to more conventional input types to create an interactive and dynamic music experience. The main interaction method used in the demo is based on mapping the scroll position to the playback position (Figure 1). Using the granular synthesis technique described in section 3.1, we enable users to move through the sound smoothly as they scroll down or up the page. In addition to the grains, we can use the scroll position to trigger certain events such as triggering notes, changing chords and changing AudioParam[3] values e.g. gain value and pan position. Scrolling will allow the users to move bi-directionally through the experience or stay in one section as long as they prefer.

## 2.2 Music Composition Tool

From the early uses of tape based granular synthesis techniques by Greek composer Iannis Xenakis in 1959[15] to the first software implementations of the technique by Curtis Roads[16] and to more recent tools such as Paul Stretch[17], the method has been widely used for music composition throughout the years. Granulation is capable of generating rich, evolving textures and musical soundscapes. Although the algorithm used in the prototype is basic in comparison to the others used in commercial products, it can still be used to create tools for extreme time-stretching. The user could use a combination of automatic playback speed and scrolling to alter the playhead and could navigate back and forth through the sound for more musical expression if necessary.

## 2.3 Other

Projects such as Responsive Radio[18] by BBC Research and Development have investigated the possibilities for more customisable radio listening experience by allowing users to alter the length of programmes or control the volume of music and speech independently. The methods described in this paper can also be used in a similar way. For example the user can move through a radio programme while different parts of the speech are triggered according to the scrolling position.

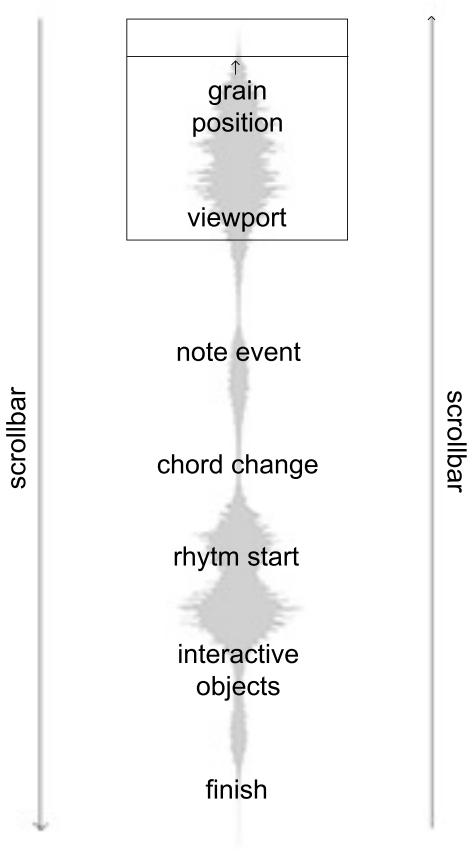


**Figure 1: Scrolling and audio events**

One of the other fields where scrolling is being used an interaction method is storytelling and branding. Apple's iMac with Retina web site[19] is a concrete example of using scrolling creatively to enhance user engagement with a brand. The same methods can be used for audio in order to create multi-sensory experiences to tell stories on the web.

The method can also be used in music education where students can move through a piece of music where notes are triggered and visualised helping them to study any given piece of music interactively.

# 3. TECHNICAL OVERVIEW

## 3.1 Granular Synthesis Engine

The main technique that allows us to move through sound smoothly is granular synthesis. The process of sampling short pieces of audio (1 to 200ms) known as grains and playing them back within short intervals will allow us to navigate through the sound file without any interruption. This technique is a powerful audio manipulation system that can be used to adjust the speed, pitch or other characteristics of a given sound independently and in real-time[20]. An example of this technique can be seen in HTML5 Granular[21]where the mouse or touch position is mapped to the grain position and gain. The code snippet below (Listing 1) is the simpli-

fied version of the grain function which is responsible for creation and playback of a grain according to the position in the audio file[22]. This function will be called using `setInterval` with the updated `offset`. In the case of the prototype presented here, offset is calculated according to scrolling-y position and duration of the sound file (See Listing 2). After loading an audio file and decoding it using `decodeAudioData` we can read it as many times as needed using the `BufferSource` node. This method is reasonably performant and it will provide us with great flexibility for creating large numbers of grains and layering them for smoother sound. As you can see, the offset is then randomised with a small margin each time before being used in `s.start` for a smooth representation of sound. Otherwise we would encounter glitches when the position is static.

```
function grain(ctx, bfr, dest, a, r, offset){
 var s = ctx.createBufferSource();
 var g = ctx.createGain();
 s.buffer = bfr;
 s.connect(g);
 g.connect(dest);
 var random = ((Math.random() * 0.3) - 0.15);
 var now=ctx.currentTime;
 s.start(now,randomisedOffset,now+a+r);
 g.gain.setValueAtTime(0, now);
 g.gain.linearRampToValueAtTime(1,now + a);
 g.gain.linearRampToValueAtTime(0,now + a + r);
 s.stop(now + a + r);
}
```

**Listing 1: Grain Function [22]**

As mentioned before, then the grain function will be called with an updated progress value as `offset`. We are aware that `setInterval`[23] is not a reliable option for precise timing and scheduling since the timing is sensitive to other events in the execution thread [24]. However, in this case, the exact timing of events is not an important factor and the fluctuations in the speed will not have an audible effect as long as the events are fired frequently in short intervals. Every 40ms in this case. The limitation of this method is that using `setInterval` could potentially affect the animation frame rate if Javascript and `requestAnimationFrame`[25] are used for drawing.

```
setInterval(function() {
// scales input range to output range
 var offset=map(prog,0,1,0,bfr.duration);
  grain(ctx, bfr,master,0.3,0.5,offset);
 }, 40);
```

**Listing 2: setInterval and offset calculation**

## 3.2 Interaction Model

There are several input types widely used for scrolling on the web. Namely mouse wheel, keyboard arrow keys and touch events. In order to use scrolling for moving through sound and updating the progress, we need to keep track of the scroll position. This can be achieved using the `onscroll`[26] events and updating the progress using `window.pageYOffset` or properties such as `element.scrollTop`[27]. However, there is a limitation to this approach since mobile browsers minimise script execution during scrolling. This will result in unresponsive user experience. One of the solutions to solving this issue is using the `touchdrag` events to emulate scrolling behaviour and keeping track of progress manually. This task is non-trivial and is out of this paper's concern, but there are libraries that we can use to address this issue.

## 3.3 Scrolling Libraries

There are several libraries available that address the issue of cross platform scrolling with slightly different approaches. `iScroll.js`[28] and `scroller.js`[29] resolve this issue by using the same mechanism in all platforms by even replacing the scrolling mechanism in desktop with a custom implementation. They both have JavaScript APIs for controlling events and defining animations where `skrollr.js`[30] focuses on declarative HTML approach by keeping CSS animation definitions to the element by using `data` attributes. Skrollr does not replace the default scrolling mechanism in desktop while offering a good user experience in mobile. Skrollr is particularly suitable for our use-case, since it will let us focus on audio. It offers keyframe event handling which can be used to fire audio events such as sound playback in certain parts of the page, changes in gain or other `audio params` and synchronising audio events with animations.

## 3.4 Event Handling

The code snippet below demonstrates the way in which Skrollr element declaration works. In this example opacity of the element `example` will be 0 when top of the element is at the top of the viewport and will be animated to 1 when bottom of the element hits the top of the viewport. The last data attribute allows us to receive keyframe events using the keyframe event handler(Listing 4). For example, when `data-top-bottom` occurs, we receive an event which can be used to trigger sounds or updating `AudioParams`. For example, we can use the `opacity` value of an element to change the `gain` value of an audio node providing an audio-visual feedback to the user.

```
<div id='example'
  data-top="opacity:0"
  data-top-bottom="opacity:1"
  data-emit-events
></div>
```

**Listing 3: Skrollr element declaration**

The code snippet below demonstrates `skrollr`'s initialisation method. There are two types of useful events for manipulating audio dynamically according to scrolling position. Render is a callback method for every time a change occurs in the scrolling position with useful properties such as `curTop` and `maxTop` which we use to calculate the progress value. And as mentioned before, keyframe is an event handler for keyframe events.

```
s=skrollr.init({
  smoothScrolling: true,
  render: function(e){
   //calculating the progress as a percentage
    progress = e.curTop / e.maxTop;
  },
  //the keyframe hanlder
  keyframe: keyframeHandler
});
```

**Listing 4: Skrollr initialisation and event hanlders**

Keyframe events will be called with arguments such as name of the element, name of the keyframe and the direction in which the user is scrolling. We can match patterns against these arguments to trigger specific events in relation to each event. For example, if the element `x` enters the viewport from the bottom, then play a certain note and start a CSS animation.

# 4. CONCLUSIONS AND FUTURE WORK

The proposed method of interaction encourages music listening on the web as a more participative activity as opposed to a passive and linear experience. By using this system, we can keep a balance between the familiar sense of narrative and the dynamic nature of generative music. A prototype has been presented making use of granular synthesis and demonstrating an interactive music experience where scrolling is the main interaction method.

The primary aim of this paper has been to address the technical challenges in using scrolling to navigate through sound. A granulation system was developed and issues such as cross-platform availability, multiple input support and performance were covered. However, the creative possibilities based on this approach need to be explored further.

An important area of future exploration is usability. Initial user testing on the prototype shows that given the different types of input such as mouse wheel, trackpads and touch, users may have different experiences due to different implementations. For example, using trackpads or touch screens will result in a fast progression while using the arrow keys or the mouse wheel will have significantly slower speeds. The system can benefit from more systematic user studies to address this issue by finding optimal values for speed or even allowing users to customise the experience by adjusting the overall length of the piece or the progression speed.

Future technical work may also include further refinement of the interaction system for a better performance across platforms as well as further development of the granular engine.

# 5. REFERENCES


[1] <bg-sound>. https://developer.mozilla.org/en-US/docs/Web/HTML/Element/bgsound. Accessed: 2014-10-07.

[2] <audio>. https://developer.mozilla.org/en-US/docs/Web/HTML/Element/audio. Accessed: 2014-10-07.

[3] Web audio api. http://webaudio.github.io/web-audio-api/. Accessed: 2014-10-07.

[4] Sound trap. https://www.soundtrap.com/. Accessed: 2014-10-07.

[5] Weird kids. http://cabbi.bo/weirdkids/. Accessed: 2014-10-07.

[6] Canvas api. https://developer.mozilla.org/en-us/docs/web/api/canvas_api/. Accessed: 2014-10-07.

[7] Opengl es 2.0 for the web. https://www.khronos.org/webgl/. Accessed: 2014-10-07.

[8] Css animations. http://dev.w3.org/csswg/css-animations/. Accessed: 2014-10-07.

[9] Webrtc 1.0: Real-time communication between browsers. http://www.w3.org/TR/webrtc/. Accessed: 2014-10-07.

[10] Web sockets. https://developer.mozilla.org/en/docs/WebSockets. Accessed: 2014-10-07.

[11] Best websites with parallax scrolling of 2013. http://goo.gl/xajPUR. Accessed: 2014-10-07.

[12] Flora drift by white vinyl design. http://whitevinyldesign.com/floradrift/. Accessed: 2014-10-10.

[13] George and jonathan iii. http://www.georgeandjonathan.com/#1. Accessed: 2014-10-10.

[14] Scroll sound source code. https://github.com/zya/scrollsound. Accessed: 2014-10-20.

[15] Xenakis - analogique a + b. https://www.youtube.com/watch?v=DTzOWKaDrVI. Accessed: 2014-10-10.

[16] Curtis Roads. *Microsounds*. MITPress, Cambridge, Massachusetts, 2001.

[17] Paulstretch by paul nasca. http://hypermammut.sourceforge.net/paulstretch/. Accessed: 2014-10-10.

[18] Responsive radio by bbc rd. http://www.bbc.co.uk/programmes/p026gcms. Accessed: 2014-10-10.

[19] Apple imac with retina display. http://www.apple.com/imac-with-retina/. Accessed: 2014-10-20.

[20] Granular synthesis - how it works and ways to use it. http://goo.gl/JE0MbN. Accessed: 2014-10-08.

[21] Html5 granular. http://zya.github.io/granular/. Accessed: 2014-10-07.

[22] Scroll sound grain function. https://github.com/zya/scrollsound/blob/master/js/grain.js. Accessed: 2014-10-20.

[23] setinterval. https://developer.mozilla.org/en-US/docs/Web/API/WindowTimers.setInterval. Accessed: 2014-10-10.

[24] A tale of two clocks - scheduling web audio with precision. http://goo.gl/gJuBue. Accessed: 2014-10-08.

[25] Requestanimationframe. http://goo.gl/raIzXb. Accessed: 2014-10-10.

[26] window.onscroll. https://developer.mozilla.org/en-US/docs/Web/API/window.onscroll/. Accessed: 2014-10-07.

[27] Element.scrolltop. https://developer.mozilla.org/en-US/docs/Web/API/Element.scrollTop. Accessed: 2014-10-07.

[28] iscroll.js by matteo spinelli. https://github.com/cubiq/iscroll/. Accessed: 2014-10-07.

[29] Scroller.js by zynga. https://github.com/zynga/scroller. Accessed: 2014-10-07.

[30] Skrollr.js. https://github.com/Prinzhorn/skrollr. Accessed: 2014-10-07.