Live streaming platform as an instrument,experience and pastime

Louis Foster
louisfoster.com
Karlsruhe, Germany
contact@louisfoster.com

Abstract
This paper introduces a design and the implementation of a proof of concept for a sonic cyberspace. The purpose of this is to explore new media, and find potential in our existing technology and infrastructure. The central theme of this cyberspace is collective collaboration, and documenting the process of developing speculative creativity platforms. It is discovered some streaming technology, such as Icecast, is not suitable for more complex use-cases. The paper proposes an appropriation of modern streaming protocols, and discusses the potential of incorporating out-of-band metadata to explore unique applications of this design. The paper discusses how the attitude towards composition transforms when the ability to dominate experience is countered by randomness. Additionally, the design suggests only the creative experience can have no latency as well as a certainty of realness, questioning the relevance of real-time and live streaming for performance and collaboration in music.

1. Introduction
This paper introduces a design and implementation of a proof of concept for a sonic cyberspace. A sonic cyberspace is a hypermedia platform that comprises primarily of audio, rather than, more commonplace, text. The cyberspace can be collaboratively constructed using decentralised audio streams. An interface would allow the streams to be linked together.

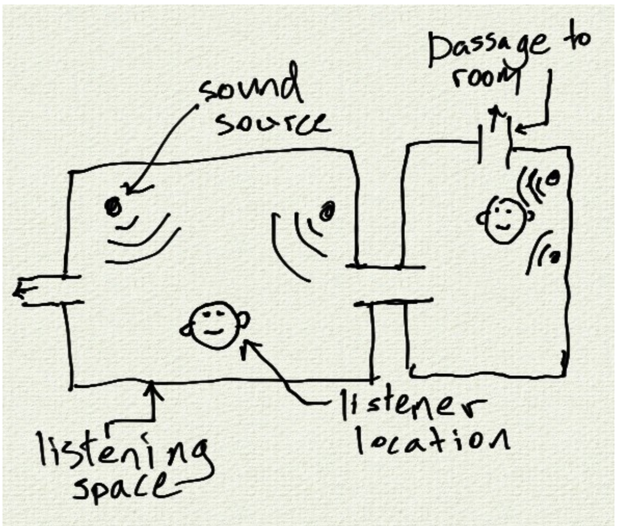
The purpose of this is to explore new media, and find potential in our existing technology and infrastructure, partially as a response to the abundance of generative adversarial network automated content. This project attempts to explore hybrid implementations of existing technology, "lateral thinking with withered technology" [11], and forge new ways of interacting, interfacing, and building new media for this, "we shape our building and afterwards our buildings shape us" [12] in order to produce semi-automated music.

The central theme of this cyberspace is collective collaboration. The intention to build this project came before the COVID-19 pandemic [43]. I speculated that a changing global environment, and possible shifts in social conditions, could lead to a greater desire or requirement for people to remain within their homes.

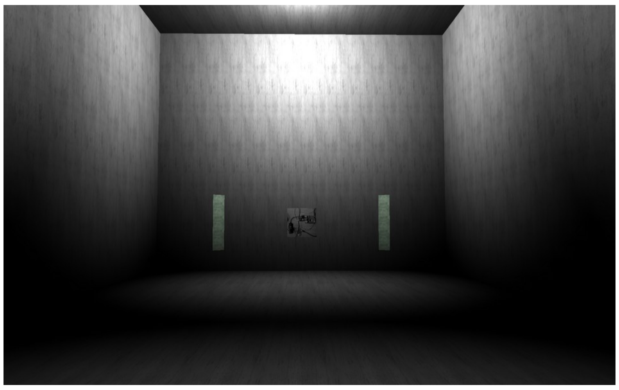
Prior to and during the pandemic, gaming has been continually increasing as a popular pastime for people [1]. The trend in gaming has been towards increasingly immersive design [38], which has likely contributed to competing media improving its immersiveness [42]. The sonic cyberspace, as a virtual "reality," is intended to participate in this space as a medium for exploring sound in an engaging, entertaining, and informative way, while simultaneously providing a powerful telepresent art platform.

2. Early Explorations
The journey towards something that could be comparable with the cyberspace known as the "world wide web" began with building two different models of explorable audio.

2.1 Harsh room
The first model of explorable audio, harsh room, was a 3D space where you are encouraged to explore rooms and discover 3D positioned audio. The design intended for people to upload sounds and get a "room" that they could place the sounds into spatially by assigning it to an object with three dimensional coordinates. Other people can then virtually enter a room and listen to and interact with the sounds.

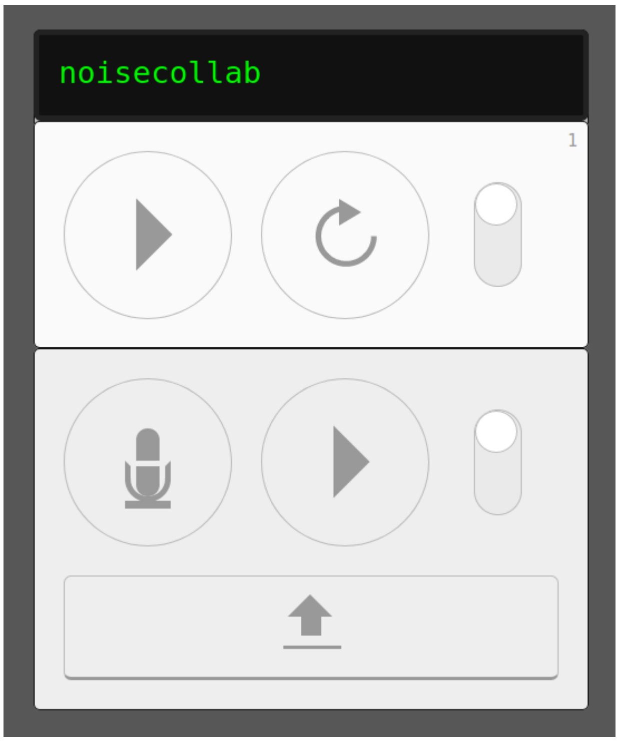


Harsh room only got as far as creating a room and a hallway and having example audio in the room. 3D graphics in the browser are still very limited in terms of what is possible and what devices can really "handle" it, plus it was time consuming to really get to the final "goal." People's response to seeing this seemed to generate more interest in the spatial audio exploration rather than the 3D space, which although was not very interesting visually, it also seemed to be less important than the experience of the sound.



2.2 Noisecollab

The feedback from designing and prototyping harsh room, and the unfulfilled objective of creating a collaborative system, led to the design and implementation of noisecollab. Noisecollab involved uploading short audio snippets to a server, which would insert the snippet into an existing 10 second audio loop. After the file is updated on the server, the user's interface updates to play this new updated loop.



Due to inexperience, I could only figure out how to upload raw PCM data at the time, so the files were pretty large even for 1-10 second files, so for these purposes I concluded it would not be scalable. The UI was minimal. Some people found the idea interesting, however the effects applied to the audio made it too loud and somewhat unlistenable.

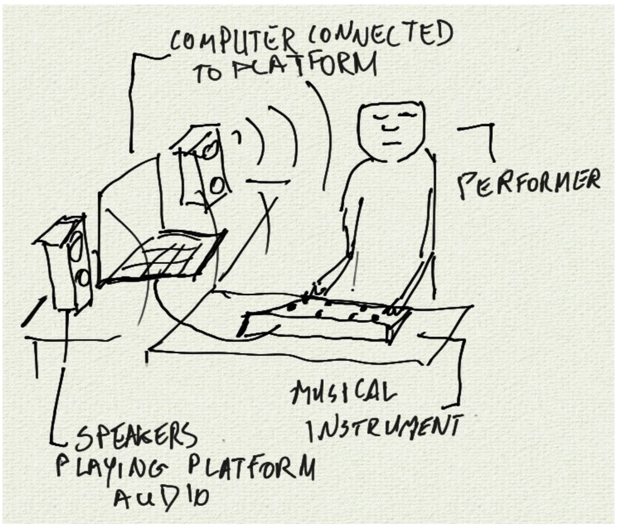
Noisecollab led to focusing on the idea of streams of audio, as well as improving the ease of access to and downloading of audio. As I used noisecollab, the interaction with the system and manipulation of sound within it felt as though it was a kind of instrument, even if it was with a high latency. This was an instrument many people could play simultaneously, and for each player it would be slightly unique.

3. Design

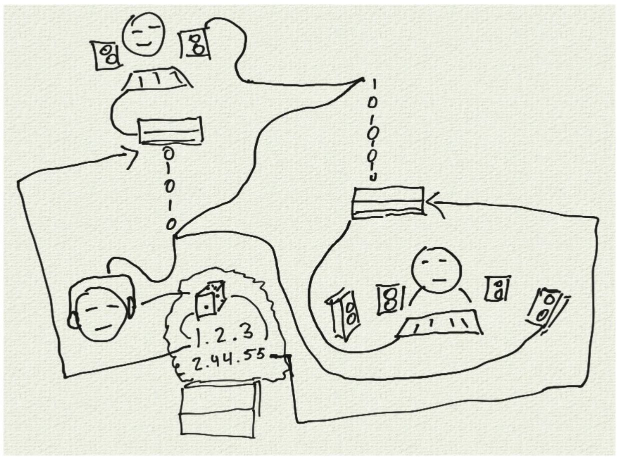
3.1 Concept

When designing harsh room, I envisioned the usage of web audio's 3D spatialisation for separate audio streams [22]. However, after attending music performances at Kraftwerk [16] and Geoff Stern Sonic Space [10], I chose to de-virtualise the spatialisation of audio. Instead of multiple streams being combined into a single piece or having a virtual location, speakers (one for each output channel) could be attached to an audio interface and placed around a room. Initially, there were two designs for achieving this, the first would use multiple computers with standard stereo audio interfaces. The second design was a single computer with an audio interface capable of providing more than 2 channels.

Each channel is to receive an individual, continuous audio stream. When using the platform, some source of audio would be always available, whether pre-recorded or live. If it were to be live, this would involve a participant actively streaming audio from their device via a source within the device or externally. This performing participant might also be listening to streams from the platform, in which case there would be potential for both an auditory and cognitive feedback loop, similar to a standard jam session.


COMPUTER CONNECTED TO PLATFORM
PERFORMER
MUSICAL INSTRUMENT
SPEAKERS PLAYING PLATFORM AUDIO

The design identified 3 separate components for this to work with multiple users. The first component enabled providing audio to others via a server being fed the encoded data stream of a raw audio source. The second component was a server allowing people to share and find the IP address and port of the servers in the first component. The addresses would be added to a list when they're available for streaming and removed when they stop. The third component requests locations of streaming servers from the address server, stream the encoded audio data, decode the data, and send it to audio output channels.



3.2 Early prototypes

Icecast [13] was chosen for the streaming server, as this technology is free, documented, and appeared to be a common choice when searching for streaming technology. A basic web server, written in any language, would handle the streaming server address lists. To receive, decode, and output the stream, Sound Exchange (SoX) [36] was chosen as it is a mature, versatile tool that appeared to be capable of this functionality. The automation and management of these components was to be handled by any scripting language, such as Bash or Python.

This design, although it worked within a local network, couldn't be effective with multiple participants for various technical and social reasons. Icecast servers continuously keep an HTTP socket open and if something interrupts this connection then the client needs to reconnect. If there is no more audio to stream, unless the audio is being preserved somewhere locally or there's a fallback stream, the client will stop receiving audio. This means there would need to be additional infrastructure, and complexity, to provide a source of audio data when Icecast doesn't have an active input.

The Icecast server requires a dedicated remote server, rather than local instance. The reason for this is security, as many people's home routers and Internet Service Providers use firewalls to prevent machines outside of the home network from connecting directly to a user's computer. This may be overcome, but it becomes technically complex and the average user might not be comfortable doing this, nor is it a good idea.

To send audio to a remotely hosted Icecast server, the user providing the input audio would need to use additional software, such as Darkice [4], again increasing the complexity of the overall system. SoX also had limitations with handling multi-channel output streams and required additional software, such as JACK audio [15]. This was difficult to set up on various systems, such as a standard modern Macbook running MacOS or a Raspberry Pi running a headless Linux distribution.

These issues culminated in a redesign of the platform such that it could operate the streaming and discovery servers as services, while the input and output, encoding and decoding, would be easier to manage. The first re-design intended to use Python and audio libraries [33] for handling the multi-channel output and input, encoding and decoding the data, and communicating with the servers. The location server wouldn't change and, at this stage, Icecast was still the preferable streaming option.

To solve the issue of a stream becoming unavailable, the Python program could also store previously streamed audio locally and replay the audio from the file when necessary. However, as the complexity of multiple streams, files, channels, audio devices and servers compounded, and as limited experience with concurrent and threaded Python slowed the process, it became too difficult to achieve something robust. I presented the ideas and designs to other people, and there was a general negative response to the level of complexity of setting up software.

The technical aspects were overwhelming and made the barrier to entry too high, even for people who might be familiar with the technology. Attempting to solve these issues by building part of the platform as a web application came after further research, design improvements, and test participants' feedback. The browser can trivially fetch the location addresses [20], receive data from an Icecast server [14] and stream audio to individual channels [19].

There appeared to be a lack of useful resources for sending encoded streams of data via the required "PUT" request to an Icecast server via the browser. When trying to implement this, it was clear it would not be possible. After some more research, the reason there is no prior art, is that it is impossible to implement due to the streaming limitations of the fetch API [2].

One alternative, avoided due to technical restraints, is to implement WebRTC and either stream audio directly between users, or stream the data to a remote or local server and handle it somehow for further streaming. Another option is to utilise the same ideas behind HLS [29] and DASH [7], by turning small segments of the audio input buffer into individual files. These small files are sent to a server and accessed by other users by fetching the files, decoding them and putting the decoded data into an audio buffer.

4. Data and Metadata

4.1 Existing standards

Hypermedia implies the ability to markup and annotate pieces of information, so audio streams with this kind of metadata enables augmentation and interaction. This metadata could be a URL hyperlink, where a whole or part of an audio stream would also be a portal between content. Information like spatial movement or additional signal processing effects, could be authored by anyone and designed for various contexts and tools.

It is difficult to find reference material of how to do this with Icecast, and how the various encoding containers would be able to combine metadata. It is technically possible with OGG [27], however I couldn't find any useful information for authoring or accessing this data for an Icecast stream, and technology such as Annodex [32], designed for this purpose, is now obsolete. Providing a separate stream of data, that is out-of-band, is the preferred option, and WebVTT [23] replaced Annodex as the standard format for this purpose.

4.2 A novel approach

The implementations of HLS/DASH were designed for standard, linear playback, with a focus on video and adapting to varying bandwidth. For this project, a simpler design would use JSON [8] format and only indicate the audio file segment URLs, and an audio encoder and decoder either from a native browser API or a WASM-compiled tool [21] within a web worker. Out-of-band metadata would align data points with the content-addresses of individual audio file segment URLs, instead of a timestamp.

5. Randomness

5.1 Intention for randomness

Randomness has been utilised in this project for a number of reasons. The first is its relationship to the "cut-up" method [41]. In music, cutups are the act of taking ribbons of recorded cassette tape, literally slicing it to pieces and carefully sticking the pieces together in a random order [27], which was similar to the literary version involving slices of newspaper and other reading material, or visual arts which might encroach on the concept of collage and assemblage.

Further, this randomness is a playful act. Musique Concrète [26], an earlier incarnation of this composition technique described this improvised structuring as playful, just as one might play a musical instrument. Play is an important part of learning, a feedback loop helping the individual explore the potential of a medium, tool, and idea, without fearing failure or severe repercussions for misuse or breaking conventions and traditions [30]. Randomness provides a kind of "escape hatch" from our expectations, "making (assembling) and unmaking (disassembling) are viewed as one and the same: part of the same cyclic process of discovery. And making becomes a critical process from which to reflect, reinvent and rejuvenate and not just a means to an end" [28].

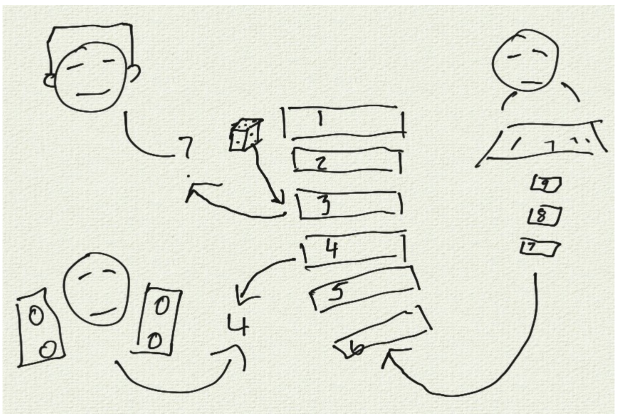
For the purposes of this platform, randomness provides a vector for exploring composition variations when removing the capacity for an individual's influence during collaborative music creation. In a traditional improvised jam session [24], there is often adherence to specific song structures, or standards [25], which serve the purpose of enabling the overall sound and performance to resemble that of a "piece" of music, where a particular "feel" to the music is desired. Therefore, an intention of this project is to observe how attitude of the individual towards playing and composition transforms when the ability to dominate the perception of the piece with traditional conventions and other social dynamics is countered by randomness.

5.2 Randomness from the performer

In order to achieve this, the platform includes multiple layers of computed randomness. This randomness prevents the performing participants from perceiving any composition on the platform identically at any given moment, nor will a listening experience be repeatable. Any individual is unable to predict what they will hear, and a performer's learned or chosen response will be non-deterministic as a result. This provides an opportunity to those who perform with unusual instruments, such as synthesizers or no-input mixers [34], which aren't always designed to be played in a traditional ensemble.

5.3 Randomness from the audio stream server

The audio server providing the audio segment file URLs incorporates randomness to decide the start point of a stream. If a listener requests a new sequence of audio files from a particular recording to be streamed, the initial file won't consistently correspond to the start point of the recording. However, to prevent extreme choppiness in playback, only the starting point is random. The ordering of a stream's individual audio segment files after this start point isn't randomised, so the continuous playback will be as it was recorded. The rationality for this decision is aesthetic as the audio segment files are around 1 second in length, and inconsistency at this level of granularity causes the sound to converge towards overtly random noise patterns, dominating the composition.



5.4 Randomness for the listener

When receiving a stream, there are additional sources of randomness. When there are multiple sources, only one is randomly chosen per output channel at any given moment at random intervals. The intervals are limited to minimum and maximum time frame to avoid the choppiness effect and avoid a particular source from playing so long as to overtly influence the sound.

The location server helps users discover stream servers by sharing their addresses within lists stored in hubs. These addresses can be added to and removed from a hub by a user at any moment, so there is no certainty whether a particular address will be available from a hub at any given time. When requesting an address from a hub, the selection is random, and it is returned as an HTTP redirect so a hub address can be used in place of a stream address.

6. Streaming latency, stream as an instrument

The performer provides audio to the platform as they play, and the recording is available to listeners moments later, but due to the design the sense of live streaming within the platform becomes untenable. Therefore, an intention of this project is to question the purpose and effect of realness and immediacy of real-time and low-latency streaming. There is no linearity, feedback from a moment ago could be integrated sooner than recordings from an hour ago.

A hypothesis for this project is, only the creative experience has no latency and a certainty of realness. This experience is unfiltered as it originates from within and in a sense original and unchanged with time. Therefore operating the platform, even as a listener, is considered “playing” a musical instrument, as in David Tudor’s notion of composing inside electronics, that is to “release what’s there” [39], because the individual’s use of the platform is unique and their choices of interaction directly affect what is produced.

7. Technology

7.1 Typescript

The language of the platform is Typescript [40], as it compiles directly to Javascript for running in the browser. Deno [6] is utilised for the server-side components, due to the support of Typescript and to test out the new technology, despite the preference of using mature technology. The high quality sound with small encoded file size [3] and availability of an existing encoder and decoder [35] for opus format are reasons it was chosen for the file type used in the platform, other formats were not tested.

7.2 Nginx and Content Delivery Networks

As regular HTTP/1.1 is used to serve the audio segment files, a static file server or content delivery network (CDN) can be used. Although it is a highly effective solution to possible scalability issues, the current demonstration of this platform doesn’t use a CDN. Instead, the audio files are accessed via an nginx static server, the implementation is simple and provides both good caching and performance.

7.3 Sqlite

The server applications only define a restricted amount of HTTP routes and methods and there is a relatively small amount of logic for handling of uploaded files and manage data. The database is sqlite [37], which isn’t the most powerful or a scalable system, but for the purposes of a proof of concept it is simple to implement and can be easily adapted to a more powerful relational database, such as Postgres [31], in future iterations.

7.4 Audioworklet

Initial implementations of the platform struggled to handle constant buffer updates for the audio data. The recording component now utilises the Audioworklet API [18], and will also be utilised in future iterations of the playback system. This API replaces the deprecated script processor node. Handling the buffer off the main thread reduces performance issues, as overloading this thread can contribute to reduced audio quality under heavy loads.

8. User experience and feedback

The primary feedback received by users was regarding the user interface. As the intention and mechanism of the platform was difficult to explain, the interface was not helpful as it provided few affordances. The original design utilised minimal interactive elements, words and inputs in order to preserve a kind of simplicity and reduce possible confusion. However, it had the effect of increasing confusion concerning the actions to be taken to achieve the goals described within testing sessions and workshops.

This feedback resulted in a major redesign and implementation of the user interface. The current design unifies the user interface into a single new component and attempts to provide a descriptive and goal-oriented approach, while still preserving minimalism, and is intended to be tested for further improvements. This update also included refactoring all component’s code bases, converting the previous client-side components into importable libraries, and includes the implementation of Audioworklets.

9. Workshop

9.1 Introduction to ideas

To commence the workshop, it is valuable to articulate what problems may arise when attempting to stream audio to other people around the world. Discussions around solutions known to workshop participants can help with encouraging the kind of thinking necessary when taking on complex problems. The experimental aspects of the platform, such as randomness and philosophy, are then introduced to propose uncommon requirements and problems the design attempts to solve.

9.2 Usage of existing technology

In conjunction with the introduction, demonstrations of more commonly understood technology are utilised as a stepping-stone metaphor towards how the platform was designed. This involves how people understand and use communication technology, such as phones and the Internet, and how items, such as objects or data files, are shared especially en masse. A visual demonstration of the cut-up technique is utilised to demonstrate the intention behind randomness.

9.3 Using the platform

As each concept is grasped by the workshop participants, its relative component is introduced. Upon introduction, the participants are invited to use the interface until they have a mental model of the platform. When the explanation of metaphors and explanation is completed, participants should be able to actively use and describe the platform.

9.4 Collaboration and feedback

The remaining time of the workshop should be spent creating a collective music making experience. There is high likelihood of questions relating to certain unclear aspects of the platform in order to help complete individual’s mental models. There will possibly be usability issues or bugs, which can be addressed during this time. Participants are encouraged to participate in further development of the platform and adjacent technology, as well as discuss what they hope will be added or improved.

10. Discussion

10.1 Implementation issues

Although the design included support for multiple output and input channels, an issue with supporting this feature in the browser prevented complete realisation of the project. The development of this project relied upon an affordable multi-channel audio interface, a generic 8 channel output and 4 channel input device based on the CM106 chip [5]. Although it was possible for Chrome browser to acknowledge the full 8 output channels, neither the ordering of channels nor playback was consistent.

It was difficult to determine if this problem was due to lack of support from the browser vendor, an issue with the operating system and drivers, or a problem with the device itself. Fortunately, an effective demonstration of the platform's concept was possible with the few channels that could be accessed, as well as with mono and stereo devices. The inconsistency and complexity of device support and a lack of users with multi-channel interfaces reinforced the value of virtually combining sources into a single channel, possibly with spatialisation.

10.2 Future development goals

- In-browser application-to-application communication via the symbiosome system [9].
- Metadata integration and authoring system.
- Non-random modes of usage allowing standard streaming and audio playback.
- Improvement of user experience and the user interface.
- Comprehensive documentation for the platform and code base.
- A community for software and hardware creation for integration with the platform.
- Workshops and collaborative music making sessions.

10.3 Questions for further investigation

- The appropriation of recordings and existing audio, de-structured through demolition and random assembly, allows for emergent and unexpected forms, should we disallow this form of exploration for licensing and fair usage compliance?
- If the platform enables immersion in the moment, despite a lack of true linear representation of a "now" or "near-now," is there a superior value to true immediacy within this technology?
- It might be argued, it's a given there's a lack of certainty of realness without direct experience, does this mean real-time and live streaming is merely a form of validation, and if so, what is the purpose of the external validation of a moment?

11. Acknowledgement

Acknowledgement of the support from friends who tested the platform, provided valuable feedback and encouraged the realisation of this project.

12. References

[1] *3, 2, 1 Go! Video Gaming is at an All-Time High During COVID-19.* (2020, June 3). Retrieved June 04, 2021 from https://www.nielsen.com/us/en/insights/article/2020/3-2-1-go-video-gaming-is-at-an-all-time-high-during-covid-19/

[2] Archibald, J. (2020, July 22). *Streaming requests with the fetch API.* https://web.dev/fetch-upload-streaming

[3] *Codec Landscape*. Retrieved June 4, 2021 from https://opus-codec.org/comparison/

[4] *DarkIce.* Retrieved June 4, 2021 from http://darkice.org/

[5] *Delock USB Sound Box 7.1.* Retrieved June 4, 2021 from https://www.delock.com/produkte/G_61803/merkmale.html

[6] *Deno*. Retrieved June 4, 2021 from https://deno.land/

[7] *Dynamic adaptive streaming over HTTP (DASH).* Retrieved June 4, 2021 from https://www.iso.org/standard/57623.html

[8] ECMA International. (December 2017). *The JSON Data Interchange Syntax.* https://www.ecma-international.org/wp-content/uploads/ECMA-404_2nd_edition_december_2017.pdf

[9] Foster, L. [Drohen]. (2021, January 24). https://github.com/drohen/symbiosome/

[10] *Geoff Stern Sonic Space.* Retrieved June 4, 2021 from https://www.geoffsternartspace.com/sonic-space/

[11] Gunpei, Y. (1997). *Yokoi Gunpei Game House.* ASCII

[12] *Hansard, United Kingdom Parliament, Commons, House of Commons Rebuilding, Speaking: The Prime Minister (Mr. Churchill)*. (1943, October 28). HC Deb 28, volume 393, cc403-73.

[13] *Icecast.* Retrieved June 4, 2021 from https://icecast.org/

[14] *Internet Radio.* Retrieved June 4, 2021 from https://www.internet-radio.com/

[15] *JACK audio.* Retrieved June 4, 2021 from https://jackaudio.org/

[16] *Kraftwerk Berlin.* Retrieved June 4, 2021 from http://kraftwerkberlin.de/en/

[17] Lewis, S. (May/June, 1992). *Human Rites: Coil's Agony and Ecstasy*, Option *No. 44.* http://brainwashed.com/common/htdocs/publications/coil-1992-option.php?site=coil08

[18] MDN Contributors. (2021, May 29). *AudioWorkletProcessor.* https://developer.mozilla.org/en-US/docs/Web/API/AudioWorkletProcessor

[19] MDN Contributors. (2021, May 29). *Channel Splitter Node.* https://developer.mozilla.org/en-US/docs/Web/API/ChannelSplitterNode

[20] MDN Contributors. (2021, April 17). *Fetch API.* https://developer.mozilla.org/en-US/docs/Web/API/Fetch_API

[21] MDN Contributors. (2021, March 27). *WebAssembly.* https://developer.mozilla.org/en-US/docs/WebAssembly

[22] MDN Contributors. (2021, March 9). *Web audio spatialization basics.* https://developer.mozilla.org/en-US/docs/Web/API/Web_Audio_API/Web_audio_spatialization_basics

[23] MDN Contributors. (2021, June 4). *Web Video Text Tracks Format (WebVTT).* https://developer.mozilla.org/en-US/docs/Web/API/WebVTT_API

[24] Merriam-Webster. (n.d.). Jam session. In *Merriam-Webster.com dictionary*. Retrieved June 4, 2021, from https://www.merriam-webster.com/dictionary/jam%20session

[25] Merriam-Webster. (n.d.). Standard. In *Merriam-Webster.com dictionary*. Retrieved June 4, 2021, from https://www.merriam-webster.com/dictionary/standard

[26] Musique concrète. (2021, June 4). In *Wikipedia*. https://en.wikipedia.org/wiki/Musique_concr%C3%A8te

[27] *Ogg documentation.* Retrieved June 4, 2021 from https://xiph.org/ogg/doc/oggstream.html

[28] Richards, J. Wainwright, M. (2018) *Statements on microcomputer music.* Dirty Electronics.

[29] Pantos & May. (September 30, 2011). *HTTP Live Streaming.* https://datatracker.ietf.org/doc/html/draft-pantos-http-live-streaming-07

[30] Papert, S. (1980). *Mindstorms: children, computers, and powerful ideas.* Basic Books, Inc.

[31] *Postgresql*. Retrieved June 4, 2021 from https://www.postgresql.org/

[32] Pfeiffer, S. Parker, C. Pang, A. (March 19, 2005). *The Annodex exchange format for time-continuous bitstreams.* https://web.archive.org/web/20050624103915/http://annodex.org/TR/draft-pfeiffer-annodex-02.html

[33] *Pyo.* Retrieved June 4, 2021 from http://ajaxsoundstudio.com/software/pyo/

[34] Reid, S. (2019, August 16). *No Input Mixer Tutorial* [Video]. Youtube. https://www.youtube.com/watch?v=oUhfkaVUPY8

[35] Rudmin, C. (2021, May 27). *Opus and Wave Recorder*. https://github.com/chris-rudmin/opus-recorder

[36] *Sound Exchange*. Retrieved June 4, 2021 from http://sox.sourceforge.net

[37] *Sqlite.* Retrieved June 4, 2021 from https://www.sqlite.org/index.html

[38] Troy, D. (2018, Aug 1). *How Gaming Is Becoming More Immersive.* https://www.vrfocus.com/2018/08/how-gaming-is-becoming-more-immersive/

[39] Tudor, D. Schonfeld, V. (1972, August). *From Piano to Electronics.* Music and Musicians 20 pp. 24–26.

[40] *Typescript*. Retrieved June 4, 2021 from https://www.typescriptlang.org/

[41] Tzara, T. (December 12, 1920). *Dada Manifesto On Feeble Love And Bitter Love.* https://391.org/manifestos/1920-dada-manifesto-feeble-love-bitter-love-tristan-tzara/

[42] Violante, M.G., Vezzetti, E. & Piazzolla, P. Interactive virtual technologies in engineering education: Why not 360° videos?. *Int J Interact Des Manuf* **13,** 729–742 (2019). https://doi.org/10.1007/s12008-019-00553-y

[43] World Health Organization. (2021, June 4). *WHO Coronavirus (COVID-19) Dashboard*. https://covid19.who.int/