

Trak.gen

A random music generator on the blockchain

Abstract

Trak.gen is a dapp which randomly generates music based on individual pre-composed samples from a variety of sources. It brings together listeners and composers in a co-creative partnership. Listeners can become “finders” of newly generated musical pieces by purchasing them via a non fungible token (NFT), putting their name as “finder” and naming the piece. The samples are removed from the random music generator and become part of a standalone piece which is uploaded permanently to the blockchain, with the name of the “finder” as well as all of the sample composers. A smart contract automatically distributes revenue from streaming to the “finder” as well as the various composers. Trak.gen is thus the first decentralized global musical co-creation app which allows composers as well as listeners/finders to co-create music in an innovative way.

Glossary

Finder

The person who decided to purchase a randomly generated music track and name it, uploading it to a decentralized hosting platform for permanent hosting.

1) Introduction

Randomly generated music is not new. In fact, it goes all the way back to the times of Mozart who allegedly popularized the concept with random music generation for the piano using dices (known as his “musical dice game”). The game worked in the following way: a piano player was given a music sheet with pre-composed musical bars with associated numbers. A person rolled dices several times in a row and the piano player would play the associated bars.

The music was composed in such a way to always yield a coherent musical piece based on musical composition theory, in a similar fashion as randomly recomposing sentences using grammatical rules.

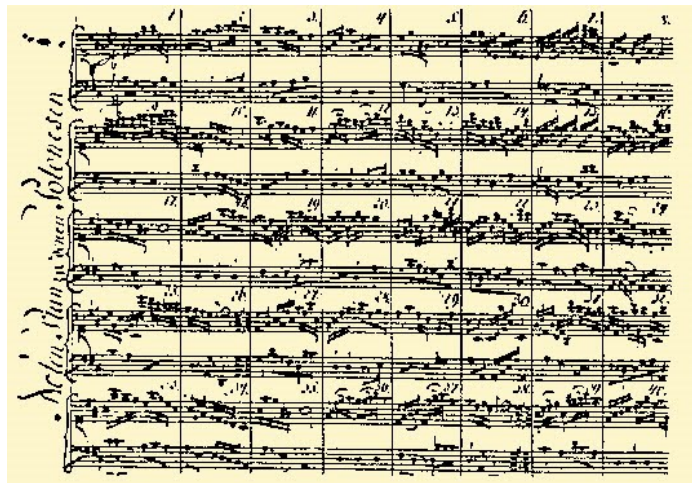
The three sentences below can be recomposed randomly between sentences A, B and C, provided the order (1, 2, 3, 4) remains unchanged¹.

1 2 3 4

- A. The cow ran past the field.
- B. The pig walked through the yard.
- C. The sheep ran into the marsh.

Since then, randomly generated music has advanced so far as to generate music using precomposed samples recomposed randomly², machine learning using samples³, or simply directly composing music as a musical score using machine learning and analyzing a various number of scores⁴.

The current project aims at taking the next step: creating an open platform where composers can upload pre-composed samples in a variety of styles, and building more and more complex and evolving algorithms to randomly generate a soundtrack using these samples.



¹ https://en.wikipedia.org/wiki/Musikalisches_W%C3%BCrfelspiel

² <https://soundraw.io/>

³ <https://openai.com/blog/jukebox/>

⁴ <https://futurism.com/a-new-ai-can-write-music-as-well-as-a-human-composer>

2) Features and architecture

Early prototype

Trak.gen was born out of a Unity random music generation asset using Native Unity scripting (C#), created by [Martin Schmalzried \(Marma\)](#)⁵. The first iteration of the asset was designed for Medieval music, using pre-composed song structures and randomly assigning samples to the various layers of the songs. The asset has since evolved to cover many other musical styles.

The first iteration of the Trak.gen platform will thus use Unity as its native coding language. Porting the code and creating a standalone software solution/music player will be envisaged at a later stage.

Hosting and architecture

The architecture of the software solution will be hosted on a decentralized cloud storage. Sia's Skynet is the current preferred choice.

It will be broken down into seven parts:

- 1) The music generation software, which will have the interface of a standard music player. See the demo of one random music generation asset here: https://siasky.net/AABvTMNYAa0Jab0OfcMxGznySjWgf2m3oDB4rXnRAy_K4Q/
- 2) The random music generation algorithm may also, in the future, benefit from user engagement. For instance, any user could propose a track “structure” which can be added to a pool of structures from which random soundtracks can be generated.
- 3) The “upload” interface for composers’ samples, preferably onto a decentralized cloud service such as Sia, with an interface allowing them to accurately label their sample, selecting the tempo, pitch (tonality), style, and “layer” or sample type (drum loop, melody, accompaniment, bass,...) The composer will be able to “test run” his sample by hearing how it renders in a sandbox environment associated with other pre-composed samples. Once uploaded and validated, the sample will be added to a log file listing all the available samples, sorted according to the assigned labels. This log file will feed into the music generation software to update the available sample list to generate random music. Composers would need to purchase a certain amount of TGEN tokens (Trak.gen) in order to upload their samples to cover for hosting fees and the

⁵ <https://assetstore.unity.com/packages/audio/music/authentic-early-medieval-ages-audio-pack-34367>

change of being “found” by a “finder” (listener), and having their sample featured in a standalone track.

- 4) The composer would have to link his sample to at least 3 or 4 compatible samples such as a drum loop, a bass loop, an accompaniment loop, which combine nicely with his or her sample. In this manner, an algorithm would be able to create a compatibility map where other compatible samples can be inferred from such links. For instance, if a melody sample is compatible with a certain bass loop, and a drum loop is compatible with that same bass loop, the algorithm can assume that the drum loop will be compatible with the melody sample. It is similar to a “web-of-trust” mapping, and will allow for ensuring compatibility between samples which cannot be done strictly by creating labels such as identifying the tonality or tempo.
- 5) An interface allowing listeners to purchase a track that they like, generating a Non Fungible Token (NFT) as proof that they “found” the track, allowing them to name the track and putting their own name as “finder”. The samples would be removed from the random music generation log file. Samples are also removed from the sample “pool” if no one has claimed them via an NFT after a certain number of plays, or skipped the song when playing.
- 6) The track “finder” would be able to modify the randomly generated track to “tweak” it to his/her liking using an intuitive editing interface such as the one featured in the soundraw.io service.
- 7) Once the “finder” is satisfied with the track, it will be rendered in high quality format and uploaded to a separate decentralized cloud storage in a more permanent way, for instance, via allocating enough funds to host the file for a number of years.
- 8) Finally, a separate interface would regroup all of the NFT tracks, and would require paying a small fee for streaming the tracks. A smart contract would automatically distribute the profits to the sample composers as well as the “finder” of the track. Both the NFT and the smart contract would be built on top of the Ethereum blockchain.

Figure A.

