Vizualizacija zvuka u stvarnom vremenu

Džimbeg, Luka

SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

DIPLOMSKI RAD br. 709

VIZUALIZACIJA ZVUKA U STVARNOM VREMENU

Luka Džimbeg

Zagreb, veljača 2025.

SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

DIPLOMSKI RAD br. 709

VIZUALIZACIJA ZVUKA U STVARNOM VREMENU

Luka Džimbeg

Zagreb, veljača 2025.

SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

Zagreb, 30. rujna 2024.

DIPLOMSKI ZADATAK br. 709

Pristupnik:	**Luka Džimbeg (0036522771)**
Studij:	Računarstvo
Profil:	Računarska znanost
Mentor:	prof. dr. sc. Igor Sunday Pandžić

Zadatak: **Vizualizacija zvuka u stvarnom vremenu**

Opis zadatka:

Vizualizacija zvuka uključuje razne načine vizualnog prikaza jedne ili više značajki zvuka ili glazbe. Zvuk se može vizualizirati u različite svrhe, od primjena u medicini do analize signala. U okviru ovog rada usredotočit ćemo se na vizualizaciju u estetske svrhe, pri čemu je cilj stvaranje atraktivnih vizualnih sadržaja povezanih s glazbom. Za to je potrebno analizirati zvuk te generirati vizualni sadržaj, sve u stvarnom vremenu. Vizualni sadržaji mogu se generirati na više načina korištenjem 2D ili 3D grafike, algoritama zasnovanih na fraktalima i raznih drugih. Analiza zvuka može uključivati procjenu ritma kao i razne vrste frekvencijske analize. Vaša je zadaća kroz literaturu proučiti postojeće metode vizualizacije zvuka te odabrati i implementirati jednu ili više metoda.

Rok za predaju rada: 14. veljače 2025.

Sadržaj

1 Uvod **3**

2 Komponente **4**

3 Analizator **5**

3.1 Brza Fourierova transformacija 6

3.2 Podjela u frekvencijske kante 8

3.3 Osvježavanje spektralnog toka 10

3.3.1 Spektralni tok 10

3.4 Detekcija otkucaja 11

3.5 Izlaz analizatora 13

4 Vizualizator **14**

4.1 Ulazni parametri 14

4.2 Simulacija 15

5 Povezivač **17**

5.1 StateChanger 17

5.2 Connector 20

6 Program **22**

6.1 Ulazni parametri 22

6.2 Korisničko sučelje 23

7 Zaključak **25**

7.1 Poboljšanje Analizatora 25

7.2 Poboljšanje Vizualizatora 25

7.3 Poboljšanje Povezivača 25

1 Uvod

Vizualizacija zvuka problem je subjektivne prirode, svaki čovjek drugačije proživljava zvuk, a samim time i vizualizacija istog nije egzaktan i izračunljiv problem. Rad se bavi vizualizacijom zvuka uz pomoć pojavnog (eng. "emergent") algoritma [14], odnosno algoritma čije ponašanje proizlazi iz seta jednostavnih pravila. Zvuk se analizira i zatim setom pravila utječe na parametre jasno definiranog algoritma čije ponašanje tada "vizualizira" zvuk odnosno čije ponašanje prati promjene zvuka.

2 Komponente

Logički pogled na ovaj rad dijeli ga u 3 osnovne komponente:

1. Analizator zvuka
2. Povezivač
3. Vizualizator

Analizator se sastoji od ulaza u obliku `*.mp3` datoteke i izlaza u obliku više parametara dobivenih analizom zvuka. **Povezivač** na ulaz prima izlaz analizatora te primjenjuje filtere i funkcije prijelaza kako bi kreirao izlazne parametre koje tada na ulazu prima **Vizualizator**. **Vizualizator** dobivene parametre koristi za simulaciju pojavnog algoritma, a izlaz algoritma je slika koja se šalje na prikazni zaslon.

Slika 1: Logički pogled na komponente

Prilikom svakog prikaznog okvira dohvaćaju se trenutačni podaci o zvuku koji se pušta, na njima se radi analiza, povezivanje i u konačnici simulacija te prikaz slike na prikaznom zaslonu.

Slika 2: Proces osvježavanja prilikom svakog prikaznog okvira

3 Analizator

Analizator je komponenta koja trenutačni zvučni okvir analizira u više koraka i na kraju proizvodi set vrijednosti koje opisuju stanje zvučne reprodukcije u stvarnom vremenu. Koraci analize zvuka:

1. Brza Fourierova transformacija (FFT)
2. Podjela u frekvencijske kante
3. Osvježavanje spektralnog toka
4. Detekcija otkucaja

Prije same analize cijela zvučna datoteka normalizirana je po formuli 1 kako bi rezultati bili jednostavnije interpretirani, no to nužno znači da glasnoća zvuka ne igra ulogu prilikom simulacije nego samo relativne amplitude i međusobni odnosi pojedinih frekvencijskih komponenti.

$$s'_i = \frac{s_i}{\max(S)} \tag{1}$$

gdje je S zvučna datoteka, s_i vrijednost amplitude uzorka i, a s'_i vrijednost normaliziranog uzorka.

3.1 Brza Fourierova transformacija

Brza Fourierova transformacija (FFT) jest algoritam koji izračunava diskretnu Fourierovu transformaciju (DFT) ili njen inverz. Diskretna Fourierova transformacija je ništa drugo doli Fourierova transformacija na diskretnim točkama neke funkcije koju proučavamo, a sama Fourierova transformacija pretvara signal iz originalne domene, u našem slučaju vremenske domene, u signal u frekvencijskoj domeni i obratno.

FFT koristimo kako bismo dobili raspodjelu signala u frekvencijskim pojasevima. Konkretnije kako bismo razložili ulazni signal na njegove komponente u frekvencijskoj domeni. Koristimo `1024` uzorka zvučnog signala kao ulaz u FFT, što ovisno o stopi uzorkovanja odgovara određenom djeliću sekunde.

Format	**Stopa uzorkovanja**	**Trajanje zvučnog okvira**
MP3	8000 - 16000 Hz	0.128s - 0.064s
	16000 - 32000 Hz	0.064s - 0.032s
	44100 Hz	0.023s
	48000 Hz	0.021s

Tablica 1: Odnos stope uzorkovanja i trajanja zvučnog okvira

Na izlazu dobivamo `1024` uzorka, no zbog simetričnosti same Fourierove transformacije, donji dio spektra, negativne frekvencije, zanemarujemo i time imamo efektivno `512` uzoraka u frekvencijskoj domeni. Svaki od uzoraka predstavljen je kompleksnim brojem, gdje njegov modul označava relativnu amplitudu frekvencija unutar tog određenog pojasa, a njegova faza označava fazni pomak vala frekvencije koju frekvencijski pojas predstavlja. Veličina frekvencijskog pojasa odnosno frekvencijska rezolucija FFTa ovisi o broju uzoraka i stopi uzorkovanja po formuli:

$$\Delta f = \frac{F_s}{N}$$

gdje je Δf frekvencijska rezolucija, F_s je stopa uzorkovanja, a N je broj uzoraka. Maksimalna frekvencija koju možemo prikazati, Nyquistova frekvencija, može se izračunati pomoću:

$$f_{\max} = \frac{F_s}{2}$$

I konačno centralna frekvencija svakog frekvencijskog pojasa `k (za k = 0,1,2,...,N-1)` dana je sa:

$$f_k = \frac{k \cdot F_s}{N}, \quad k = 0, 1, 2, \ldots, N-1$$

Format	**Stopa uzorkovanja**	**Frekvencijska rezolucija FFTa**
MP3	8000 - 16000 Hz	7.8Hz - 15.625Hz
	16000 - 32000 Hz	15.625Hz - 31.25Hz
	44100 Hz	43.07Hz
	48000 Hz	46.875Hz

Tablica 2: Odnos stope uzorkovanja i veličine frekvencijskog pojasa za FFT od 1024 uzorka

Slika 3: Rezultat izlaza u nasumičnom trenutku na korisničkom sučelju programa.

3.2 Podjela u frekvencijske kante

Podjela u frekvencijske kante označava proces podjele izlaza FFT faze u kante većeg frekvencijskog raspona kako bismo imali manje parametara i jače signale za upravljanje simulacijom. Podjela prati okvirno osnovne frekvencijske podskupove, vidljive u tablici 3, korištene u dizajniranju zvučnih sustava [10].

Frekvencijski podskup	**Frekvencijski raspon**
Sub-bass	16 Hz - 60 Hz
Bass	60 Hz - 250 Hz
Lower midrange	250 Hz - 500 Hz
Midrange	500 Hz - 2 kHz
Higher midrange	2 kHz - 4 kHz
Presence	4 kHz - 6 kHz
Brilliance	6 kHz - 20 kHz

Tablica 3: Frekvencijski podskupovi zvuka

Stvarna podjela sastoji se od `9` skupina, koje okvirno prate gore navedenu raspodjelu. Važno je napomenuti da je rad programa isključivo prilagođen za stopu uzorkovanja od `44100 Hz`. Podjela kanti navedena je sljedećom tablicom.

Grupa	**Frekvencijski raspoon (Hz)**	**Broj izlaza FFTa unutar grupe**
0	0 to 86.14	2
1	86.14 to 172.28	2
2	172.28 to 344.56	4
3	344.56 to 689.12	8
4	689.12 to 1378.24	16
5	1378.24 to 2756.48	32
6	2756.48 to 5512.96	64
7	5512.96 to 11025.92	128
8	11025.92 to 22051.84	256

Tablica 4: Podjela na frekvencijske kante

Podjela se može opisati i pomoću formule 2. Valja napomenuti da ovdje koristimo prvu polovicu izlaza FFTa zbog činjenice da procesiranjem prirodnih signala dobivamo zrcaljeni spektar po vrijednostima između pozitivnih i negativnih frekvencija. S obzirom na to da negativna strana ne nosi nikakvu novu informaciju ovdje efektivno prelazimo u jednostrani spektar [7].

$$\text{Grupa}(i) = \frac{2}{\text{N}} \cdot \sum_{k=K(i-1)}^{K(i)-1} \text{FFT\_out}(k),$$

$$\text{gdje } K(i) = \begin{cases} 0, & \text{ako } i < 0, \\ 1 + \sum_{j=0}^{i} 2^j, & \text{za } i >= 0, \end{cases} \tag{2}$$

$$\text{i} \quad \text{N} = 1024$$

Podjela s pomoću potencija broja dva izabrana je zbog činjenice da ljudski sluh lakše raspoznaje razlike u nižim frekvencijama nego u višim frekvencijama [8], stoga eksponencijalno povećavanje razlika kanti prilikom većih frekvencija "grupira" izlaze FFT faze koje ljudsko uho ne raspoznaje relativno jednostavno čime dobivamo grupe koje su lako prepoznatljive ljudskom uhu, a samim time čine dobre parametre za modifikaciju slike. Skaliranje faktorom $\frac{2}{N}$ radimo kako bismo dobili amplitude početne zvučne datoteke, valja napomenuti kako ćemo zapravo dobiti normalizirane amplitude jer smo amplitude početne zvučne datoteke normalizirali prije same analize.

3.3 Osvježavanje spektralnog toka

Faza osvježavanja spektralnog toka uključuje izračunavanje relativne snage pojedinih frekvencijskih komponenti. Izračun relativne snage vrši se po formuli 3, gdje skaliranje faktorom `2` radimo kako bismo uračunali činjenicu da je snaga podijeljena između pozitivnih i negativnih frekvencija, a vrijednosti su zrcaljene i k tome radimo samo na pozitivnom dijelu spektra [7]. A skaliranje faktorom $\frac{1}{N}$ vršimo jer se u pozadini samog izračuna FFT-a odnosno DFT-a radi sumacija po svim vrijednostima, čime je sve skalirano za faktor `N` odnosno broj uzoraka FFTa.

$$P(k) = 2 \cdot \left(\frac{|\text{FFT\_out(k)}|}{N} \right)^2 \quad \text{za} \quad k = 0, 1, 2, \ldots, \frac{N}{2} - 1 \tag{3}$$

Zatim ponovno radimo podjelu u frekvencijske kante, samo ovaj put umjesto sumacije amplituda radimo sumaciju snaga. Time dobivamo još jedan set od 9 grupa kao i u odjeljku 3.2, njih ćemo nazivati `Grupa_P(i)`.

3.3.1 Spektralni tok

Spektralni tok je mjera brzine promjene u snazi signala [16]. Računa se često kao L2-norma između dva normalizirana spektra snage konkretnije između trenutačnog okvira i prošlog okvira [4].

$$Fl_{(i,i-1)} = \sum_{k=1} \left(Grupa\_Pn_i(k) - Grupa\_Pn_{i-1}(k) \right)^2, \tag{4}$$

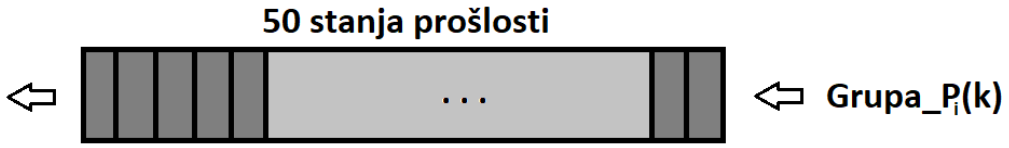
$$Grupa\_Pn_i(k) = \frac{Grupa\_P_i(k)}{\sum_{l=1} Grupa\_P_i(l)}, \tag{5}$$

Gdje `i` označava trenutni prikazni/zvučni okvir.

3.4 Detekcija otkucaja

Otkucaj ili takt (*engl. beat*) u glazbi predstavlja percipirani ritmički događaj koji služi kao referentna točka za vremenski slijed i tok glazbe. Često je povezan s primjetnim promjenama energije, poput glasnijih zvukova, naglašenih nota ili ritmičkih obrazaca koji se ističu. Iako se otkucaji često smatraju ravnomjerno raspoređenima u jednostavnijim kontekstima, njihov stvarni razmak može varirati zbog izražajnosti izvedbe ili ljudske interpretacije.

Detekcija otkucaja problem je teško rješiv u računalnim sustavima, suprotno tome, detekcija otkucaja ljudima i životinjama je urođena stvar, no postoje algoritmi koji ga mogu dovoljno dobro aproksimirati. Ljudsko uho zapravo raspoznaje razlike u energiji, odnosno jaka povećanja energije u određenom spektru detektiramo kao otkucaj. Stoga algoritam koji koristimo [12] uzima tu činjenicu i na temelju toga gradi logiku same detekcije. Ovdje se koristi blago modificirana verzija algoritma kako bi više odgovarao potrebama programa.



Slika 4: Analizator otkucaja.

Otkucaj detektiramo za svaku frekvencijsku kantu pojedinačno te jednom na sumi svih kanti što nazivamo `Tempo`. Detekcija se sastoji od čuvanja povijesti određene duljine, konkretno u našem slučaju `50`, što uz stopu uzorkovanja od `44100Hz` te pomak i veličinu okvira od `1024` uzorka otprilike odgovara `1.1s`, te izračuna srednje vrijednosti povijesti i njene varijance.

$$\text{Srednja vrijednost povijesti } H: \quad \langle H \rangle = \frac{1}{50} \sum_{i=1}^{50} H[i] \tag{6}$$

$$\text{Varijanca povijesti } H: \quad V(H) = \frac{1}{50} \sum_{i=1}^{50} \left(H[i] - \langle H \rangle\right)^2 \tag{7}$$

Nakon izračuna srednje vrijednosti i varijance, ubacujemo snagu trenutačnog zvučnog okvira u povijest te računamo koeficijent otkucaja i postojanje samog otkucaja. For-

mule 8 i 9 su dobivene manualno testiranjem različitih zvučnih datoteka.

$$\text{Koeficijent otkucaja:} \quad C_{\text{otkucaj}} = (max)(-0.005 \cdot V(H) + 1.5) + 1.1 \tag{8}$$

$$\text{otkucaj} = (Grupa\_P_i(k) > C_{\text{otkucaj}} \cdot \langle H \rangle) \tag{9}$$

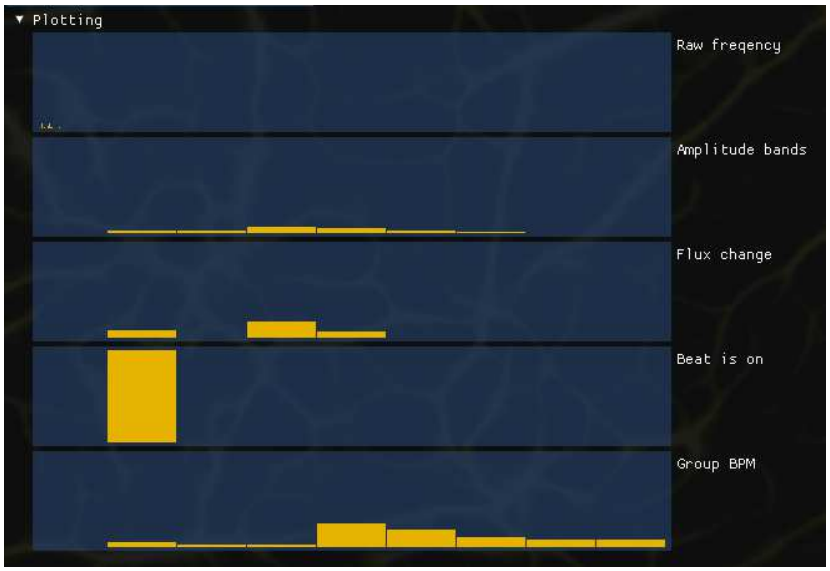
Za izračun BPM-a (otkucaja po minuti, *engl. beats per minute*) svake komponente i globalnog tempa (BPM kroz sve komponente), koji nazivamo tempo u sklopu ovog projekta, uz već navedene izračune, pamtimo i vremenske oznake za svaki unos u povijesti te broj otkucaja koji su se pojavili u samoj povijesti. Pomoću te informacije izračunavamo "trenutačni" broj otkucaja po minuti (*engl. BPM*) što koristimo kao mjeru tempa muzike.

3.5 Izlaz analizatora

Izlaz analizatora sastoji se od rezultata svih dosad izračunatih faza. Konkretno to su

1. Grupa(k) - 9 vrijednosti relativnih amplituda,
2. Fl(k) - 9 vrijednosti spektralnog toka,
3. Beat(k) - 9 binarnih vrijednosti prisutnosti otkucaja
4. BPM(k) - 9 vrijednosti BPMa svake komponente u trenutku
5. Tempo - tempo u trenutku

gdje `k` označava frekvencijsku kantu.



Slika 5: Izlazi analizatora

4 Vizualizator

Komponenta vizualizatora zapravo samo obavlja osvježavanje parametara simulacije te korak simulacije. Za simulaciju koristimo računske sjenčare kako bismo efikasno i raspodijeljeno izvrtjeli korak naše simulacije. Algoritam koji smo odabrali za simulaciju jedan je od izranjajućih algoritama (engl. emergent algorithms [14]), simulira plijesan Physarum polycephalum [5] setom jednostavnih pravila primijenjenih na više agenata, gdje je agent samo jedna nit izvođenja, čime dolazi do izranjanja kompleksnog ponašanja za simulaciju s puno agenata. Konkretan broj agenata našeg programa jest `2^20` odnosno `1048576`. Svaki od agenata uz sebe ima vezana dva atributa:

1. `position` - Pozicija agenta na teksturi
2. `angle` - Smjer kretanja agenta

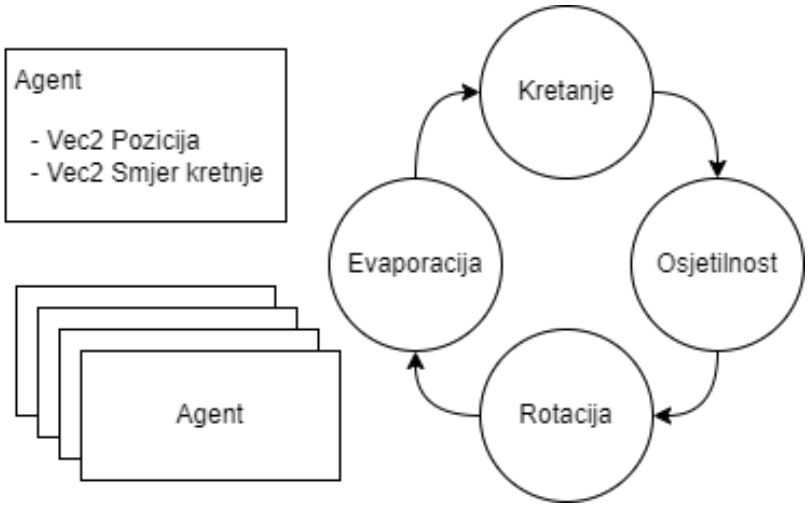
4.1 Ulazni parametri

Ulazni parametri algoritma su predefinirani i koriste se za kontroliranje simulacije odnosno uzoraka koji se pojavljuju na ekranu. Parametri su sljedeći:

1. `movementSpeed (ms)` - brzina kretanja svakog agenta
2. `trailEvaporationSpeed (tao)` - brzina isparavanja "tragova"
3. `sensorAngleOffset (sao)` - odmak od smjera kretanja u kojemu agent reagira na tragove drugih agenata
4. `sensorDistance (sd)` - odmak od pozicije u smjeru sakupljanja signala odakle agent sakuplja vrijednosti tragova
5. `turnSpeed (ts)` - brzina rotacije agenta
6. `color (c)` - boja odnosno vrijednost traga koju agent postavlja na trenutačnoj poziciji
7. `colorCoeff (cf)` - brzina "isparavanja" svake komponente boje koju agent postavlja
8. `colorCap (cc)` - maksimalne vrijednosti traga po komponentama
9. `deltaTime (dt)` - vrijeme proteklo od zadnjeg prikaznog okvira

S tim parametrima, izuzev `deltaTime`, opisano je stanje simulacije. Dakle stanje simulacije `S` definiramo kao osam dimenzionalni prostor parametara simulacije.

4.2 Simulacija



Slika 6: Agenti i faze izvođenja

Simulacija agenta sastoji se od više koraka:

1. **Faza kretnje** - Tijekom faze kretnje agent se kreće u smjeru definiranom unutar samog agenta za broj piksela definiranim izrazom `movementSpeed * deltaTime`, te osvježava svoju poziciju i postavlja "trag" odnosno boja teksturu s vrijednosti `color`.

2. **Osjetilna faza** - Agent sakuplja vrijednosti "tragova" odnosno vrijednosti teksture na pozicijama koje se nalaze u smjerovima `angle + sensorAngleOffset`, `angle - sensorAngleOffset` i `angle` na udaljenosti `position + sensorDistance`. S tih pozicija senzora sumiraju se vrijednosti teksture u matrici `3x3` gdje je središnje polje pozicija senzora za taj smjer prikupljanja tragova.

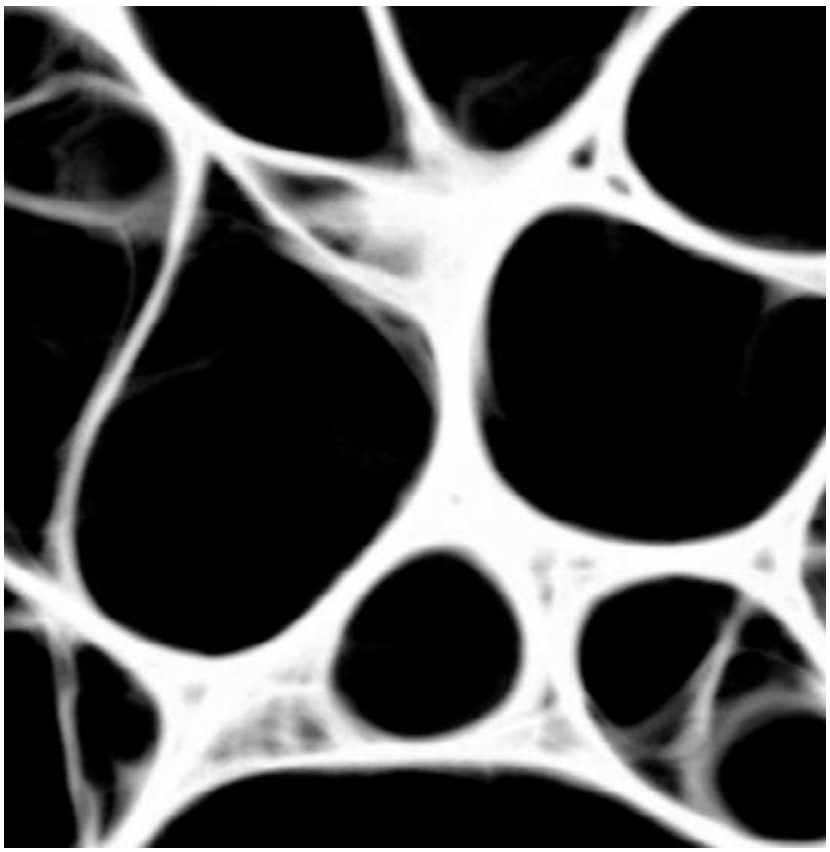
3. **Rotacija** - Ovisno o prikupljenim tragovima odvija se rotacija po zadanim pravilima:

```
Ako je najjači signal naprijed
    nastavi u istom smjeru;
Ako je najslabiji signal naprijed
```

```
        okreni se u nasumičnom smjeru za kut turnSpeed * deltaTime
    Inače ako je jači signal lijevo
        okreni se lijevo za kut turnSpeed * deltaTime;
    Inače ako je jači signal desno
        okreni se desno za kut turnSpeed * deltaTime;
```

4. **Evaporacija** - Svaki trag, svakog agenta umanjuje se za vrijednost`colorCoeff * deltaTime * trailEvaporationSpeed`

Svaki korak simulacije se izvršava za svakog agenta na ekranu. I na kraju izvođenja svih faza za svakog agenta radi se zamućivanje slike radi estetski ugodnijih rezultata. Konkretno radi se Gaussov Blur `3x3` [15].

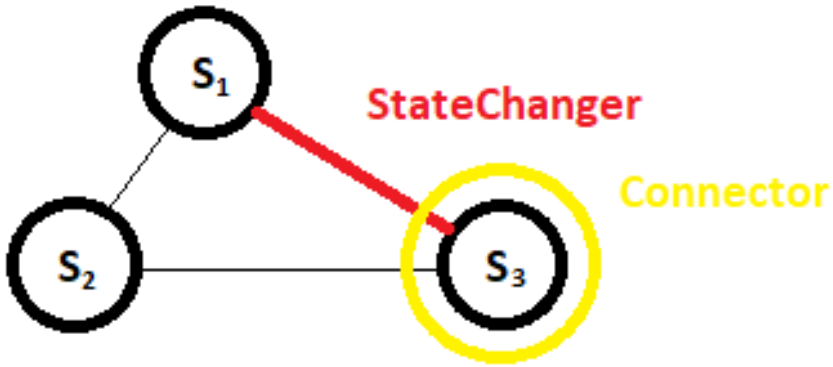


Slika 7: Primjer rezultata simulacije

5 Povezivač

Povezivač je komponenta koja spaja izlaze iz **Analizatora** 3 te ulaze u **Vizualizator** 4. Kako je algoritam kaotične naravi te visoko dimenzionalnog prostora, ova komponenta koristi heuristiku kako bi povećala stabilnost simulacije. Konkretno ova komponenta sadrži dvije potkomponente:

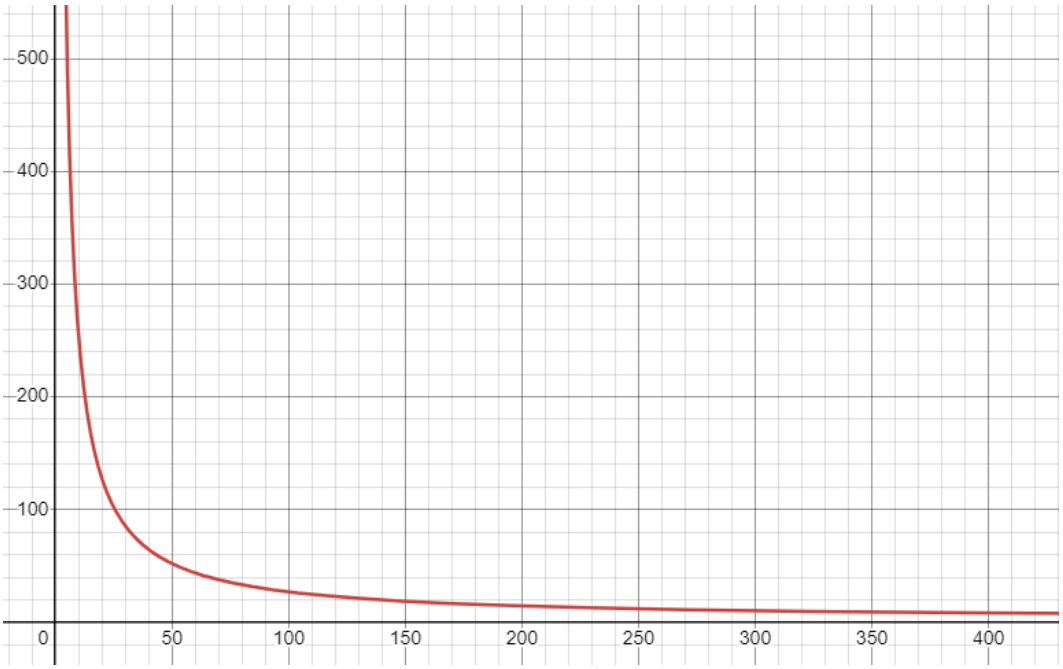
1. StateChanger - potkomponenta odgovorna za globalnu promjenu stanja
2. Connector - potkomponenta odgovorna za lokalnu promjenu stanja, točnije oscilaciju parametara u ovisnosti izlaza iz **Analizatora**



Slika 8: Odnos globalne i lokalne promjene stanja

5.1 StateChanger

Ova komponenta odgovorna je za prijelaze između stabilnih stanja. Stabilna stanja simulacije su stanja koja su izabrana ručnim testiranjem raznih parametara te su subjektivno ocijenjena kao vizualno privlačna i stabilna. Primjeri nekih stabilnih stanja dani su slikama 10.



Slika 9: Funkcija izračuna trajanja prijelaza

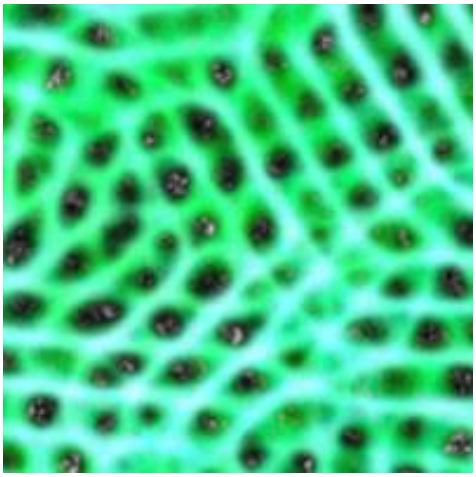
Pri početku programa izabere se nasumična permutacija stabilnih stanja, te se radi interpolacija ovisno o tempu zvučne datoteke. Koristimo funkciju 10 kako bismo dobili koliko traje prijelaz za pojedini tempo. Graf funkcije za izračun trajanja prijelaza prikazan je slikom 9. Dodavanje male konstante u djelitelju je radi jednostavnosti rada programa.

$$\text{trajanje\_prijalaza} : f(\text{tempo}) = \frac{(3000 - \text{tempo})}{\text{tempo} + 0.0001} + 5 \tag{10}$$

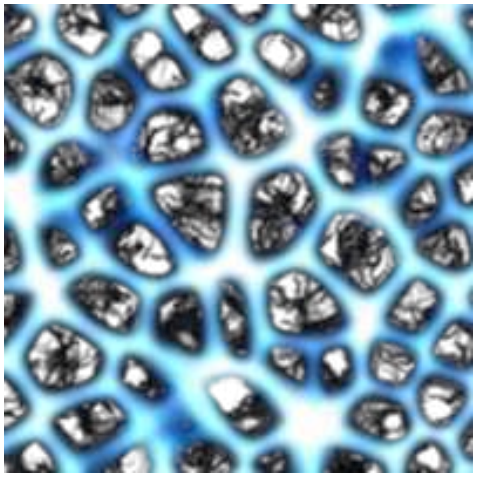
Moglo bi se reći da je ova komponenta odgovorna za globalnu promjenu stanja, jer uvjet unutar ove komponente utječe na promjenu svih parametara simulacije te to odrađuje direktno. Za razliku od toga **Connector** je odgovoran za lokalnu promjenu stanja, odnosno oscilaciju parametara oko trenutnog stabilnog stanja.

Prijelaz između stanja odnosno postavljanje parametara vizualizatora na pojedino stanja dešava se prilikom ispunjenja uvjeta da je trajanje prijelaza za prosječni tempo od zadnjeg prijelaza veće nego vrijeme proteklo od zadnjeg prijelaza.

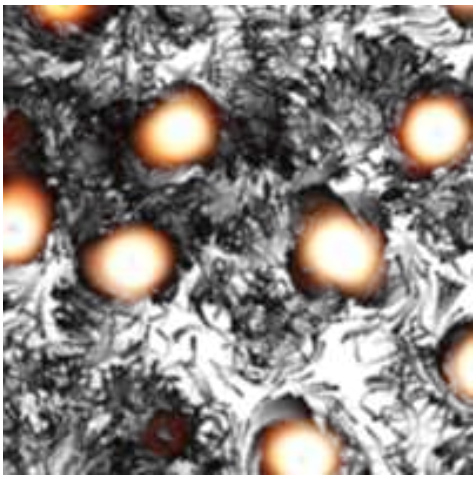
$$f\left(\text{avg}_{T_{i-1} \to T_i}(\text{tempo})\right) > T_i - T_{i-1} \tag{11}$$



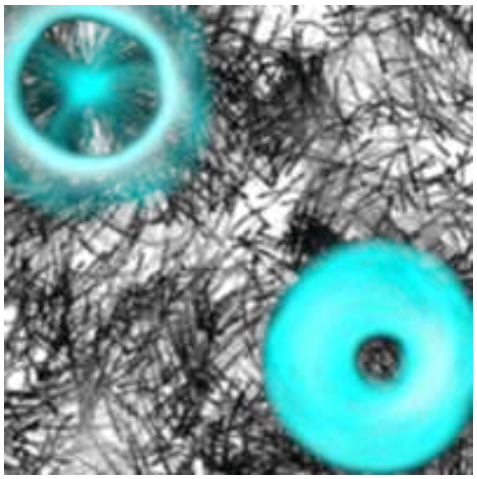
(a) Stanje 1



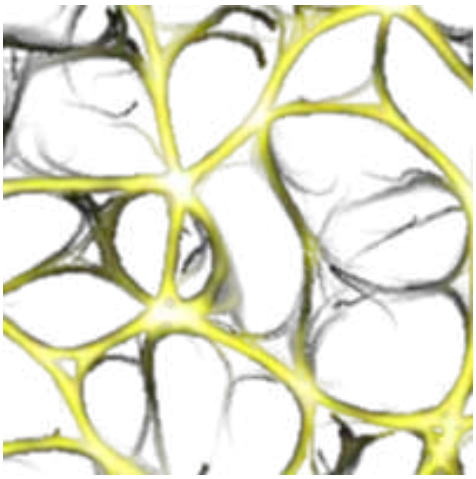
(b) Stanje 2



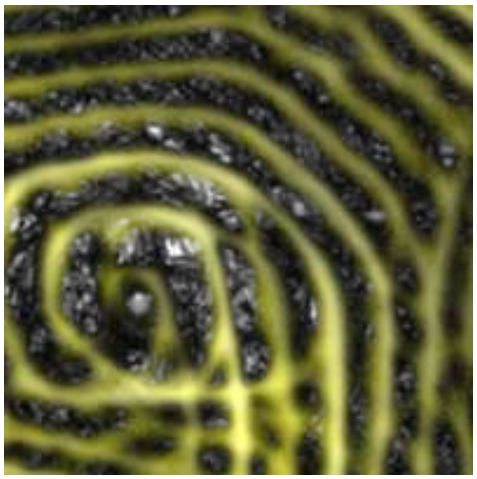
(c) Stanje 3



(d) Stanje 4



(e) Stanje 5



(f) Stanje 6

Slika 10: Primjeri stabilnih stanja

5.2 Connector

Ova komponenta odgovorna je, kako je rečeno, za lokalnu promjenu stanja, odnosno promjenu stanja u stanje unutar osam dimenzionalne sfere, kako smo već spomenuli svako stanje opisano je s `8` parametara, u čijem je središtu trenutačno stanje.

Logika ove komponente ne prati nikakva pravila nego je evoluirala tijekom niza manualnih testiranja, stoga su samo neki od odgovora na promjenu intuitivne prirode dok neki samo rezultiraju lijepim vizualima bez jasnih intuitivnih interpretacija zašto je to tako.

Na ulazu ove komponente nalazi ze izlaz iz **Analizatora**, a pomoću njih se rade promjene trenutačnog stanja `S` koje se zatim šalju na ulaz **Vizualizatora** koji se sastoji, kako smo već napomenuli, od 8 parametara.

Definiramo funkciju promjene `chg(var, base, diff, signal, strength, scale)`

$$\underbrace{\text{var}}_{\substack{\text{varijabla koja}\\ \text{se šalje Vizualizatoru}}} = \underbrace{\text{base}}_{\substack{\text{vrijednost varijable}\\ \text{trenutnog stanja}}} + \underbrace{\text{diff}}_{\substack{\text{pomak od}\\ \text{preddefinirane vrijednosti}}} \cdot \underbrace{(\text{signal} \cdot (\text{strength} + 0.5))}_{\substack{\text{skaliranje snagom}\\ \text{ukoliko je signal prisutan}}}$$

$$\text{var} = \text{var} \cdot \underbrace{(1 + \text{signal} \cdot (\text{diff} \geq 0 \,?\, \text{scale} : -\text{scale}))}_{\substack{\text{finalno skaliranje postotkom}\\ \text{uz prisutnost signala}}}$$

Signali korišteni kao indikator aktivacije odstupanja od trenutačnog stanja definirani su sljedećom tablicom, gdje `Formula` označava izraz kojim se signal aktivira, `Trajanje` označava koliko dugo signal ostaje aktivan bez obzira na formulu, `Pauza` označava koliko dugo signal ostaje inertan na aktivacijski signal formule nakon završetka trajanja. Vizualna interpretacija definicije signala vidljiva je slikom 11

Signal	**Formula**	**Trajanje (s)**	**Pauza (s)**
bass	$Beat(0)\|\|Beat(1)\|\|Beat(2)$	0.1	0.1
bassLong	$Beat(0)\|\|Beat(1)\|\|Beat(2)$	0.5	0
lmid	$Beat(3)$	0.25	0
mid	$Beat(4)\|\|Beat(5)$	0.5	0
hmid	$Beat(6)$	1	1
prebri	$Beat(7)\|\|Beat(8)$	0.1	0
splong	$BPM(3) + BPM(4) > \frac{BPM(0)+BPM(1)+BPM(2)}{3}$	1.5	2

Tablica 5: Signali

Promjene su dane tablicom 6, `trailEvaporationSpeed` i `colorCap` su fiksirani i ne

mijenjaju se tijekom izvedbe programa s obzirom na to da se tijekom testiranja pokazalo da njihove promjene uvelike utječu na kvalitetu vizualizacije, pa bi u ekstremnim situacijama dolazilo do potpunog degradiranja ljepote vizualizacije. `sensorDistance` je jedina varijabla koji ne koristi istu funkciju generiranja promjena.

Var	**Base**	**Diff**	**Strength**	**Scale**	**Signal**
movementSpeed	S^{ms}	$S_i^{ms} - S_{i-1}^{ms}$	$Fl(3) + Fl(4) + Fl(5)$	0.25	lmid | mid
trailEvaporationSpeed	-	-	-	-	-
sensorAngleOffset	S^{sao}	$S_i^{sao} - S_{i-1}^{sao}$	$Fl(6)$	0.0125	hmid
sensorDistance	-	-	-	-	-
turnSpeed	S^{ts}	$S_i^{ts} - S_{i-1}^{ts}$	$Fl(4) + Fl(5)$	0.1	mid
color	S^{c}	$S_i^{c} - S_{i-1}^{c}$	$Fl(0) + Fl(1) + Fl(2)$	0.75	bass
colorCoeff	S^{cf}	$S_i^{cf} - S_{i-1}^{cf}$	$\max(\text{color}/S^c - 1.5, 0)$	0	bassLong
colorCap	-	-	-	-	-

Tablica 6: Tablica promjena funkcijom chg(...)

Formula promjene `sensorDistance` varijable dana je s izrazom ispod.

$$\text{sensorDistance} = S_i^{sd} + \text{sign}(S_i^{sd}) \cdot (100 \cdot splong + 30 \cdot bass \cdot (Fl(0) + Fl(1) + Fl(2)))$$

Slika 11: Vizualna interpretacija signala

6 Program

Prilikom izgradnje programa korištene su mnoge već izrađene biblioteke kako bi se olakšala implementacija cjelokupnog programa. Ukratko ćemo navesti svaku od njih te njihove svrhe i nećemo ulaziti više u detalje.

1. **GLFW**: Više platformska biblioteka za razvoj unutar OpenGL, OpenGL ES i Vulkan okruženja. [11].
2. **Dear ImGui**: Jednostavno grafičko sučelje za C++. [1].
3. **JSON for Modern C++**: Biblioteka za manipulaciju `.json` datotekama. [6].
4. **LODEPNG**: Jednostavna biblioteka za čitanje i pisanje `.png` datoteka. [13].
5. **FFmpeg**: Manipulacija i reprodukcija video i audio zapisa. [2].
6. **FFTW**: Implementacija FFT algoritma. [3].
7. **CLI11**: Biblioteka za jednostavnu implementaciju komandnog korisničkog sučelja. [9].

6.1 Ulazni parametri

Ulazni parametri programa koriste se za upravljanje programom kako bismo ga pokrenuli s određenom zvučnom datotekom, odnosno kako bi analizator bio uključen što odgovara zastavici `f` nakon koje slijedi put do željene datoteke. Tu je i zastavica `c` koja kontrolira paljenje komponente povezivača, konkretnije želimo li da simulacija odgovara na podražaje iz analizatora ili ne. Cijeli ispis parametara nalazi se u isječku koda ispod.

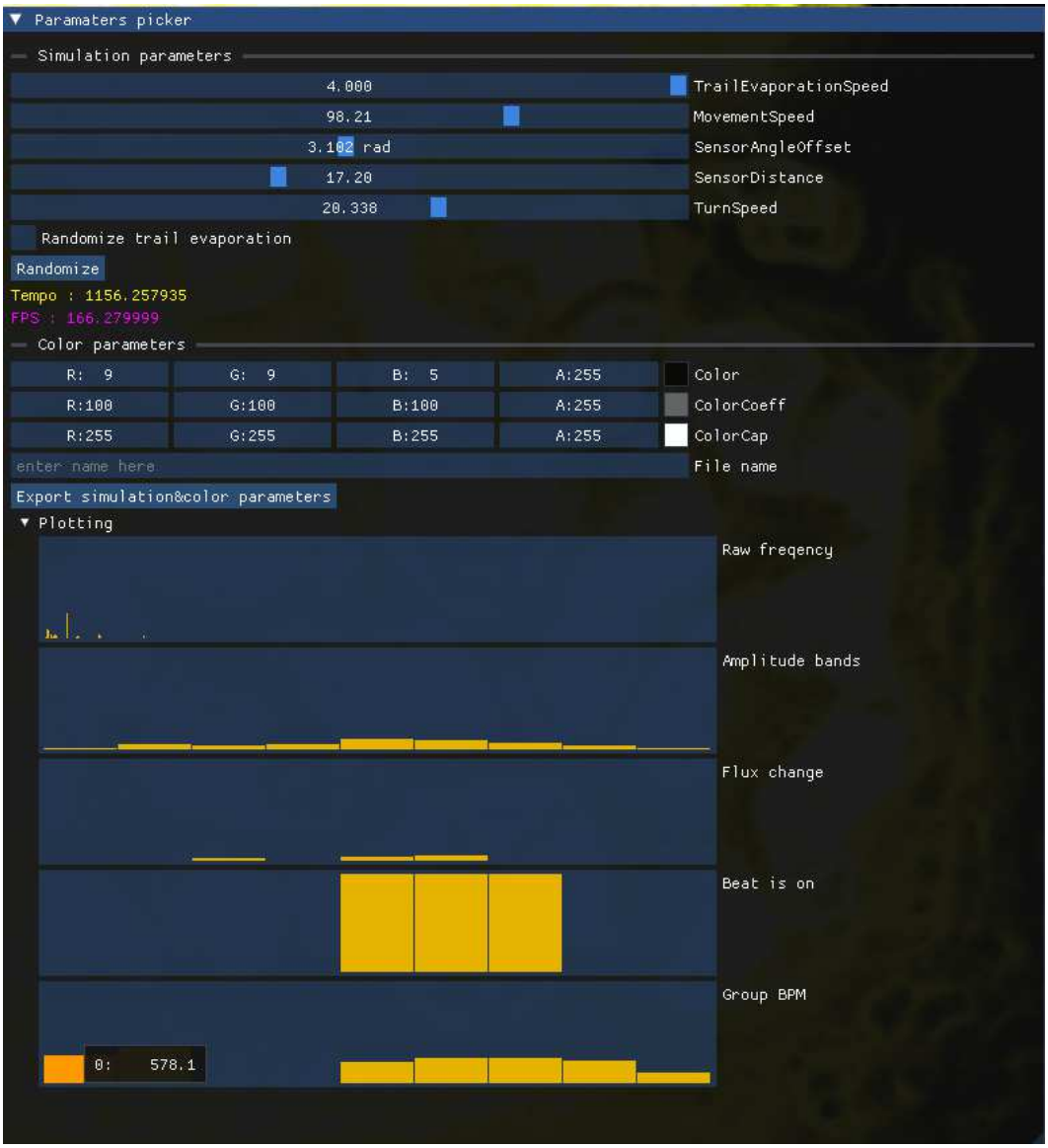
```
App description
Usage: ./audiovizz.exe [OPTIONS]

Options:
  -h,--help                   Print this help message and exit
  -f,--file TEXT              Audio to play
  -c,--use_config [0]         Use config file
```

6.2 Korisničko sučelje

Korisničko sučelje programa šturo je jer i osnovni naum nije korištenje istoga, barem ne u trenutačnom stanju programa. Korisničko sučelje prikazano je slikom 12. Sučelje nudi pogled na sve parametre simulacije kao i mogućnost njihove promjene te pogled na sve izlaze iz analizatora, ako je isti uključen. Od dodatnih informacija i mogućnosti tu je broj prikaznih okvira u sekundi, mogućnost nasumičnog odabira parametara simulacije te izvoz istih i spremanje na disk. Mogućnost nasumičnog odabira parametara simulacije ekstenzivno je korištena u građenju programa i pronalaženju stabilnih stanja.

Kako se prilikom pokretanja programa koristi i konfiguracijska datoteka u obliku liste datoteka stanja, korisnik može tražiti stabilna stanja, spremiti ih na disk i modificirati konfiguracijsku datoteku kako bi ona bila korištena prilikom sljedeće simulacije. Primjer konfiguracijske datoteke prikazan je isječkom koda dolje.

```
{
    "states": [
        "detailed\\blue_comb",
        "detailed\\blue_small_net",
        "detailed\\cyan_circles",
        ...
        "detailed\\yellow_net",
        "detailed\\yellow_spin"
    ]
}
```



Slika 12: Korisničko sučelje

7 Zaključak

Program ispunjava svoju svrhu stvaranja vizualno privlačnih slika koje odgovaraju zvuku i radi to relativno dobro. Program je zbog trenutačnog stanja izvedbe unutar Povezivača prilagođen za rad s glazbom gdje su područja "bass" i "sub-bass" najizraženija i jer je ta područja lako detektirati i razlikovati s obzirom na mali frekvencijski raspon. Prostora za poboljšanje je mnogo i to ćemo navesti nadalje.

7.1 Poboljšanje Analizatora

Analizator iako robustan, ne generira jednako primamljive vizuale za sve tipove glazbe. Ponajviše reagira na *techno* muziku ili muziku gdje je područje basa najizraženije. Odvajanje ili analiziranje ulaznih datoteka po tipu muzike kojoj pripadaju te modifikacija parametara, koeficijenata detekcije otkucaja i slično, ovisno o tom tipu rezultiralo bi rješenjem koje bolje odgovara za svaki pojedini tip muzike.

Također testiranje s drugačijim brojem uzoraka algoritma FFT-a jedno je od potencijalnih poboljšanja, time bismo dobili bolju rezoluciju u frekvencijskoj analizi čime bismo mogli i bolje razlikovati prisutnost određenih instrumenata i zvukova.

Zasebna detekcija ljudskog glasa također je područje vrijedno istraživanja, kako bi algoritam drugačije odgovarao na muške i ženske vokale ili pak prisutnost višeglasja, ali i općenito na prisustvo ili odsustvo vokala.

7.2 Poboljšanje Vizualizatora

Vizualizator iako građen po jasno definiranom algoritmu, ipak ima neke modifikacije specifične za ovaj rad. Dodatna testiranja i eventualne promjene koje bi proizašle iz toga imaju mogućnost ubrzati rad simulacije ili pak popraviti njen sveukupni vizualni dojam.

7.3 Poboljšanje Povezivača

Najviše poboljšanja ipak postoji u ovoj komponenti koja je građena ručno testiranjem. Jedna od ideja vrijedna razmatranja je treniranje umjetne inteligencije za detekciju ovisnosti između izlaza analizatora i ulaza u simulaciju koje bi rezultirale privlačnim vizualima. Iako je ta ideja donekle bila istražena, nedostajalo je resursa i znanja za

provedbu iste. Vrijedno bi također bilo građenje kompleksnije mreže odluka ili možda čak mreže odluka specifične za pojedini tip muzike ili stanje, no to bi iziskivalo suviše manualnog rada za opseg ovog rada.

Literatura

[1] Omar Cornut and Contributors. *Dear ImGui: Bloat-free Graphical User Interface for C++*. https://github.com/ocornut/imgui. 2025.

[2] FFmpeg Developers. *FFmpeg: A Complete, Cross-platform Solution to Record, Convert and Stream Audio and Video*. https://ffmpeg.org. 2025.

[3] Matteo Frigo and Steven G. Johnson. *FFTW: Fastest Fourier Transform in the West*. http://www.fftw.org. 2025.

[4] Theodoros Giannakopoulos and Aggelos Pikrakis. *Introduction to Audio Analysis: A MATLAB® Approach*. Waltham, MA: Academic Press, 2014. ISBN: 978-0-08-099388-1. URL: https://www.sciencedirect.com/book/9780080993881/introduction-to-audio-analysis.

[5] Jeff Jones. "Characteristics of pattern formation and evolution in approximations of Physarum transport networks". In: *Artificial Life* (2010). DOI: 10.1162/artl.2010.16.2.16202. URL: https://pubmed.ncbi.nlm.nih.gov/20067403/.

[6] Niels Lohmann. *JSON for Modern C++*. https://github.com/nlohmann/json. Version 3.x. 2025.

[7] Audrey F. Harvey Michael Cerna. "The Fundamentals of FFT-Based Signal Analysis and Measurement". In: *National Instruments Corporation* (2000). Pristupljeno Siječanj 4, 2025. URL: https://www.sjsu.edu/people/burford.furman/docs/me120/FFT_tutorial_NI.pdf.

[8] Jemma Frost Nicholas LaPlant Alex Konzmann. "Frequency Discrimination and Inter-Stimulus Interval". In: *Journal of the Society of Auditory Measurement (JOSAM)* DOI: 10.37714/josam.vi0.98 (2022). Pristupljeno Siječanj 4, 2025. URL: https://www.josam.org/josam/article/view/98?download=pdf.

[9] Henry Schreiner. *CLI11: Command Line Interface for Modern C++*. https://github.com/CLIUtils/CLI11. 2025.

[10] Jeff Smoot. *A Look at Audio Frequency Range and Audio Components*. Pristupljeno Siječanj 4, 2025. 2025. URL: https://www.digikey.com/en/articles/a-look-at-audio-frequency-range-and-audio-components.

[11] GLFW Development Team. *GLFW: A Multi-platform Library for OpenGL, OpenGL ES, and Vulkan Development*. https://www.glfw.org. Version 3.x. 2025.

[12] George Tzanetakis. *Beat Detection Algorithms*. Pristupljeno Siječanj 6, 2025. 2003. URL: https://archive.gamedev.net/archive/reference/programming/features/beatdetection/index.html.

[13] Lode Vandevenne. *LODEPNG: Portable PNG Decoder and Encoder in C++*. https://github.com/lvandeve/lodepng. 2025.

[14] Wikipedia contributors. *Emergent Algorithm — Wikipedia, The Free Encyclopedia*. https://en.wikipedia.org/wiki/Emergent_algorithm. [Pristupljeno Siječanj 10, 2025]. 2025.

[15] Wikipedia contributors. *Gaussian Blur — Wikipedia, The Free Encyclopedia*. Pristupljeno 7-Siječanj-2025. 2025. URL: https://en.wikipedia.org/wiki/Gaussian_blur.

[16] Wikipedia contributors. *Spectral Flux*. https://en.wikipedia.org/wiki/Spectral_flux. Pristupljeno Siječanj 5, 2025. 2023.

Vizualizacija zvuka u stvarnom vremenu

Sažetak

Ovaj rad opisuje implementaciju programa za vizualizaciju zvuka u stvarnom vremenu pomoću pojavnog algoritma Physarum. Rad se bavi analizom zvuka te generiranjem kvalitetnih signala za upravljanje audio responzivnih sustava te implementacijom istih signala u jednostavni audio responzivni sustav kojim se pokreće simulacija spomenutog algoritma.

Ključne riječi: vizualizacija, FFT, audioanaliza, OpenGL, pojavni algoritam

Visualising Sound in Real Time

Abstract

This paper describes the implementation of a program for real-time sound visualization using the Physarum emergent algorithm. The paper focuses on sound analysis and the generation of high-quality signals for controlling audio-responsive systems, as well as the implementation of these signals in a simple audio-responsive system that drives the simulation of the mentioned algorithm.

Keywords: visualization, FFT, audio analysis, OpenGL, emergent algorithm