Audio efekti u vremenskoj domeni

Balatinec, Josip

SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

ZAVRŠNI RAD br. 1251

AUDIO EFEKTI U VREMENSKOJ DOMENI

Josip Balatinec

Zagreb, lipanj 2024.

SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

ZAVRŠNI RAD br. 1251

AUDIO EFEKTI U VREMENSKOJ DOMENI

Josip Balatinec

Zagreb, lipanj 2024.

SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

Zagreb, 4. ožujka 2024.

ZAVRŠNI ZADATAK br. 1251

Pristupnik:	**Josip Balatinec (0036538079)**
Studij:	Elektrotehnika i informacijska tehnologija i Računarstvo
Modul:	Računarstvo
Mentor:	prof. dr. sc. Ivan Đurek

Zadatak: **Audio efekti u vremenskoj domeni**

Opis zadatka:

Teoretski objasnite efekte koji se provode na audio signalu u vremenskoj domeni. Detaljno objasnite princip rada efekta kašnjenja. Navedite ostale efekte koji se baziraju na efektu kašnjenja. Navedite i objasnite bitne značajke tih audio efekata. Na računalu programirajte i izvedite efekt kašnjenja u stvarnom vremenu s mogućnosti podešavanja osnovnih parametara.

Rok za predaju rada: 14. lipnja 2024.

Zahvaljujem se obitelji i bližnjima na podršci tijekom studiranja te mentoru prof. dr. sc. Ivanu Đureku na strpljivosti i pruženoj pomoći.

Sadržaj

1. Uvod

Zvuk je jedna od najznačajnijih pojava u našoj svakodnevici. Uvelike utječe na naš doživljaj stvarnosti i naše osjećaje. U glazbi, filmovima, reklamama, video igrama i ostalim medijima zvuka, audio efekti igraju veliku ulogu u doživljaju i kvaliteti zvuka.

Audio efekti metode su obrade audio signala koje na neki način utječu na njegove karakteristike. Jedna vrsta audio efekata jesu audio efekti u vremenskoj domeni. Ti efekti doprinose doživljaju prostora i vremena u zvuku koji čujemo pomoću preslikavanja izvornog zvuka te miješanjem preslikanog zvuka s izvornim uz određeni vremenski pomak. Osnovni efekt u vremenskoj domeni jest efekt kašnjenja, a na njemu se baziraju ostali efekti od kojih su neki: *chorus*, *flanging* i *reverb*.

U ovom radu razmatrat ćemo teorijsku pozadinu audio efekata u vremenskoj domeni, pobliže objasniti princip rada efekta kašnjenja, objasniti ostale efekte koji se baziraju na efektu kašnjenja te razmotriti programsku implementaciju efekta kašnjenja u stvarnom vremenu.

2. Obrada audio signala u vremenskoj domeni

2.1. Zvuk

Zvuk je mehanički val koji nastaje titranjem izvora zvuka. Kako izvor titra, u mediju nastaje poremećaj tlaka koji se prenosi preko susjednih čestica najčešće u obliku longitudinalnog vala. To je val u kojem čestice titraju u smjeru širenja vala. Zvuk se u tekućinama i plinovima širi longitudinalno, dok se u krutinama može širiti i transverzalno.

Osnovna svojstva zvuka su:

1. **Frekvencija**: označava broj ponavljajućih ciklusa valnog oblika u jednoj sekundi. Mjeri se u hercima (Hz), te određuje visinu tona. Ljudi najčešće mogu čuti tonove od 20 Hz do 20 kHz. Tonovi niži od 20 Hz nazivaju se infrazvuk, a viši od 20 kHz nazivaju se ultrazvuk.

2. **Amplituda**: označava maksimalnu promjenu tlaka zvučnog vala. Veća amplituda odgovara glasnijem zvuku, dok manja odgovara tišem. Mjeri se u decibelima (dB). Decibel označava odnos između dvije veličine, a pogodan je zbog logaritamske veličine kojom se veći brojevi mogu prikazati pomoću manjih [1].

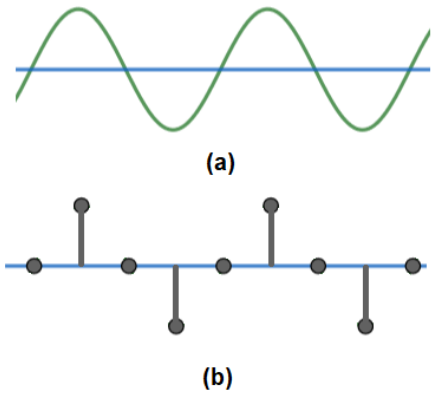
3. **Brzina**: označava put koji zvuk prevali u jednoj sekundi u nekom mediju. U zraku, zvuk se širi brzinom od 343 m/s. Brzina zvuka veća je u vodi i iznosi 1485 m/s.

4. **Valna duljina**: označava udaljenost između dviju točaka u istoj fazi na dva susjedna vala. Manja valna duljina odgovara višim frekvencijama, dok veća valna duljina odgovara nižim frekvencijama.

2.2. Digitalni audio signal

Digitalni audio signal reprezentacija je analognog signala u digitalnom obliku koja se može obrađivati, mijenjati i reproducirati. Sastoji se od diskretnih vrijednosti koje predstavljaju uzorke snimljenog zvuka u određenim vremenskim intervalima. Do tih vrijednosti dolazi se uz pomoć uzorkovanja i kvantizacije.

Uzorkovanje (engl. *sampling*) je proces kojim se kontinuirani analogni signal pretvara u diskretni digitalni signal. A/D pretvarač u određenim vremenskim intervalima mjeri vrijednost analognog signala te dobivene vrijednosti pretvara u niz diskretnih uzoraka. Na slici 2.1. gornji graf prikazuje kontinuirani analogni signal, dok donji graf prikazuje digitalni signal dobiven uzorkovanjem u jednakim vremenskim intervalima.



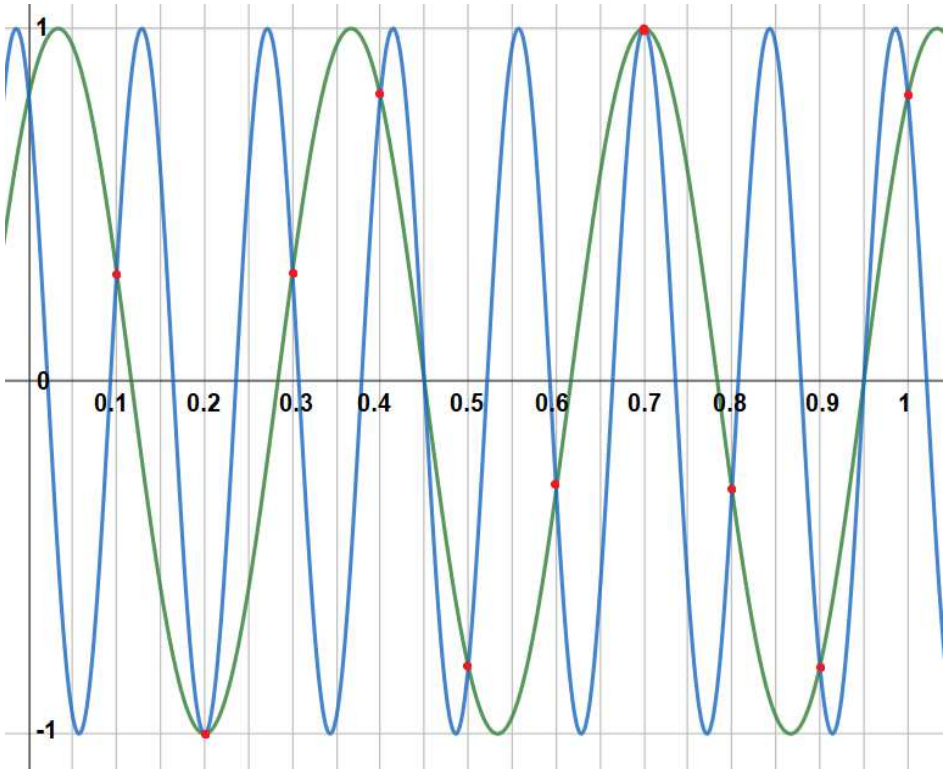
Slika 2.1. Kontinuirani analogni signal (a), digitalna reprezentacija dobivena uzorkovanjem (b). Sliku sam napravio pomoću alata GeoGebra.

Frekvencija uzorkovanja važna je karakteristika uzorkovanja. Označava koliko se puta u sekundi očita vrijednost analognog signala, a jedno očitanje odgovara jednom uzorku. Izražava se u hercima (Hz), a najčešće iznosi 44 100 Hz. Pri višim frekvencijama uzorkovanja više frekvencije zvuka kvalitetnije se snimaju i reproduciraju. Važan pojam kod uzorkovanja jest **Nyquistova frekvencija**. Po Nyquist-Shannonovu teoremu, frekvencija uzorkovanja mora biti najmanje 2 puta veća od najviše frekvencije koja može biti prisutna u signalu koji se uzorkuje. Kada frekvencija uzorkovanja iznosi 44.1 kHz, to povlači da najveća frekvencija koja se može uzorkovati bez izobličenja iznosi 22.05 kHz, što je zadovoljavajuće s obzirom na to da prosječno ljudsko uho ne čuje frekvencije više od 20 kHz.

Izobličenja koja nastaju kod uzorkovanja frekvencija viših od Nyquistove frekvencije

nazivaju se **preklapanja (engl. *aliasing*)**. Do preklapanja dolazi jer je frekvencija uzorkovanja preniska da bi precizno prikazala uzorkovani signal.

Na slici 2.2. prikazane su dvije sinusoide frekvencija 0.3 f_s i 0.7 f_s. Kada bi se izvršilo uzorkovanje frekvencijom f_s, u trenutcima označenim crvenom točkom došlo bi do uzorkovanja. Na svim mjestima uzorkovanja sinusoide se preklapaju, što dovodi do pogrešnog tumačenja uzorkovanog signala jer uzorci mogu predstavljati i sinusoidu frekvencije 0.3 f_s, kao i onu frekvencije 0.7 f_s.



Slika 2.2. Dvije sinusoide frekvencija 0.3 f_s i 0.7 f_s, uzorkovane frekvencijom f_s. Sliku sam napravio pomoću alata GeoGebra i programa Paint.

Sljedeći važan korak jest **kvantizacija (engl. *quantization*)**. To je proces pretvaranja uzoraka u diskretne vrijednosti koje se mogu prikazati određenim brojem bitova. Broj bitova kojim se te vrijednosti prikazuju naziva se **bit dubina (engl. *bit depth*)**. Ako je, na primjer, broj bitova kojim se podaci spremaju jednak 8, to znači da se može prikazati 256 različitih razina signala. Svaki uzorak nastao uzorkovanjem uspoređuje se s dostupnim razinama kvantizacije te se zaokružuje na najbližu razinu. Rezultat zaokruživanja jest pogreška koja se naziva šumom kvantizacije. Što je bit dubina manja, to je veći šum kvantizacije. Za kvantizaciju često se koristi od 16 do 32 bita [2].

Brzina prijenosa (engl. *bit rate*) digitalnog signala označava količinu podataka koja se

obradi u jedinici vremena. Najčešće se izražava u bitovima po sekundi. Brzina prijenosa produkt je frekvencije uzorkovanja, bit dubine i broja kanala koji se koriste. Kada je riječ o broju kanala, mono sustavi imaju 1 kanal, stereo sustavi 2 kanala, a surround sustavi imaju 3 i više kanala. Jednadžba 2.1 predstavlja izračun brzine prijenosa za stereo audio signal uzorkovan frekvencijom 44.1 kHz s bit dubinom od 16 bitova.

$$\text{Brzina prijenosa} = 44\,100\,\text{Hz} \times 16\,\text{bit} \times 2\,\text{kanala} = 1\,411\,200\,\frac{bit}{s} \tag{2.1}$$

2.2.1. Obrada digitalnog signala

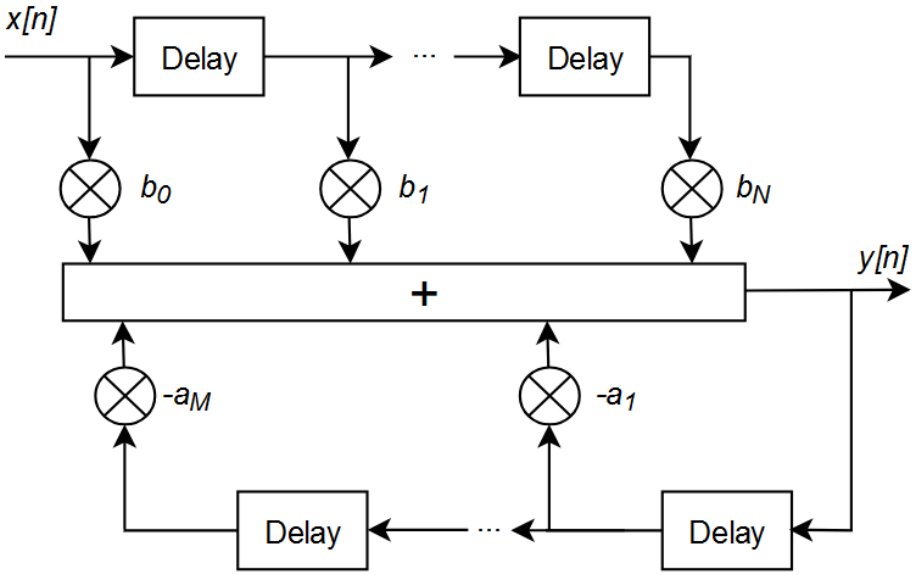
Diskretni signal niz je vrijednosti koje opisuju amplitudu signala u različitim vremenskim točkama. Ulazni signal označuje se kao niz vrijednosti $x[n]$, gdje n predstavlja trenutak u vremenu, dok se izlazni signal označuje kao niz vrijednosti $y[n]$. Jednadžbe 2.2 prikazuju ulazni i izlazni signal s N uzoraka.

$$\begin{aligned} x[n] &= x[0], x[1], ..., x[N-1] \\ y[n] &= y[0], y[1], ..., y[N-1] \end{aligned} \tag{2.2}$$

Jednadžba 2.3 jednadžba je razlike digitalnog filtra u kojem izlaz ovisi o trenutnim i prethodnim ulazima i izlazima. Konstante $b_0, ..., b_N$ i $a_0, ..., a_N$ nazivaju se koeficijenti.

$$y[n] = b_0x[n] + b_1x[n-1] + ... + b_Nx[n-N] - a_1y[n-1] - ... - a_Ny[n-N] \tag{2.3}$$

Na slici 2.3. prikazan je blok dijagram digitalnog filtra čija je jednadžba razlike jednadžba 2.3



Slika 2.3. Blok dijagram digitalnog filtra. Sliku sam napravio pomoću alata draw.io.

Impulsni odziv $h[n]$ predstavlja izlaz iz sustava kada je ulaz Diracov impuls $\delta[n]$. Ulazni signal niz je vrijednosti koje su jednake 0, osim kada je $n = 0$, tada je vrijednost jednaka 1. Impulsnim odzivom može se analizirati ponašanje sustava pomoću najjednostavnijeg ulaznog signala. Jednadžbe 2.4 prikazuju matematički zapis impulsnog odziva.

$$h[n] = y[n]$$
$$x[n] = \delta[n] \equiv \begin{cases} 1, n = 0 \\ 0, n \neq 0 \end{cases} \tag{2.4}$$

Kao primjer, uzmimo jednostavan sustav $y[n] = x[n] + ay[n-1]$. U slučaju kad je koeficijent a jednak 0.5, impulsni odziv $h[n]$ iznosi $1, 0.5, 0.25, 0.125, ...$ Kako $h[n]$ konvergira ka nuli, sustav je stabilan. No, u slučaju kad je koeficijent a jednak 2, $h[n]$ iznosi $1, 2, 4, 8, ...$ Kako $h[n]$ konvergira ka beskonačnosti, sustav nije stabilan.

2.2.2. Z-transformacija

Z-transformacija matematički je alat kojim se jednadžbe razlike iz vremenske domene prebacuju u algebarske jednadžbe u Z-domeni. Koristi se za analizu diskretnih sustava. Z-transformacija signala $x[n]$ definira se kao:

$$Z\{x[n]\} = X(z) = \sum_{n=0}^{\infty} x[n]z^{-n} \tag{2.5}$$

$X(z)$ predstavlja transformirani signal u Z-domeni, a z je kompleksna varijabla definirana kao $z = e^{j\omega}$, gdje je ω normalizirana frekvencija $2\pi f / f_S$.

Z-transformacija jednadžbe 2.3 je:

$$Y(z) = b_0X(z) + b_1z^{-1}X(z) + ... + b_Nz^{-N}X(z) - a_1z^{-1}Y(z) - ... - a_Nz^{-N}Y(z) \tag{2.6}$$

Prijenosna funkcija $H(z)$ prikazuje omjer Z-transformacija izlaznog signala $Y(z)$ i ulaznog signala $X(z)$:

$$H(z) = \frac{Y(z)}{X(z)} = \frac{B}{A} = \frac{b_0 + b_1z^{-1} + ... + b_Nz^{-N}}{1 + a_1z^{-1} + ... + a_Nz^{-N}} \tag{2.7}$$

Zbog jednostavnijeg prikaza, polinome iz jednadžbe 2.7 prikazujemo kao pozitivne potencije od z te ih faktoriziramo:

$$H(z) = \frac{b_0z^N + b_1z^{N-1} + ... + b_N}{z^N + a_1z^{N-1} + ... + a_N} = g\frac{(z - q_1)(z - q_2)...(z - q_N)}{(z - p_1)(z - p_2)...(z - p_N)} \tag{2.8}$$

Polovi i nule prijenosne funkcije pokazuju ponašanje i stabilnost sustava. Polovi su vrijednosti z za koje je nazivnik prijenosne funkcije jednak 0, odnosno za koje funkcija postaje beskonačna. Ako su polovi smješteni unutar jediničnog kruga, sustav je stabilan, no ako su oni izvan kruga, sustav je nestabilan i dolazi do rezonancije. Nule su vrijednosti z za koje je brojnik prijenosne funkcije jednak 0, odnosno za koje je izlaz sustava jednak 0 [3].

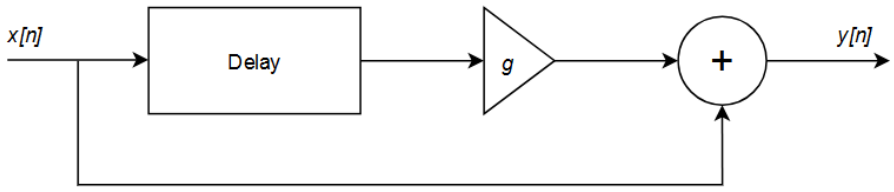
3. Efekt kašnjenja

3.1. Osnove efekta kašnjenja

Efekt kašnjenja (engl. *delay effect*) osnovni je efekt kad je riječ o audio efektima u vremenskoj domeni. U najjednostavnijoj izvedbi, nakon što se reproducira originalni audio signal, nakon određenog vremena kopija tog signala ponovno se reproducira.

Neka je izlazni uzorak audio signala $y[n]$ funkcija ulaznog signala $x[n]$, vremena odgode N i koeficijenta g zakašnjelog signala, prikazana jednadžbom 3.1

$$y[n] = x[n] + g * x[n - N] \tag{3.1}$$



Slika 3.1. Blok dijagram osnovnog efekta kašnjenja. Sliku sam napravio pomoću alata draw.io.

Na slici 3.1. prikazan je blok dijagram osnovnog efekta kašnjenja. Nakon što signal uđe u sustav, jedna kopija šalje se kroz liniju kašnjenja te se množi koeficijentom g, a druga kopija zbraja se sa zakašnjelim signalom koji je prošao kroz liniju kašnjenja te izlazi iz sustava.

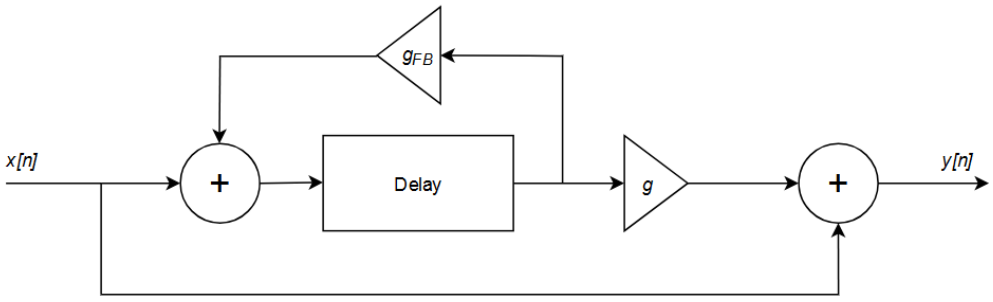
Z-transformacija jednadžbe 3.1 glasi:

$$
\begin{aligned}
Y(z) &= X(z) + g * X(z) * z^{-N} \\
H(z) &= \frac{Y(z)}{X(z)} = 1 + g * z^{-N} = \frac{z^N + g}{z^N}
\end{aligned}
\tag{3.2}
$$

Prijenosna funkcija iz jednadžbi 3.2 nema polove izvan jediničnog kruga te je stoga osnovni efekt kašnjenja stabilan sustav.

Dodavanjem **povratne veze (engl. *feedback*)** u sustav, signal se konstantno vraća na početak sustava. Vraćeni signal zauvijek ostaje u sustavu, no ako je koeficijent pojačanja povratne veze manji od 1, ponavljani signal u nekom trenutku postaje nečujan. Jednadžba 3.3 prikazuje kakav je izlazni signal uz dodatak povratne veze. Komponenta $d[n]$ opisuje zakašnjeli signal $x[n-N]$ uz dodatak prethodno zakašnjelog signala umnoženog s koeficijentom g_{FB}.

$$
\begin{aligned}
y[n] &= x[n] + g * d[n] \\
d[n] &= x[n-N] + g_{FB} * d[n-N]
\end{aligned}
\tag{3.3}
$$



Slika 3.2. Blok dijagram efekta kašnjenja s povratnom vezom. Sliku sam napravio pomoću alata draw.io.

Na slici 3.2. prikazan je blok dijagram efekta kašnjenja s povratnom vezom. Jedna kopija ulaznog signala zbraja se s signalom vraćenim putem povratne veze, prolazi kroz liniju kašnjenja nakon koje se jedna kopija šalje povratnom vezom i množi koeficijentom g_{FB}, a druga se množi koeficijentom g i zbraja se s drugom kopijom ulaznog signala te izlazi iz sustava.

Kako bismo izlaz izrazili isključivo pomoću x i y uzoraka, u prvoj jednadžbi iz jednadžbi 3.4 potrebno je supstituirati $d[n-N]$ pomoću druge jednadžbe iz jednadžbi 3.3

Zatim se iz dobivene jednadžbe izrazi $d[n]$, koji se supstituira pomoću prve jednadžbe iz jednadžbi 3.3 Posljednja jednadžba prikazuje izlazni uzorak $y[n]$ kao funkciju ulaznog signala te prethodnog ulaznog i izlaznog uzorka.

$$\begin{aligned} y[n-N] &= x[n-N] + g * d[n-N] \\ y[n-N] &= x[n-N] + \frac{g}{g_{FB}} * (d[n] - x[n-N]) \end{aligned} \tag{3.4}$$

Nakon supstitucije $d[n-N]$, supstituira se $d[n]$ pomoću prve jednadžbe iz 3.3 te se tako dobije jednadžba 3.5 u kojoj je izlazni uzorak $y[n]$ prikazan kao funkcija zakašnjelog izlaza $y[n-N]$, ulaza $x[n]$ i zakašnjelog ulaza $x[n-N]$.

$$y[n] = g_{FB} * y[n-N] + x[n] + (g - g_{FB}) * x[n-N] \tag{3.5}$$

Z-transformacija jednadžbe 3.5 glasi:

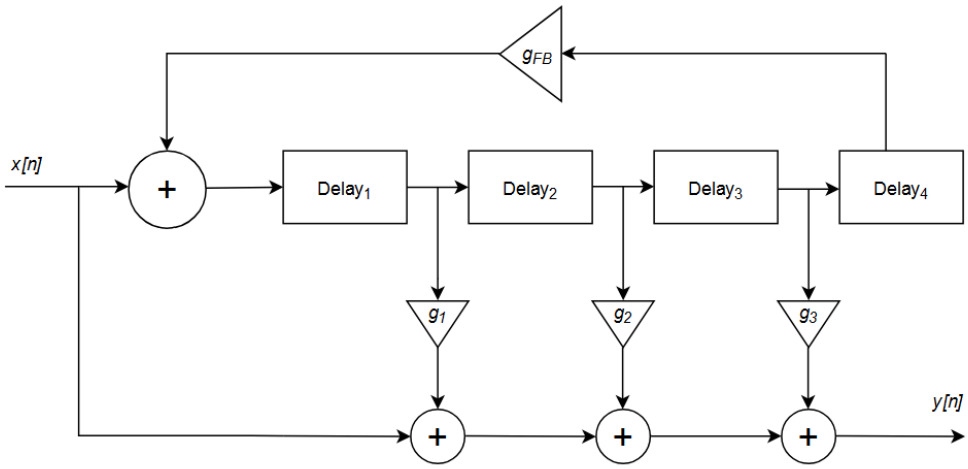
$$\begin{aligned} Y(z) - g_{FB} * Y(z) * z^{-N} &= X(z) + (g - g_{FB}) * X(z) * z^{-N} \\ H(z) = \frac{Y(z)}{X(z)} &= \frac{1 + z^{-N} * (g - g_{FB})}{1 - z^{-N} * g_{FB}} = \frac{z^N + g - g_{FB}}{z^N - g_{FB}} \end{aligned} \tag{3.6}$$

Prijenosna funkcija iz jednadžbi 3.6 polove ima unutar jediničnog kruga u slučaju kad je koeficijent pojačanja povratne veze g_{FB} manji od 1, što znači da je tada sustav stabilan. Ako je koeficijent jednak 1, polovi su točno na jediničnog krugu, što znači da je sustav na granici stabilnosti. Kad je koeficijent veći od 1, polovi su izvan jediničnog kruga, što znači da je sustav nestabilan jer će amplituda izlaznog signala konstantno rasti. Zvuk postaje sve glasniji te dolazi do šuma i izobličenja signala.

3.2. Ostale izvedbe efekta kašnjenja

Multi-tap kašnjenje efekt je kašnjenja kod kojeg u liniji kašnjenja postoji više *tapova*. *Tapom* se naziva trenutak kad signal izlazi iz linije kašnjenja, a svaki *tap* može imati različito vrijeme kašnjenja i različiti koeficijent pojačanja.

Na slici 3.3. prikazan je blok dijagram *multi-tap* efekta kašnjenja s 3 *tapa*. Jedna kopija ulaznog signala šalje se prema izlazu, a druga prema *tapovima*. Nakon svakog vremena kašnjenja svakog *tapa* jedna kopija signala šalje se prema idućem *tapu*, dok se druga množi koeficijentom pojačanja i zbraja s ulaznim signalom. Nakon svih *tapova*, signal dolazi do još jednog kašnjenja i povratnom vezom vraća se na početak sustava.

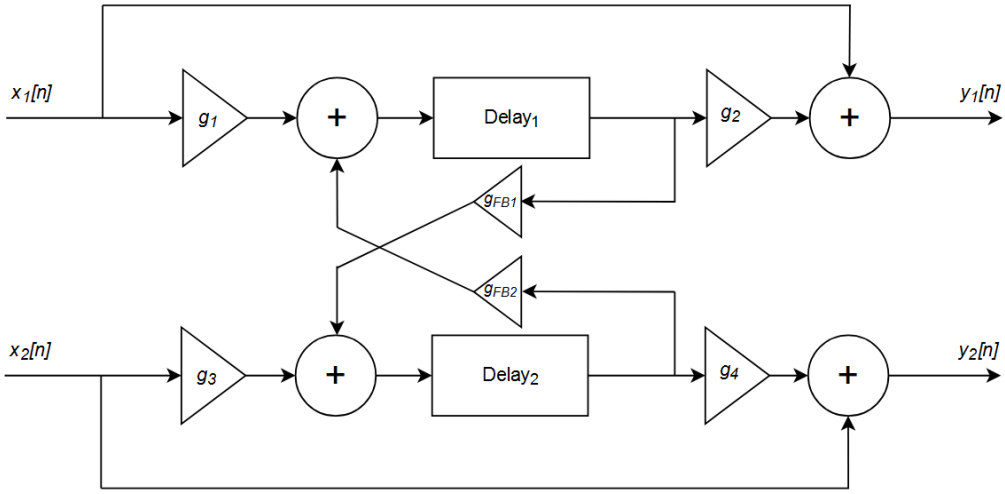


Slika 3.3. Blok dijagram *multi-tap* efekta kašnjenja. Sliku sam napravio pomoću alata draw.io.

Ping-pong kašnjenje efekt je kašnjenja koji je moguće ostvariti kod sustava s više kanala. Zakašnjeli signal naizmjenično se šalje između kanala i time dolazi do efekta odskakivanja, kao kod ping-pong loptice. Kod stereo sustava, postoje dvije linije kašnjenja, koje signal izmjenjuju preko povratnih veza. Na ulaze linija kašnjenja može biti doveden isti signal, no može biti i različit.

Na slici 3.4. prikazan je blok dijagram *ping-pong* efekta kašnjenja u stereo sustavu. Na svaku liniju kašnjenja doveden je zaseban ulazni signal. Jedna kopija ulaznog signala šalje se na izlaz sustava, dok se druga množi koeficijentom pojačanja i zbraja s zakašnjelim signalom iz suprotne linije. Nakon prolaska kroz kašnjenje, jedna kopija signala šalje povratnom vezom u suprotnu liniju, a druga se množi koeficijentom pojačanja, zbraja s

ulaznim signalom i izlazi iz sustava.



Slika 3.4. Blok dijagram *ping-pong* efekta kašnjenja. Sliku sam napravio pomoću alata draw.io.

4. Efekti bazirani na efektu kašnjenja

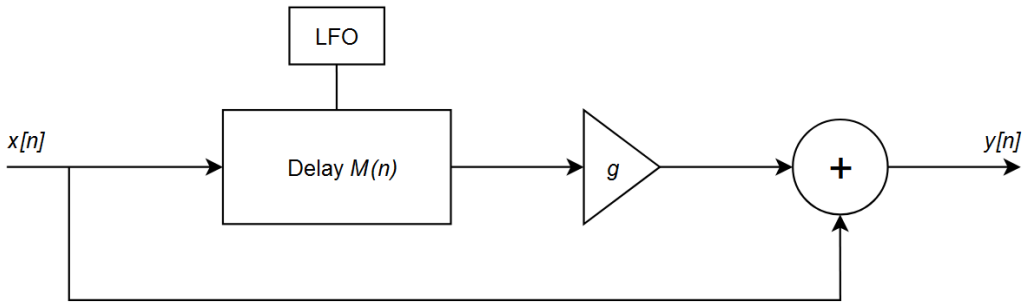
4.1. Chorus efekt

Chorus efekt ime je dobio po tome što stvara iluziju da se zvuk sastoji od više zvukova koji dolaze iz više izvora, kao što je to slučaj u zboru ili orkestru gdje pjevači i glazbenici sviraju u isto vrijeme.

Jednadžba 4.1 prikazuje kako izlazni signal $y[n]$ ovisi o trenutnom i zakašnjelom ulaznom signalu. $M[n]$ predstavlja promjenjivu vrijednost kašnjenja. Vrijednost $M[n]$ određena je niskofrekvencijskim oscilatorom (engl. *low frequency oscillator*), što je generator signala niske frekvencije (najčešće sinusnog oblika [2]).

Na slici 4.1. prikazan je blok dijagram *chorus* efekta. Generator signala *LFO* utječe na vrijeme kašnjenja ulaznog signala. Kod ovog efekta ne postoji povratna veza.

$$y[n] = x[n] + gx[n - M[n]] \tag{4.1}$$



Slika 4.1. Blok dijagram *chorus* efekta. Sliku sam napravio pomoću alata draw.io.

Osnovni parametri *chorus* efekta su:

1. **Dubina**: odnos zakašnjelog i originalnog signala. Pri vrijednosti 0, izlazni signal jednak je ulaznom i nema efekta, dok je pri vrijednosti 1 efekt najizraženiji.

2. **Kašnjenje**: vrijednost osnovnog kašnjenja. Najčešće iznosi od 20 do 30 ms.

3. **Raspon modulacije**: amplituda signala u LFO-u. Utječe na vrijednost $M[n]$. Najčešće iznosi od 1 do 10 ms.

4. **Brzina modulacije**: vrijednost frekvencije signala u LFO-u. Najčešće iznosi od 0.1 do 3 Hz.

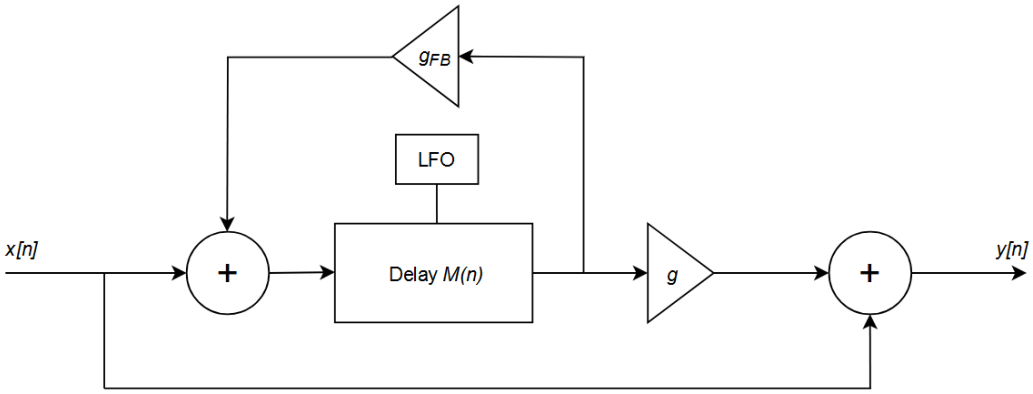
4.2. Flanging efekt

Flanging efekt zvuku dodaje efekt "lepršavosti". Kao i kod *chorus* efekta, iznos kašnjenja moduliran je LFO-om, no sveukupna vrijednost kašnjenja je manja. Posljedica vrlo kratkog kašnjenja jesu konstruktivne i destruktivne interferencije signala. Ako su originalni i zakašnjeli signal u fazi, kod izlaznog signala dolazi do frekvencijskog pojačanja, što je konstruktivna interferencija. No, ako su ta dva signala u protufazi, kod izlaznog signala dolazi do frekvencijskog pojačanja, što je destruktivna interferencija.

Jednadžba 4.2 ista je kao kod *chorus* efekta, gdje $M[n]$ predstavlja varijabilnu vrijednost kašnjenja.

Na slici 4.2. prikazan je blok dijagram *flanging* efekta s povratnom vezom, no ona nije obavezna.

$$y[n] = x[n] + gx[n - M[n]] \tag{4.2}$$



Slika 4.2. Blok dijagram *flanging* efekta s povratnom vezom. Sliku sam napravio pomoću alata draw.io.

Osnovni parametri *flanging* efekta su:

1. **Dubina**: odnos zakašnjelog i originalnog signala.
2. **Kašnjenje**: vrijednost osnovnog kašnjenja. Najčešće iznosi od 1 do 10 ms.
3. **Raspon modulacije**: amplituda signala u LFO-u. Najčešće iznosi od 1 do 10 ms.
4. **Brzina modulacije**: vrijednost frekvencije signala u LFO-u. Najčešće iznosi od 0.1 do 10 Hz.
5. **Koeficijent povratne veze**: vrijednost pojačanja povratne veze.

4.3. Reverb efekt

Reverb efekt (efekt odjeka) bazira se na načinu na koji zvuk dolazi od svog izvora do slušatelja u prostoriji. Osim direktnog puta do slušatelja, zvuk se reflektira od površina prostorije, a te refleksije imaju različita vremena kašnjenja i razine intenziteta te se dijele na rane i kasne refleksije. Broj i intenzitet refleksija određen je veličinom i oblikom prostorije te materijalom površina u prostoriji.

Gate reverb izvedba je ovog efekta s određenim vremenom trajanja odjeka. Dodatni parametri kojima se mogu podešavati karakteristike efekta kod ove izvedbe: *gate time, gate decay time* i *gate threshold*.

Reverse reverb izvedba je ovog efekta na način da se iz tiših refleksija ide ka glasnijima.

Osnovni parametri *reverb* efekta su:

1. **Vrijeme odjeka**: vrijednost trajanja odjeka. Najčešće u milisekundama.
2. ***Predelay***: vrijednost kašnjenja ranih i kasnih odjeka.
3. **Dubina**: odnos originalnog signala i odjeka.
4. **Difuzija**: količina ranih refleksija
5. **Gustoća**: količina kasnih refleksija.
6. **Filtriranje**: jačina gušenja visokih frekvencija.
7. ***Gate time***: vrijeme trajanja odjeka.
8. ***Gate decay time***: brzina jenjavanja odjeka.
9. ***Gate threshold***: razina signala na kojoj odjek prestaje.

5. Programska izvedba efekta kašnjenja

Programska implementacija napravljena je u programskom jeziku *C++* korištenjem radnog okvira *JUCE* u razvojnom okruženju *Visual Studio 2022*. Implementacija je razvijena i testirana na računalu s procesorom *AMD Ryzen 5 5600H*, 16 GB radne memorije i operacijskim sustavom *Windows 10*.

5.1. Radni okvir JUCE

JUCE (*Jules' Utility Class Extensions*) radni je okvir otvorenog koda koji se koristi za izradu računalnih i mobilnih audio aplikacija i razvoj dodataka (engl. *plug-in*). Nastao je 2004. godine, a danas je dostupan u 7. inačici. Značajke kojima se ovaj radni okvir ističe jesu:

- podrška za platforme *Windows, Linux, macOs, Android* i *iOS*,
- ugrađene komponente za obradu zvuka,
- grafičko korisničko sučelje.

JUCE omogućuje jednostavno korištenje standardnih *C++* biblioteka, što pruža širok spektar mogućnosti za razvijanje kako jednostavnih, tako i kompleksnih audio aplikacija i dodataka. Modularna arhitektura omogućuje odabir i korištenje samo onih dijelova radnog okvira koji su potrebni, čime dolazi do optimiziranja performansi aplikacija.

Kada je riječ o razvoju dodataka, *JUCE* pruža podršku za razvoj dodataka u formatima *VST3, AU* i *AAX*. Sveobuhvatna podrška postoji za audio ulaze i izlaze, a također i za *MIDI* kontrolere. Za ovaj radni okvir postoji opširna dokumentacija i mnoštvo izvora informacija na internetu, što rezultira olakšanjem učenja korištenja okvira i razvoja novih projekata.

5.2. Implementacija

Cilj programskog dijela ovog rada bila je izvedba efekta kašnjenja u stvarnom vremenu s mogućnosti podešavanja osnovnih parametara.

5.2.1. Organizacija projekta

Rad se sastoji od tri datoteke koje zajedno čine osnovu aplikacije. Kako *JUCE* automatski generira osnovnu strukturu projekta, odmah se može početi s dodavanjem željenih funkcionalnosti aplikacije. Tri datoteke koje čine ovaj projekt jesu:

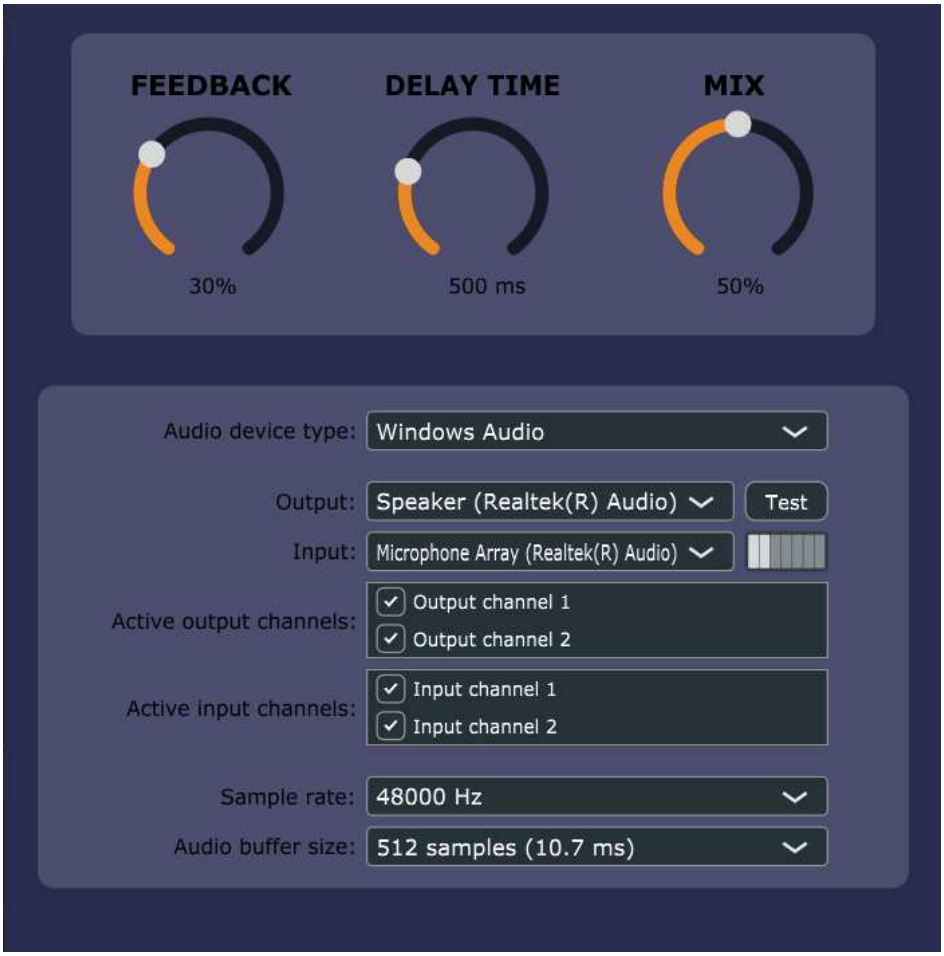
1. **Main.cpp**: Ova datoteka sadrži osnovni kod za pokretanje aplikacije. U većini *C++* aplikacija u ovoj datoteci nalazila bi se `main()` funkcija, ali ona je u kontekstu radnog okvira *JUCE* zamijenjena makro naredbom `START_JUCE_APPLICATION`, koja aplikaciju inicijalizira pomoću klase `DelayApplication`. Unutar te klase sadržana je funkcija `initialise()` koja je zadužena za inicijalizaciju glavnog prozora aplikacije, poziva se pri pokretanju programa, a u njoj se kreira objekt klase `MainWindow`. Klasa `MainWindow` postavlja osnovne parametre prozora aplikacije. Za pravilno zatvaranje aplikacije zaslužne su funkcije `shutdown()` i `systemRequestedQuit()`.

2. **MainComponent.h**: Ova zaglavna datoteka sadrži deklaraciju klase `MainComponent` koja predstavlja glavnu komponentu aplikacije. Ova klasa nasljeđuje klase `juce::AudioAppComponent` i `juce::Slider::Listener`, što omogućuje obradu audio signala i upravljanje klizačima iz korisničkog sučelja. Unutar klase deklarirane su funkcije ključne za obradu audio signala kao što su `prepareToPlay()` i `getNextAudioBlock()`, te funkcije koje omogućuju reakcije na interakcije s klizačima kao što su `sliderValueChanged()` i `sliderDragEnded()`. Klasa sadrži i privatne članove koji predstavljaju grafičke komponente korisničkog sučelja, te `juce::dsp::DelayLine` objekt kojim se omogućuje implementacija glavne funkcionalnosti efekta kašnjenja.

3. **MainComponent.cpp**: Ova datoteka sadrži sav kod ključan za implementaciju efekta kašnjenja. Detaljniji opis sadržaja ove datoteke nalazi se u potpoglavlju 5.2.3.

5.2.2. Grafičko korisničko sučelje

Grafičko korisničko sučelje, prikazano na slici 5.1., razvijeno je pomoću radnog okvira *JUCE*.



Slika 5.1. Grafičko korisničko sučelje aplikacije za efekt kašnjenja.

Grafičko korisničko sučelje podijeljeno je u dva dijela: dio s klizačima za upravljanje parametrima efekta kašnjenja i dio s izbornicima za upravljanje postavkama audio uređaja.

Dio s klizačima za upravljanje parametrima efekta kašnjenja smješten je u gornjem dijelu sučelja. Sastoji se od tri klizača, a iznad svakog je smješten naziv parametra kojim se upravlja, a ispod klizača nalaze se trenutne brojčane vrijednosti parametara.

Lijevim klizačem mijenja se vrijednost povratne veze (engl. *feedback*), a vrijednosti klizača su u granicama od 0% do 100%. Pri vrijednosti od 0% nema povratne veze i izlazni audio signal je jednak ulaznom. Pri nižim vrijednostima audio signal ponavlja se

manji broj puta, a pri većim vrijednostima ponavlja se više puta. Pri vrijednosti od 100%, audio signal ponavlja se beskonačno mnogo puta, to jest sve dok korisnik klizačem ne promijeni vrijednost.

Srednjim klizačem mijenja se vrijeme između ulaznog i zakašnjelog audio signala (engl. *delay time*), a vrijednosti klizača su u granicama od 0 ms do 2000 ms. Pri vrijednosti od 0 ms nema kašnjenja između ulaznog i zakašnjelog signala i stoga je izlazni audio signal jednak ulaznom.

Desnim klizačem mijenja se omjer "suhog" i "mokrog" audio signala (engl. *dry/wet mix*), a vrijednosti klizača su u granicama od 0% do 100%. Pri vrijednosti od 0% potpuno prevladava "suhi" signal, te je zbog toga izlazni signal jednak ulaznom, a pri vrijednosti od 100% izlazni signal isključivo je obrađeni signal.

Dio s izbornicima za upravljanje postavkama audio uređaja smješten je u donjem dijelu sučelja. Sastoji se od `juce::AudioDeviceSelectorComponent` komponente koja sadrži 7 izbornika. Sadržani izbornici su:

1. Izbornik tipa audio uređaja: omogućuje odabir tipa audio sustava podržanih od strane računala. Odabir utječe na opcije drugih izbornika.
2. Izbornik izlaznog uređaja: omogućuje odabir uređaja koji će se koristiti za reprodukciju zvuka.
3. Izbornik ulaznog uređaja: omogućuje odabir uređaja koji će se koristiti za snimanje zvuka.
4. Izbornik aktivnih izlaznih kanala: omogućuje odabir kanala izlaznog uređaja koji će se koristiti za reprodukciju zvuka.
5. Izbornik aktivnih ulaznih uređaja: omogućuje odabir kanala ulaznog uređaja koji će se koristiti za snimanje zvuka.
6. Izbornik stope uzorkovanja: omogućuje odabir stope uzorkovanja koja će se koristiti za audio uređaje.
7. Izbornik veličine međuspremnika: omogućuje odabir veličine međuspremnika koja će se koristiti.

5.2.3. Glavne funkcionalnosti

U datoteci *MainComponent.cpp* implementirana je klasa *MainComponent* i definira se njeno ponašanje. Navedena klasa upravlja svim ključnim funkcionalnostima aplikacije.

Klasa ***MainComponent*** glavna je komponenta aplikacije. Nasljeđuje klasu *juce::AudioAppComponent*, koja je temeljna klasa za rad na audio aplikacijama i omogućuje upravljanje audio ulazima i izlazima.

U konstruktoru inicijalizira se objekt *otherDeviceManager* pomoću *initialise()* metode. Time se otvara set audio uređaja. Pomoću metode *reset()*, *audioSettings* biva postavljen na novu instancu *juce::AudioDeviceSelectorComponent*, što omogućuje odabir audio uređaja i ostalih postavki unutar sučelja aplikacije. Poziva se funkcija *design()*, u kojoj se definira sav raspored i izgled vizualnih elemenata u korisničkom sučelju te se svakom klizaču dodaje slušatelj (engl. *listener*). Funkcija *setSize()* postavlja dimenzije prozora aplikacije. Na posljetku provjerava se jesu li potrebne dozvole za snimanje zvuka te se zahtijevaju po potrebi. Nakon toga inicijaliziraju se dva ulazna i izlazna audio kanala pomoću funkcije *setAudioChannels()*. U destruktoru pomoću funkcije *shutdownAudio()* na siguran se način isključuju audio uređaji i oslobađaju se korišteni resursi pri izlasku iz aplikacije.

Funkcija ***sliderValueChanged()*** služi za obradu promjena vrijednosti klizača. Funkcija biva automatski pozvana svaki put kada se promijeni položaj nekog od klizača. Slušatelj (engl. *listener*) je komponenta koja "sluša" promjene na pojedinom klizaču i obavještava ovu funkciju svaki put kada dođe do interakcije s klizačem. U funkciji se provjerava na kojem klizaču se dogodila promjena, zatim se pomoću metode *getValue()* dobiva brojčana vrijednost tipa *double*, ona se množi odgovarajućim brojem te se dobiveni rezultat, nakon pretvorbe u cjelobrojnu vrijednost pomoću operacije *static_cast*, sprema u varijablu. Ta varijabla se nadalje koristi da bi se pomoću metode *setText()* postavila vrijednost ispod odgovarajućeg klizača u korisničkom sučelju.

Funkcija ***prepareToPlay()*** poziva se na početku rada aplikacije ili pri promjeni neke od postavki audio uređaja. Služi za inicijalizaciju i pripremu audio procesiranja te osigurava da su svi procesi koji rade s audio signalom pravilno konfigurirani. Funkcija prima 2 parametra:

1. *samplesPerBlockExpected*: očekivani broj uzoraka po bloku koji će biti obrađeni. To

je broj uzoraka koji će međuspremnik sadržavati pri svakom prolazu procesiranja.

2. *sampleRate*: frekvencija uzorkovanja izražena u Hz. To je broj uzoraka audio signala koji se obrađuje u sekundi.

U funkciji stvara se *juce::dsp::ProcessSpec* struktura, koja sadržava osnovne informacije o konfiguraciji audio uređaja kao što su frekvencija uzorkovanja, maksimalna veličina blokova i broj kanala.

Nadalje, pomoću metode *reset()*, ponovno se postavljaju unutarnje varijable objekta *myDelay*. Objekt *myDelay* instanca je klase *juce::dsp::DelayLine*, koja implementira liniju kašnjenja, što je osnovna komponenta izvedbe efekta kašnjenja. Zatim se pomoću metode *prepare()* i postavljenih specifikacija prethodno navedene strukture objekt *myDelay* inicijalizira kako bi bio spreman za rad. Na kraju, frekvencija uzorkovanja dobivena iz parametra funkcije sprema se u varijablu *mySampleRate*.

Nakon poziva *prepareToPlay()* funkcije, funkcija ***getNextAudioBlock()*** poziva se kontinuirano kako bi dohvatila dolazne audio blokove, obradila ih te generirala izlazni audio signal. Ova funkcija uobičajeno se poziva na sistemskoj niti visokog prioriteta [4]. Ulazni parametar ove funkcije jest referenca na strukturu *juce::AudioSourceChannelInfo*, imenom *bufferToFill*, koja sadrži informacije o međuspremniku u kojem je spremljen niz uzoraka ulaznog audio signala.

Varijabla *currentAudioDevice* pokazivač je na audio uređaj koji je trenutno u uporabi, a dobiven je pomoću metode *getCurrentAudioDevice()*. Varijable *totalNumInputChannels* i *totalNumOutputChannels* predstavljaju trenutni broj aktivnih kanala ulaznog, odnosno izlaznog audio uređaja. Nakon inicijalizacije navedenih varijabli, ako je aktivno više izlaznih nego ulaznih kanala, pomoću *for* petlje brišu se kanali u međuspremniku koji se ne koriste.

Nadalje, u varijable *feedbackValue* i *dryWetValue* spremaju se trenutne vrijednosti odgovarajućih klizača. U slučaju da je bilo koja od tih varijabli jednaka 0, objekt *myDelay* prestaje dodavati efekt kašnjenja u izlazni audio signal. No, ako su obje varijable različite od 0, dolazi se do ugniježđene *for* petlje. Vanjska petlja iterira toliko puta koliko je uzoraka spremljenih u međuspremniku, dok unutarnja petlja iterira jednako koliko ima aktivnih kanala.

Varijable *inSamples* i *outSamples* pokazivači su koji iz *bufferToFill* metodama *getReadPointer()* i *getWritePointer* dobivaju pristup nizovima koji sadrže ulazne i izlazne

uzorke audio signala za određeni kanal. Varijabla *inSamples* koristi se za čitanje trenutnih uzoraka audio signala u određenom kanalu, dok se varijabla *outSamples* koristi za pisanje obrađenih uzoraka audio signala u isti kanal.

Iz objekta *myDelay*, pomoću metode *popSample()*, u varijablu *delayedSample* sprema se uzorak audio signala iz trenutnog kanala. Slijedi izračun novog uzorka *inDelay*, koji se dobiva zbrajanjem trenutnog ulaznog uzorka i umnoška varijabli *feedbackValue* i *delayedSample*. Dobiveni uzorak vraća se u objekt *myDelay* u trenutni kanal. Na kraju, izlazni uzorak dobiva se zbrajanjem ulaznog i zakašnjelog uzorka, ulazni uzorak pomnožen vrijednošću *(1 - dryWetValue)*, a zakašnjeli vrijednošću varijable *dryWetValue*.

Nakon prolaza svih iteracija ugniježđenih petlji, funkcija *getNextAudioBlock()* ponovno se poziva, a to se događa sve dok rad aplikacije nije prekinut.

6. Zaključak

U ovome radu glavna tema bili su načini obrade digitalnog audio signala u vremenskoj domeni, s posebnim naglaskom na efekt kašnjenja. U uvodnom dijelu predstavljene su osnovne karakteristike zvuka i digitalnog audio signala, uključujući procese uzorkovanja, kvantizacije i Z-transformacije.

U nastavku rada detaljno je analiziran efekt kašnjenja. Opisane su i analizirane osnovna izvedba i izvedba s povratnom vezom, kao i *multitap* i *ping-pong* izvedbe. Opisane su osnove efekata baziranih na efektu kašnjenja, poput *chorus, flanging* i *reverb* efekata. Opisani su njihovi osnovni parametri.

Opisana je programska izvedba efekta kašnjenja, napravljena u radnom okviru *JUCE*. Opisana je organizacija projekta, objašnjeno je grafičko korisničko sučelje aplikacije, te se objašnjene glavne funkcionalnosti aplikacije. U budućem radu na aplikaciji, implementacija efekata poput *chorus* i *reverb* efekta korisniku bi pružila više mogućnosti oblikovanja zvuka.

Literatura

[1] I. Đurek, H. Domitrović, i M. Horvat, “Audiotehnika, materijali za predavanja”, https://www.fer.unizg.hr/_download/repository/Skripta_-_kompletna_v2_03.pdf.

[2] J. Reiss i A. McPherson, *Audio Effects: Theory, Implementation and Application*. CRC Press, 2014.

[3] R. G. Lyons, *Understanding Digital Signal Processing, 3rd Edition*. Pearson, 2010.

[4] JUCE dokumentacija, https://docs.juce.com/master/.

Sažetak

Audio efekti u vremenskoj domeni

Zvuk je dio naše svakodnevice, a u digitalnom obliku obogaćujemo ga koristeći audio efekte. Digitalni audio signal dobiva se uzorkovanjem i kvantizacijom, dok za analizu sustava koristimo Z-transformacija. Efekti u vremenskoj domeni doprinose doživljaju prostora i vremena u zvuku,a osnovni među njima je efekt kašnjenja. Njegove izvedbe uključuju *multi-tap* i *ping-pong* efekte kašnjenja. Efekti poput *chorus, flanging* i *reverb* efekta baziraju se na efektu kašnjenja. Efekt kašnjenja programski je implementiran koristeći radni okvir JUCE. Aplikacija korisniku omogućuje kontrolu osnovnih parametara efekta.

Ključne riječi: zvuk, audio efekti, digitalni audio signal, vremenska domena, efekt kašnjenja, JUCE

Abstract

Time domain audio effects

Sound is a part of our everyday lives, and in digital form we enrich it by using audio effects. A digital audio signal is obtained by sampling and quantization, while for system analysis we use Z-transform. Effects in the time domain contribute to the experience of space and time in sound, and the main one among them is the delay effect. Its forms include multi-tap and ping-pong delay effects. Effects like chorus, flanging, and reverb effects are based on the delay effect. The delay effect is programmatically implemented by using the JUCE framework. The application allows users to control the basic parameters of the effect.

Keywords: sound, audio effects, digital audio signal, time domain, delay effect, JUCE