REV	DATA	ZMIANY
0.1	13.01.2020	*WiktorLechowicz (wiktorle@student.agh.edu.pl)*

EFEKT GITAROWY

Autor: Wiktor Lechowicz

Akademia Górniczo-Hutnicza

Spis treści

Lista oznaczeń

f_p	Częstotliwość próbkowania
DAC	Digital to analog converter
FFT	Fast Fourier Transform
Hz	Hertz
kHz	kiloHertz
LUT	Look-up table
GUI	Graphical user interface
OOD	Object-Oriented Design

1. Wstęp

Dokument dotyczy programu do dodawania symulacji efektów gitarowych do nagrań dźwiękowych.

Cyfrowe efekty audio służą do przetwarzania informacji o dźwięku w celu symulacji zjawisk fizycznych zachodzących dla fal mechanicznych takich jak między innymi echo, pogłos lub efekt Dopplera. Efekty operujące na nagranych wcześniej próbkach są alternatywą dla analogowych lub cyfrowych efektów działających w czasie rzeczywistym. Pierwsze efekty działały na zasadzie modyfikacji sygnału w postaci przebiegu napięcia za pomocą obwodów elektrycznych lub na zasadzie zmiany położenia elementu, z którego wydobywał się dźwięk względem słuchacza. Cyfrowy efekt gitarowy, który jest realizowany przez program, którego dotyczy ta dokumentacja wykonuje odpowiednie operacje na ciągach próbek sygnału zapisanych w pliku WAV w celu osiągniecia pożądanych zmian.

2. Opis wymagań funkcjonalnych

Program ma pozwalać na dodanie do pliku WAV czterech efektów gitarowych:
- Bit crusher – symulujący zmniejszenie częstotliwości próbkowania sygnału i rozdzielczości bitowej przetwornika. Wynikiem zastosowanie tego efektu ma być dźwięk o charakterze zbliżonym do brzemienia dźwięku generowanego przez 8 – bitowe układy scalone „chiptune".

- Overdrive – wzmacniający amplitudę sygnału oraz ograniczający wartość próbek powyżej zadanej wartości.

- Delay – efekt echa lub pogłosu.

- Tremolo – symulujący dźwięk uzyskiwany przy zastosowaniu głośników obrotowych „Leslie speaker".

Możliwe jest odtwarzanie pliku w formacie .WAV w wersji oryginalnej oraz po przetworzeniu przez powyższe efekty w dowolnej kolejności i konfiguracji.

W czasie odtwarzania dźwięku możliwa będzie zmiana parametrów nastaw każdego z efektów, a zmiany te mają być na bieżąco wprowadzanie do sygnału dźwiękowego. Obsługa oprogramowania przez użytkownika powinna być realizowana za pomocą interfejsu graficznego.

3. Analiza problemu.

Sygnał dźwiękowy dla każdego kanału nagrania jest zapisywany w pliku .WAV jako wektor próbek napięcia proporcjonalnego do napięcia na wyjściu przetwornika przetwarzającego ciśnienie akustyczne na przebieg napięcia. Wymiar wektora k jest związany z częstotliwością próbkowania oraz czasem nagrania t zależnością:

$$k = f_p * t$$

Wartości próbek przyjmują wartości całkowite z zakresu -2^(N–1) do 2^(N-1) - 1, gdzie N jest rozdzieloczścią DAC. W programie wartości wektora próbek zostały znormalizowane do zakresu od -1 do 1 i przedstawione jako liczby zmiennoprzecinkowe ze skończonego zbioru wartości. Próki odczytane z pliku są przechowywane w wektorze liczb zmiennoprzecinkowych w celu poddania ich przetwarzaniu lub odtworzeniu jako dźwięk.

Przetwarzanie sygnału może być realizowane w sposób blokowy w osobnym wątku lub natychmiast dla wszystkich próbek. Ze względu na prostotę implementacji został wybrany sposób z przetwarzaniem całościowym. Z tego powodu wybór algorytmów przetwarzających został ograniczony do tych działających w dziedzinie czasu a zatem efekty i filtry wykorzystujące FFT nie zostaną zrealizowane na tym etapie projektu.

Realizacja kolejnych efektów została przedstawiona poniżej:

Bit crusher	Redukcja częstotliwości próbkowania: *Z okresem N próbek z wektora zostaje wybierana n-ta próka, a do kolejnych N próbek zostaje wpisana jej wartość.* Redukcja rozdzielczości przetwornika: *Wartośći próbek są zaokrąglane tak, aby przyjmowały wyłącznie wartośći ze zbioru wynikającego z liczby bitów przetwornika.*
Overdrive	*Amplituda sygnału jest zwiększana proporcjonalnie do wartości wprowadzonej przez użytkownika, a następnie próbki o wartośći większej niż zadana wartość A przyjmują wartość A.*
Delay	*Do sygnału jest dodawana jego kopia przesunięta w czasie.*
Tremolo	*Amplituda sygnału jest modulowana przebiegiem sinusoidalnym o niskiej częstotliwości (kilku – kilkunastu Hz)*

Efekty wykorzystujące funkcje matematyczne takie jak sin(x) mogą obliczać wartości tych odwzorowań. Na kolejnym etapie projektu funkcje te mogą zostać zaimplementowane w postaci pamięci LUT w celu poprawienia wydajności.

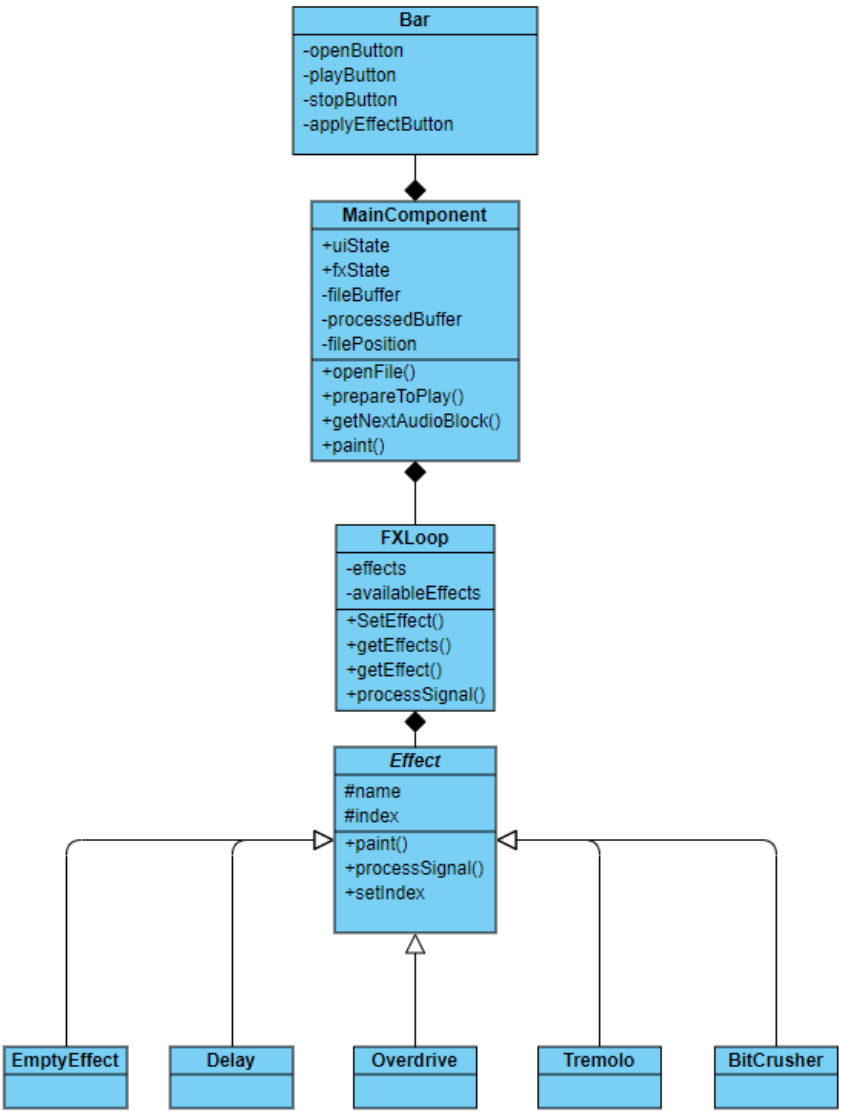
Oprogramowanie ma być przystępne w obsłudze dla użytkownika, zatem do realizacji interfejsu musi zostać użyta biblioteka umożliwiająca realizację GUI. Jednym z narzędzi umożliwiającym implementacjętych założeń w języku C++ jest framework JUCE

4. Projekt techniczny

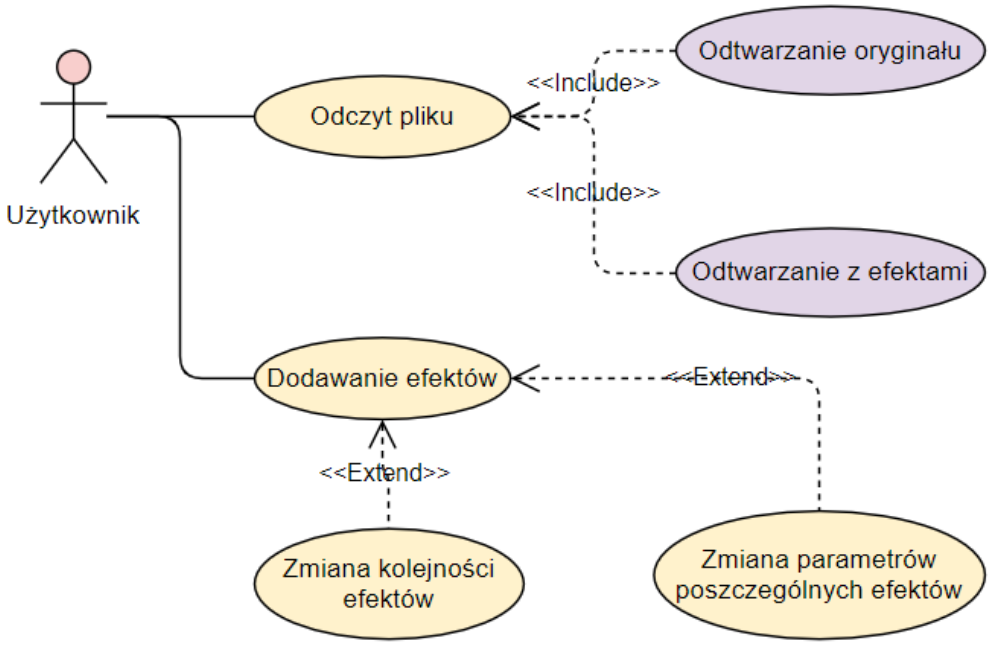
4.1. Przyjęte założenia

Do realizacji projektu założono język programowania C++ oraz framework JUCE. Oprogramowanie ma działać prawidłowo z plikami w formacie .WAV o długośći poniżej dziesięciu minut nagranymi z częstotliwością próbkowania 44.1 kHz i dowolną rozdzielczością bitową. Docelową platformą są komputery osobiste i laptopy z oprogramowaniem Microsoft Windows 10 64-bit o rozdzielczości ekranu nie mniejszej niż 1024x768 pikseli.

4.2. Diagram klas

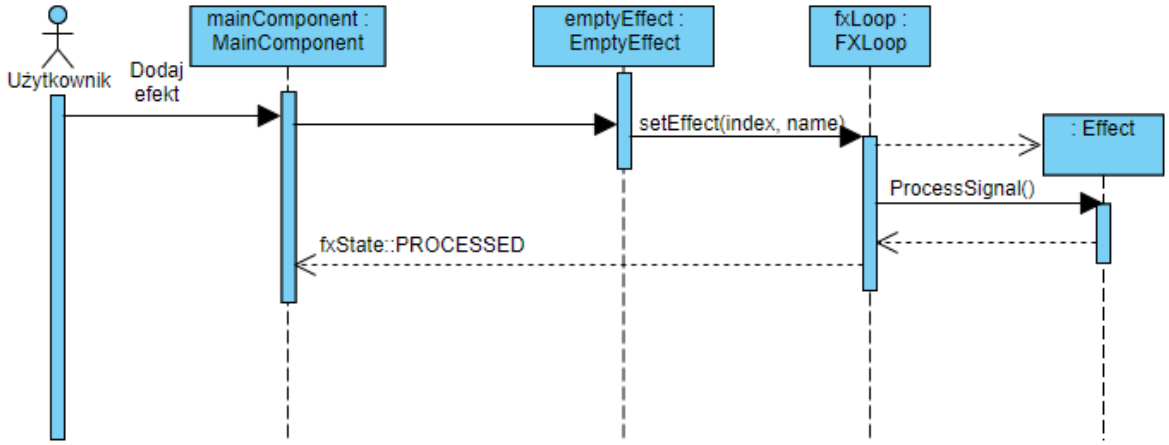


4.3. Diagram przypadków użycia

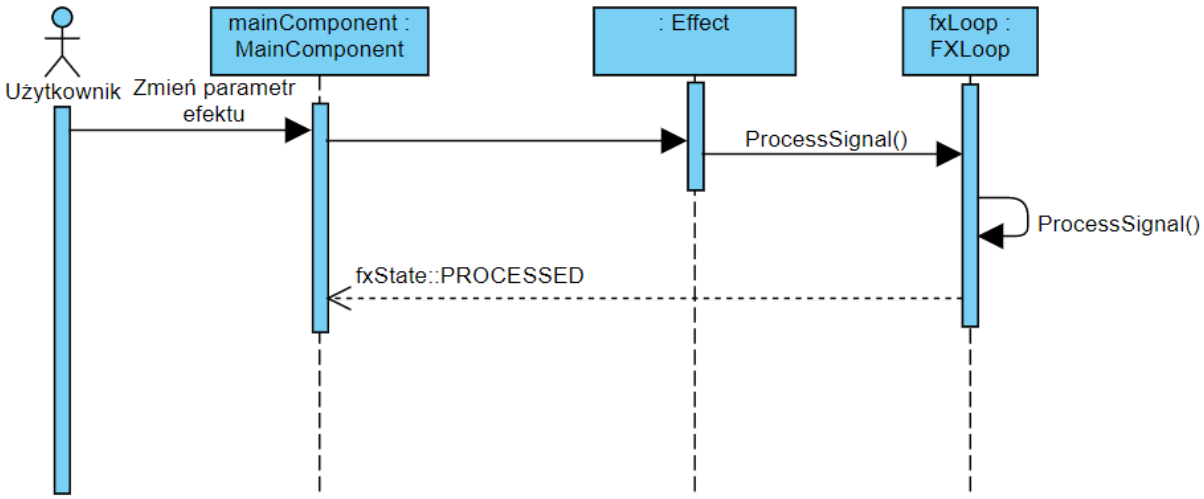


4.4. Diagramy sekwencji

Dodawanie nowych efektów



Zmiana parametrów efektów



5. Opis realizacji

Program został napisany w języku C++ w środowisku Visual Studio 2017, interfejs graficzny został wykonany za pomocą biblitoeki JUCE. Biblioteka ta została również użyta do odczytu i odtwarzania plików dźwiękowych.

Realizacja poszczególnych funkcjonalności programu:

- **Odczyt pliku** – metoda openFile() klasy MainComponent jest odpowiedzialna za odczyt plików.
- **Odtwarzanie pliku w wersji oryginalnej lub przetworzonej** – obiekt klasy Bar implementuje interfejs użytkownika umożliwiający odczyt, odtwarzanie plików i wymuszenie przetwarzania sygnału. Metody obsługi klawiszów są zaimplementowane w konstruktorze klasy za pomocą wyrażeń lambda.
- **Przechowywanie informacji o stanie i kolejności aktywnych efektów** – klasa FXLoop przechowuje wektor wskaźników na obiekty dziedziczące po abstrakcyjnej klasie Effect. Wektor jest domyślnie wypełniony wskaźnikami na obiekty EmptyEffect. Klasa FXLoop posiada metodę setEffect(index, name) umożliwiającą dodanie nowego efektu.
- **Dodawanie nowych efektów** – obiekty klasy EmptyEffect umożliwiają użytkownikowi dodanie efektów za pomocą przycisków – wywołują metodę setEffect klasy FXLoop.
- **Realizacja efektów** – metoda processSignal() klasy FXLoop przekazuje do kolejnych efektów wskaźnik na bufor AudioSampleBuffer w celu przetworzenia sygnału. Przetwarznie próbek na przykładzie implementacji w efekcie Overdive:

```
void Overdrive::processSignal(AudioSampleBuffer * processedBuffer)
{
    auto numChannels = processedBuffer->getNumChannels();
    auto numSamples = processedBuffer->getNumSamples();
    auto gain = 1.0f + gainKnob.getValue()*GAIN_FACTOR;
    auto distLevel = blendKnob.getValue();
    auto cleanLevel = 1 - distLevel;
    auto threshold = threshKnob.getValue()*THRESHOLD_FACTOR;

    for (auto channel = 0; channel < numChannels; ++channel) {
        for (auto sample = 0; sample < numSamples; ++sample) {
            float samp = gain * processedBuffer->getSample(channel, sample);
            if (samp > threshold) {
                samp = threshold;
            }
            //else if (samp < -threshold) {
            //  samp = -threshold;
            //}
            processedBuffer->setSample(channel,sample , samp*distLevel + (processedBuffer->getSample(channel, sample))*cleanLevel);
        }
    }
}
```

Istotne wartości są zapisywane w zmiennych przed głowną pętlą w celu przyśpieszenia działania kodu. Pętla przetwarzająca odczytuje wartości z bufora próbek, realizuje algorytm a następnie stare wartości próbek są zastępowane nowymi.

- **Zmiana parametrów efektów** – dowolna zmiana wprowadzona przez użytkownika wywołuje metodę processSignal() klasy FXLoop powodując ponowne przetworzenie wektora próbek kolejno przez wszystkie efekty.
- **Zmiana kolenności efektów** – każdy z efektów posiada obiekt removeButton klasy Button umożliwiający zastępienie tego efektu obiektem klasy EmptyEffect, który z kolei umożliwa zastąpienie go dowolnym innym efektem.

6. Opis wykonanych testów

Do testów zostałwykorzystany komputer osobisty z procesorem intel i5 ósmej genreacji, pamiecią operacyjną 8Gb i systemem operacyjnym Windows 10.

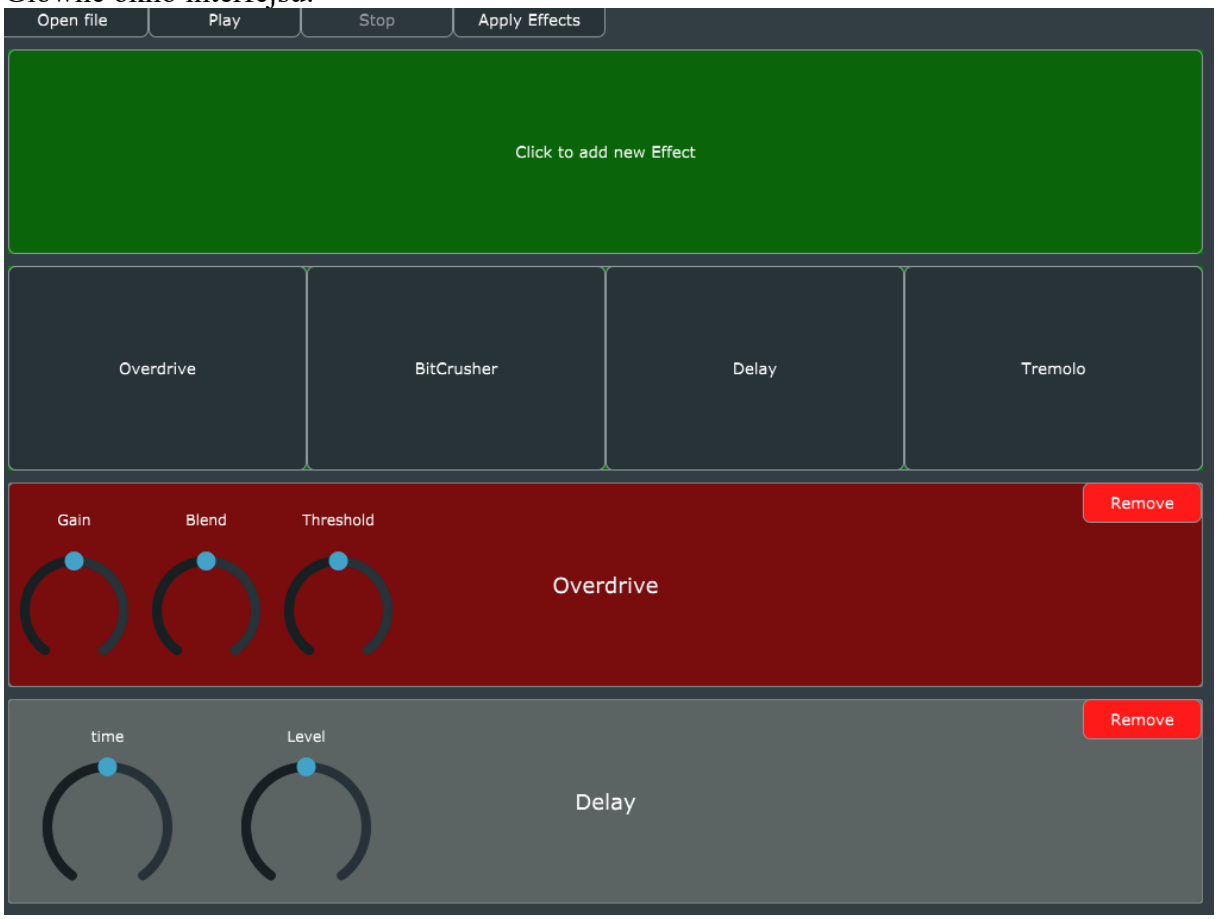
Działanie programu zostało sprawdzone dla plików .WAV o długości ponizej jednej minuty. Dla plików o tym rozmiarze czas realizacji algorytmów przetwarzających nie wpływa znacząco na komfort obsługi aplikacji.

Dotychczas nie zostały znalezione żadne błędy w działaniu interfejsu użytkownika.

W przyszłości planuje się sprawdzić działanie na systemach operacyjnych innych niż Windows 10 64-bit.

7. Podręcznik użytkownika

Program jest obsługiwany wyłącznie za pomocą myszki.
Główne okno interfejsu:



Instruckja obsługi:

- otwieranie plików – naciśnij przycisk Open File i wybierz ścieżkę pliku. Program działa prawidłowo tylko z plikami w formacie .WAV.
- odtwarzanie nagrania – użyj przycisku Apply Effect aby wybrać, czy chcesz zastosować efekty. Użyj przycisków Play/Stop do odtworzenia lub zatrzymania dźwięku.
- dodawanie efektów – naciśnij w polu opisanym „Click to add new effect” a następnie wybierz rodzaj efektu.
- zmiana parametrów efektów – użyj suwaków aby zmienić działanie efektów. Wszystkie suwaki działają liniowo z wyjątkiem „Bit Resolution” w efekcie Bit Crusher.