Video Player with effects

Autor: Neagu Fabian-Florin, 331AB

An: III

Cuprins:

1) Introducere

Scopul acestui proiect este de a replica un Player Video utilizat pentru vizionarea fisierelor video si audio. Pentru imbunatatirea calitatii experientei utilizatorului, acest Video Player pune la dispozitie o serie de efecte aplicate sunetului redat, oferind o coloana sonora imbunatatita special pentru redarea concertelor sau a filmelor.

Aplicatia propriu zisa pune la dispozitia utilizatorului o serie de functionalitati de baza precum: redarea, intreruperea, ajustarea volumului sau oprirea clipului redat, precum si posibilitatea redarii continutului cu viteze de redare diferite(Slow / Fast Motion). De asemenea, o caracteristica importanta a acestei aplicatii este reprezentata de implementarea unui egalizator (EQ – Equalizer) ce ofera posibilitatea modificarii manuale sau automate a benzilor de frecventa ale sunetului redat cu scopul de a imbunatati calitatea redarii continutului respectiv.

Pe langa punerea la dispozitie a utilizatorului posibilitatea redarii clasice continutului video sau audio, aplicatia are ca scop principal sporirea calitatii de redare prin implementarea unei palete variate de efecte audio ce sunt implementate cu ajutorul unui Equalizer.

Obiectivele impuse sunt:

- Crearea functionalitatilor de baza ale unui Player Video
- Realizarea unui Egalizator pentru modificarea frecventelor sunetelor
- Implementarea unei interfete grafice adecvate

2) Prezentarea suportului tehnic

Pentru implementarea acestei aplicatii am ales limbajul de programare Java deoarece consider ca cunostintele mele avansate din acest domeniu, combinate cu bibliotecile puse la dispozitie de acest mediu de dezvoltare, imi ofera o baza solida pentru contructia unei astfel de aplicatii.

Mediul de dezvoltare utilizat este reprezentat de Eclipse IDE, acesta oferind un mediu de programare controlat de fundatia Eclipse ce pune la dispozitie utilizatorului o serie de functionalitati ce ajuta la crearea proiectelor complexe. Libraria principala alesa pentru acest proiect este JavaFx pentru ca ofera un environment rapid si robust, si de asemenea are foarte multe functionalitati ce vor ajuta pe parcurs la implementarea atat a partii de logica din spatele programului in sine(BackEnd), cat si a interfetei grafice (FrontEnd). Deși bibliotecile mai vechi precum Java AWT și SWING au o mulțime de funcționalități, JavaFx reprezinta succesorul acestora si constituie o varianta imbunatatita de realizarea a interfetei grafice.[1]

De asemenea, dezvolatare interfetei grafice a aplicatiei a necesitat folosirea unui program specializat numit SceneBuider, acesta fiind un utilitar destinat dezvoltării de interfețe grafice pentru aplicații JavaFX, într-un mod vizual, generând fișierele FXML corespunzătoare prelucrate în cadrul aplicațiilor. Mai departe, acest fisier FXML este combinat cu proiectul Java prin legarea interfetei de utilizare la logica aplicatiei. In plus,

pentru realizarea partii de BackEnd am folosit o clasa Controller ce a fost asociata fiserului FXML cu scopul de a controla functionalitatile interfetei grafice.(ex: butoane, slidere, etc.).[7]

O alta caracteristica importanta este compatibilitatea aplicatiei cu diversele extensii alea fisierelor video / audio, programul dezvoltat acceptand doar rularea fisierelor de tip: MP3, MP4, WAV, FLV, MPEG.

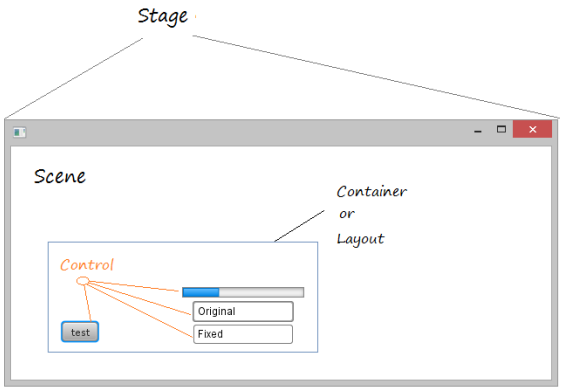
3) Prezentarea tehnica a etapei de implementare

Implementarea aplicatiei a debutat prin crearea unui proiect Java in mediul de dezvoltare Eclipse IDE, urmat de asocierea programului principal cu un fisier FXML care ulterior a fost modificat in mod vizual prin intermediul programului SceneBuilder. De asemenea, odata creata interfata grafica, pentru realizarea partii de BackEnd s-a asociat fisierului FXML un un Controller ce reprezinta o clasa ce este responsabila de implementarea functionalitatilor din spatele partii grafice expusa prin intermediul fisierului FXML.

3.1 JavaFx

Aplicatia dezvoltata se bazeaza in principal pe clasa Application (javafx.application.Application) ce pune la dispozitie un framework utilizat pentru crearea aplicatiilor JavaFx. Astfel, clasa principala, Main, extinde clasa Application si realizeaza supradefinirea (Override) metodei abstracte start() ce are rolul de a porni aplicatia in sine. [1]

Functia start primeste ca parametru un obiect de tip Stage, acesta fiind un container(o fereastra) ce reprezinta suportul pe care se vor adauga elementele vizuale ale interfetei grafice. Mai departe, in interiorul acestei ferestre am inserat o scena (JavaFx Scene) care reprezinta continutul ce se afiseaza pe ferestra (Stage-ul) transmisa ca parametru functiei. De mentionat este faptul ca acesta scena ce se insereaza pe fereastra principala are ca radacina fisierul FXML in care se insereaza vizual toate elementele cu ajutorul constructorului de scene, astfel ca fereastra generata la rulare va contine exact elementele adugate in fisierul FXML, elemente ce sunt controlate de clasa VideoController.[7]

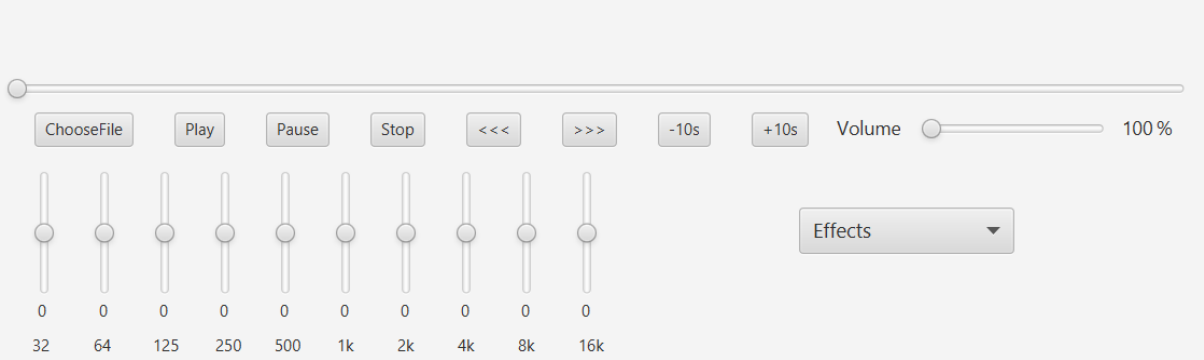


3.2 FXML & SceneBuilder

Pentru realizarea interfetei grafice in cadrul JavaFx am folosit un fisier FXML pe care l-am modificat cu ajutorul programului SceneBuilder, fisier caruia i-am atasat clasa VideoController ce are rolul de a controla actiunile ce se realizeaza odata cu interactiunea dintre utilizator si interfata grafica(ex: apasarea butaonului de play).[5]

FXML este limbajul bazat pe XML, folosit pentru construirea declarativă a interfeței aplicațiilor JavaFX. Avand oportunitatea de a lucra atat cu Java Swing, cat si cu JavaFx, am apreciat faptul ca utilizarea fisierelor de tip FXML realizeaza o separare mult mai buna a straturilor de aplicatie, realizandu-se o delimitare mult mai buna intre partae de FrontEnd si cea de BackEnd.[6]

De asemenea, cu ajutorul contructorului de scene am realizata integrarea in pagina a celui mai important element din interfata grafica, si anume, a elementului de tip MediaView ce constituie fereastra pe care va rula propriu zis coninutul video selectat. De asemenea, s-au inserat diverse butoane pentru realizarea implementarea functionalitatilor precum: redare, stop, pauza, reset, redare accelerata, redare incetinita, derulare +10 secunde, derulare -10 secunde. In plus, am adaugat multiple indicatoare de tip slider pentru afisarea si modificarea volumului, timpului de redare curent, dar, mai ales pentru modificare parametrilor Egalizatorului (frecventele de redare ale sunetului). Nu in ultimul rand, s-a adaugat un meniu pentru selectarea efectului aplicat coloanei sonore: Movie, Concert, Rock, Jazz, Pop.[8]



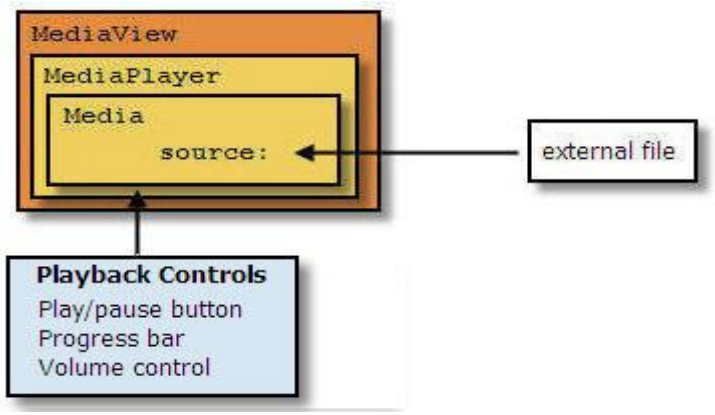
3.3 Media Player

Chiar daca partea cea mai interesanta este la capitolul clasei VideoController, aceasta avand rolul de implementa functionalitatile din partea de BackEnd, inainte de a trece la aceasta parte este necesar sa clarificam anumite aspecte legate de notiuni precum Media, MediaPlayer si MediaView.[2]

Pentru redare unui continut media a fost necesara crearea unui obiect de tip Media prin intermediul unui contructor caruia i-am transmis ca parametru un obiect de tip String ce

reprezinta calea catre fisierul media dorit a fi redat. Aceasta instanta are rolul de a retine metadatele fisierului selectat.

Mai departe, am creat o clasa de tip MediaPlayer pe baza clasei Media, primind de la aceasta metadatele respective. Importanta acestei clase este aceea ca prin intermediul ei se poate controla fluxul media, mai exact, se poate reda, opri, intrerupe sau realiza orice fel de comanda asupra fluxului media. Totusi, acest MediaPlayer nu contine niciun fel de suport vizual, astfel ca am realizat legatura dintre MediaPlayer si MediaView-ul definit prima data in fiserul FXML petntru a realiza expunerea vizuala a continutului selectat.[3]



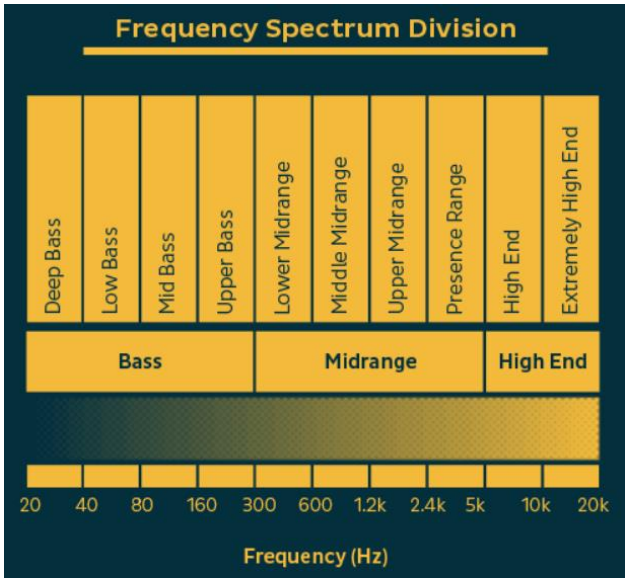
3.4 Equalizer

Egalizarea este o tehnică de procesare a semnalelor audio și reprezintă procesul de reglare a diferitelor canale de sunet. Cu alte cuvinte, egalizarea se referă la operația de netezire a diferitelor părți ale spectrului de frecvențe care alcătuiesc un semnal audio. [4]

Motivul din spatele integrari unui Egalizator in componenta Playerului Video este acela de a crea crea efecte audio particulare cu scopul de a oferi o calitate audio sporita in momentul vizionarii filmelor sau concertelor. In acest sens aplicatia aduce implementarea unor efcte audio specializate precum: Movie, Concert, Rock, Pop si Dance. Fiecare dintre acestea reprezinta o schema predefinita si particulara de setare a egalizatorului ce are rolul de modifica sunetul formatului redat astfel incat sa se plieze de nevoile si preferintele utilizatorilor. Pe langa aceasta caracteristica de efecte predefinite, aplicati ofera utilizatorului optiunea de a modifica manual canalele audio dupa bunul plac pentru a crea scheme partiulare de reglare ale frecventelor ce compun sunetul fisierului redat.

Gama de frecvențe ale egalizatorului:[9]

- Gama joasă: 20Hz – 250Hz
- Interval mediu: 250 Hz – 4 kHz
- Gamă înaltă: 4 kHz – 20 kHz



3.4 Video Controller

Pentru implementarea functionalitatilor elementelor definite in fisierul FXML este necesara crearea unuei clase ce joaca rolul uni Controller care, dupa cum ii spune si numele, controleaza cea ce se intampla „sub capota”. Cu alte cuvinte, aceasta clasa se ocupa de implementarea functionalitatilor elementelor grafice si se leaga de fisierul FXML pentru a avea controlul asupra interfetei grafice.[7]

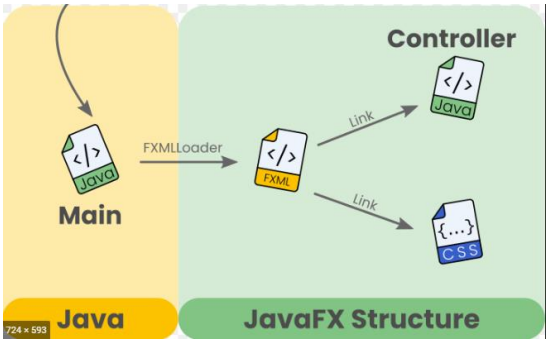
Pe langa ecele 3 elemente expuse mai sus: Media, MediaPlayer si MediaView, aceasta in aceasta clasa se mai defineste un obiect de tip FileChooser in interiorul unei metode ce are rolul de a set continutul media ce urmeaza a fi redat pe ecran. Acest element ofera o interfata grafica predefinita ce ajuta la navigarea in sistem cu scopul de a se selecta fisierul dorit a fi redat.

De asememea, in aceasta clasa se implementeaza functii precum: PlayMethod, PauseMethod, SlowRateMethod, etc, functii ce vor fi atasate butoanelor, repectiv sliderelor din partea grafica pentru implemnetare functionalitatii acestora. Aceste metode create se folosesc de functiile predefinite din clasa MediaPlayer, precum: mediaPlayer.play(), mediaPlayer.pause() sau mediaPlayer.setRate(1).[7]

Pentru implementare egalizatorului am creat un obiect de tip AudioEqualizer, obiect ce are capacitatea de a controla setarile de egalizare audio pentru un obiect de tip MediaPlayer. Aceste obiect de tip AudioEqualizer contine o lista (ObservableList) ce are elemente de tip EqualizerBand, aceasta reprezentand o clasa ce pune la dispozitie controlul pentru fiecare banda frecventiala in parte. Astfel, pentru a se putea modifica anumite canale de sunet am modifcat gain-ul elementelor din lista de EqualizerBands. Cu alte cuvinte, am modifcat benzile de frecventa ce compun spectrul audio al fisierului dat.

Implementarea fiecarui efect implica realizarea unor metode specifice in care nu facem altceva decat sa modificam valorile de la anumite frecvente(anumite benzi) in scopul

de a obtine imbunatatirea audio dorita. De asemenea, odata modificate valorile canalelor, a trebuit sa asigur sincronizarea valorilor acestora cu valorile indicate de sliderele respective din dreptul fiecarei frecvente.[7]



4) Mod de utilizare

Dupa cum am mentionat inca de la inceput, scopul acestei aplicatii este de a implementa un Player Video ce ofera utilizatorilor o modalitate facila si usor de utilizat pentru redarea continutului video si audio dorit. Procesul de utilizare al aplicatiei dezvoltate este usor de inteles si de utilizat, punandu-se la dispozitie o serie de functionalitati generaliste, dar si specifice.

4.1 Alegere fisier

In momentul in care aplicatia se porneste, se afiseaza pe ecran interfata grafica dezvoltata si detaliata la punctele anterioare si se asteapta alegerea fisierului dorit a fi redat. Acest lucru se realizeaza prin apasarea butonului „Choose File” care, odata apasat, pune la dispozitia utilizatorului o fereastra noua in care acesta poate naviga si selecta fisierul dorit. In momentul in care fisierul este selectat si deschis, fereastra deschisa de FileChooser se va inchide automat, iar videoclipul selectat se va reda pe fereastra de MediaView, asteptandu-se comenzi suplimentare.

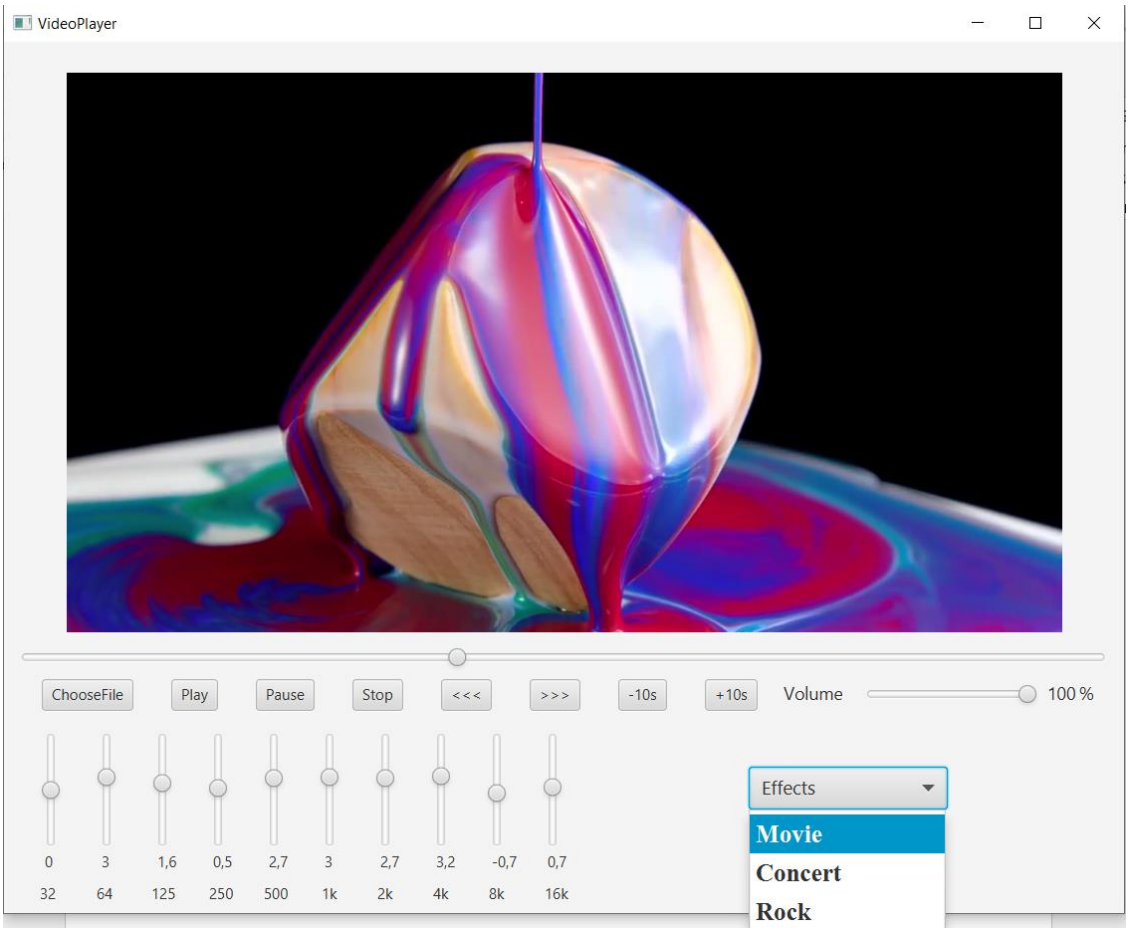
4.2 Executare comenzi uzuale

Aceasta sectiune se refera la modul de utilizare al comenzilor uzuale ce pot fi folosite in cadrul unui VideoPlayer, precum: redare, pauza, redare accelerata/incetinita, salt 10 secunde inainte/inapoi, ajustare volum, etc. Aceste comenzi se pot utiliza folosind butoanele, dar si sliderele puse la dispozitie in interfata grafica.

4.3 Aplicarea efecte audio specifice

Functionalitatile cele mai interesante ale acetui proiect sunt cele legate de procesarea audio implementara. Mai exact utilizatorul, prin intermediul unui meniu, poate alege ce efect sa se aplice coloanei sonore asociate fisierului video/audio. In plus, prin implementarea egalizatorului, se permite realizarea unui efect individual prin configurarea manuala de catre utilizator a valorilor canalelor audio puse la dispozitie. In acest fel, utilizatorul isi poate

configura dupa bunul plac frecventele audio ale fisierului dorit a fi redat. Aceasta functionalitate este importanta deoarece s-au implementat efecte precum: Movie, Concert, Jazz, Pop si Rock, necesare pentru imbunatatirea calitatii audio oferite utilizatorilor.



5) Concluzie

In concluzie, aplicatia reuseste sa implementeze cu succes atat caracteristicile de baza specifice oricarui Video Player, cat si o serie de functionalitati specifice prelucrarii audio menite sa imbunatateasca calitatea de redare a utilizatorului. Se poate afirma faptul ca obiectivele acetui proiect au fost indeplinite, intrucat utilizatorul poate folosi eficient si usor acest program pentru vizionarea calitativa a formatelor audio si video.

Pe de alta parte, limitarile aplicatiei dezvoltate sunt legate in primul rand de compatibilitatea relativ restransa a fisierelor acceptate de biblioteca JavaFx, aceasta fiind permitand doar redarea fisierelor de tip MP3, MP4, WAV, FLV, MPEG. De asemenea biblioteca JavaFx dispune de un numar de posbilitati restranse de aplicare a efectelor video.

Din punct de vedere al imbunatatirilor ce pot fi atasate acestui proiect se poate afirma faptul ca pornind de la baza constituita de program se pot realiza cateva dezvoltari ale acestuia in mai multe directii. O imbunatatire pe care ma voi concentra in viitor este

realizarea nu doar a efectelor audio, cat si a efectelor video, precum efecte alb-negru, culori negative, etc. In plus, se poate modifica selectorul de fisiere pentru a se putea alege fisiere dorite a fi redate inclusiv din spatiul de stocare Cloud. De asemenea se poate exinde acest proiect pentru sistemul de operare Android, putand fi portat ca aplicatie mobila.

Personal, acest proiect mi-a atras foarte mult curiozitatea si admiratia pentru domeniul aplicatiilor multimedia, avand oportunitatea de a descoperi complexitatea, dar si multitudinea lucrurilor ce pot fi realizate prin intermediul prelucrarilor audio si video. Am avut sansa de a utiliza pentru prima data biblioteca JavaFx si pot afirma faptul ca aceasta este extrem de utila si considerabil imbunatatita fata de predecesorul „Java Swing”. De asemenea, am fost foarte entuziasmat sa descopar cat de multe lucruri se pot realiza folosind interfetele grafice dar si domeniul aplicatiilor multimedia.[1]

6) Bibliografie

[1] JavaFx
[2] Media
[3] MediaPlayer
[4] Equalizer
[5] SceneBuilder
[6] FXML
[7] VideoController
[8] MediaEffects
[9] Canale Audio