Projet d'Objet et développement d'applications

Lecteur Audio mp3

GUIADEM KAPTUE Michelle – Thierno Saïdou

Groupe 502C

Année académique 2016–2017

Table des matières

Introduction

Le thème de notre projet est l'élaboration d'un lecteur mp3 audio. L'utilisation des designs patterns nous serve à la description d'un arrangement récurrent de rôles et d'actions joués par des modules de notre logiciel. Nous nous sommes inspirés des modèles de lecteurs connus tels que vlc, media player juste en leur version audio. En rapport avec ces derniers, les fonctionnalités de notre lecteur doivent être la lecture, la mise en pause de la lecture courante, l'arrêt, passer à la lecture suivante ou précédente, augmentation du volume. Ainsi, nous avons utilisés la librairie SFML (Simple and Fast Multimedia Library) ainsi que l' un de ses compléments TGUI (Texus' Graphical User Interface) pour gérer l'interface graphique.

1. Mode de fonctionnement

Après ouverture, la première fenêtre qui apparaît est celle de la GUI (interface utilisateur) principale. L'utilisateur peut ensuite choisir de lire un fichier audio ou image. Une deuxième fenêtre apparaît après le choix (entre audio ou image). Si l'utilisateur a cliqué sur audio, la fenêtre qui apparaît, est inscrit sur elle les actions possibles **lecture, pause et stop**. S'il a choisit plutôt image,

la fenêtre qui apparaît, est inscrit sur elle **suivante et précédente**. Les fichiers chargés proviennent du dossier “ressources/musique" pour l'audio et “ressources/images" pour l'image .

Les actions possibles pour l'audio sont les fonctionnalités standards de tout lecteur multimédia (même si nous ne sommes parvenus qu'à faire implémenter que quelques unes) :

Lecture : elle s’enclenche automatiquement à l'ouverture d'un fichier audio.

Pause : l’utilisateur peut choisir de mettre la musique en pause pendant la lecture s’il souhaite la relancer plu-tard.

Arrêt : l'utilisateur peut également choisir d'arrêter la musique en cours de lecture s'il ne souhaite plus la relancer.

Les actions possibles pour afficher une image :

Suivante : après le choix de la lecture d'une image par l'utilisateur, il peut faire afficher l'image suivante.

Précédente : une fois étant sur une image l'utilisateur peut rentrer à l'image précédente.

2. Description des classes

Cette section se propose de décrire l’architecture de notre programme c++ en listant et décrivant quelques principales classes. Le lecteur audio mp3 est ainsi constitué de trois dossiers contenant chacun les classes .hpp et .cpp des trois designs patterns implémentés dans notre projet et constitué également de la classe main.cpp tous présents dans le dossier “src".

2.1 main

La classe main est celle dans la quelle est créer les interface audio et image nécessaire à l'affichage sur l'écran.

```
int main()
{
    XInitThreads();
    //création de l'objet Gui
    sf::RenderWindow window(sf::VideoMode(300, 100), "Lecteur Audio mp3");
    tgui::Gui gui(window);
    tgui::Callback callback;

    window.setVerticalSyncEnabled(true);

    if (gui.setGlobalFont("src/fonts/DejaVuSans.ttf") == false)
        return 1;

    tgui::Picture::Ptr picture(gui);
    picture->load("src/fonts/fond-blanc.png");
    // création d'un bouton avec widget
    tgui::Button::Ptr buttonMusic(gui);
    buttonMusic->load(THEME_CONFIG_FILE);
    buttonMusic->setPosition(100, 0);
    buttonMusic->setText("Musique");
    buttonMusic->setCallbackId(1);
    buttonMusic->bindCallback(tgui::Button::LeftMouseClicked);
    buttonMusic->setSize(100, 100);
```

2.2 Audio

Classe qui met en relation les différents états que peuvent prendre un fichier en lecture. La méthode “tourne" est celle dans laquelle le changement d'état s'effectue selon l'état courant dans lequel le lecteur se trouve pour un fichier audio.

```
//Constructeur de la factory de l'interface audio
Audio::Audio(AudioInterfaceFactory* audioInterFact)
{
    Buttons* bouton = new Buttons();
    FormatSmall* fs = new FormatSmall();
    _inter = audioInterFact->creerInterface(bouton,fs);

    EtatArret eA(this);
    EtatLecture eL(this);
    EtatPause eP(this);

    _eC = &(eA); //pourquoi passage par référence;
    _eL = eL;
    _eP = eP;
    _eA = eA;
}
```

2.3 Image

Classe qui met en relation les différents formats que pouvant être appliquer à une image.

3. Patterns

State et abstract factory sont les patterns utilisés dans notre projet. Ils sont respectivement patterns de comportement et de création. Au début nous avions prévu de faire le pattern de structure “Decorator" mais nos idées n’étant pas bien mise en place pour celui-ci nous avons préférés ne pas nous attarder sur lui. Nous n'avons donc pas pu atteindre l'objectif qui était de faire son projet avec l'utilisation de trois patterns de deux familles différentes.

3.1 State

Tout naturellement nous avons vu l'implémentation du pattern state dans la réalisation de notre lecteur audio car comme dit précédemment nous nous sommes basés sur les fonctionnalités standards qui sont la lecture, la pause et l'arrêt qui sont les états que peuvent prendre un fichier audio.

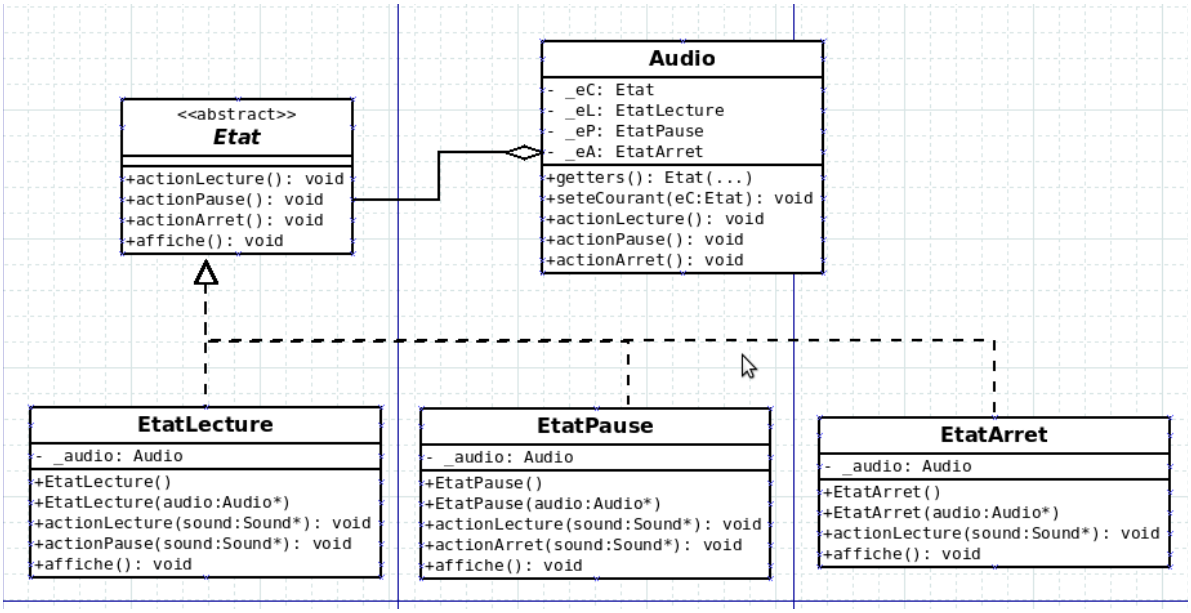

Figure1 : pattern state

3.2 Abstract Factory

La création d'une interface graphique a été difficile à mettre en œuvre. Car nous ne savons premièrement pas utiliser la librairie SFML et son complément TGui. Et en plus nous devrions les mettre en relation avec le pattern abstract factory pour créer une interface. Ce pattern permet dans le cadre de notre projet de créer les **boutons** et les **formats.** Ainsi nous avions un fabrique de boutons et une fabrique de format.

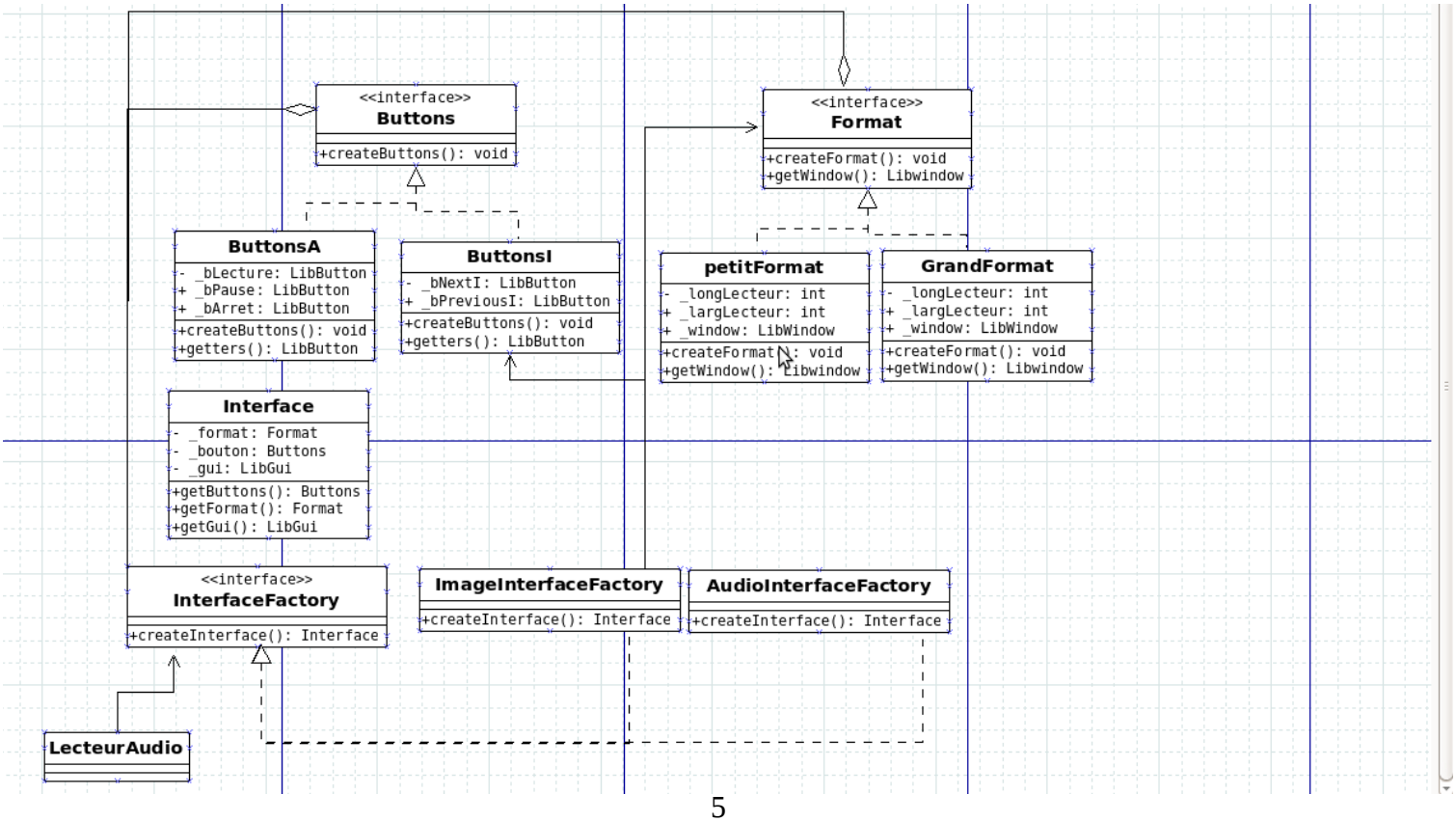

Figure2 : pattern abstract factory

4. Difficultés rencontrés

Nous avons eut des problèmes durant la réalisation, notamment dans l'apprentissage de l'utilisation de la librairie SFML ??? et son complément TGUI afin de les mettre en relation avec nos classes. **Le temps** ne nous a pas permis de mettre en œuvre toutes les fonctionnalités cités en début de rapport ce qui et très décevant car même personnellement nous ne pourrons pas utiliser ce lecteur puisque ses fonctionnalités sont moindre.

Conclusion et améliorations, développements futurs

Conclusion

Pour conclure le projet, nous avons apprécié suivre le cours "objet et développement d'application", et le projet nous a apporté les connaissances pratiques nécessaires pour la mise en place des notions théoriques vues durant le cours. Nous avons également apprécié apprendre et voir une autre face de l'informatique, un peu plus visuelle que ce que nous avons vu jusqu'à maintenant. Pour ce qui est du projet, nous sommes content du résultat obtenu. La création d'un logiciel de lecture de musique est une expérience intéressante pour nous car nous ne l'avons jamais fait auparavant. Encore plus intéressante avec les design patterns.

Améliorations, développements futurs

Puisque la durée du projet est relativement courte, il a été difficile de créer un logiciel élaboré. Cependant, avec plus de temps, il aurait été possible d'ajouter plus d'outils au lecteur audio, telles que :

- La lecture de différents formats de fichiers audio (.wav, .ogg, midi...etc.) ;

- La gestion du volume par l'interface principale ;

- L'utilisation de tous styles de musique et non de styles prédéfinis ;

- Permettre l'avance rapide également.

Annexes

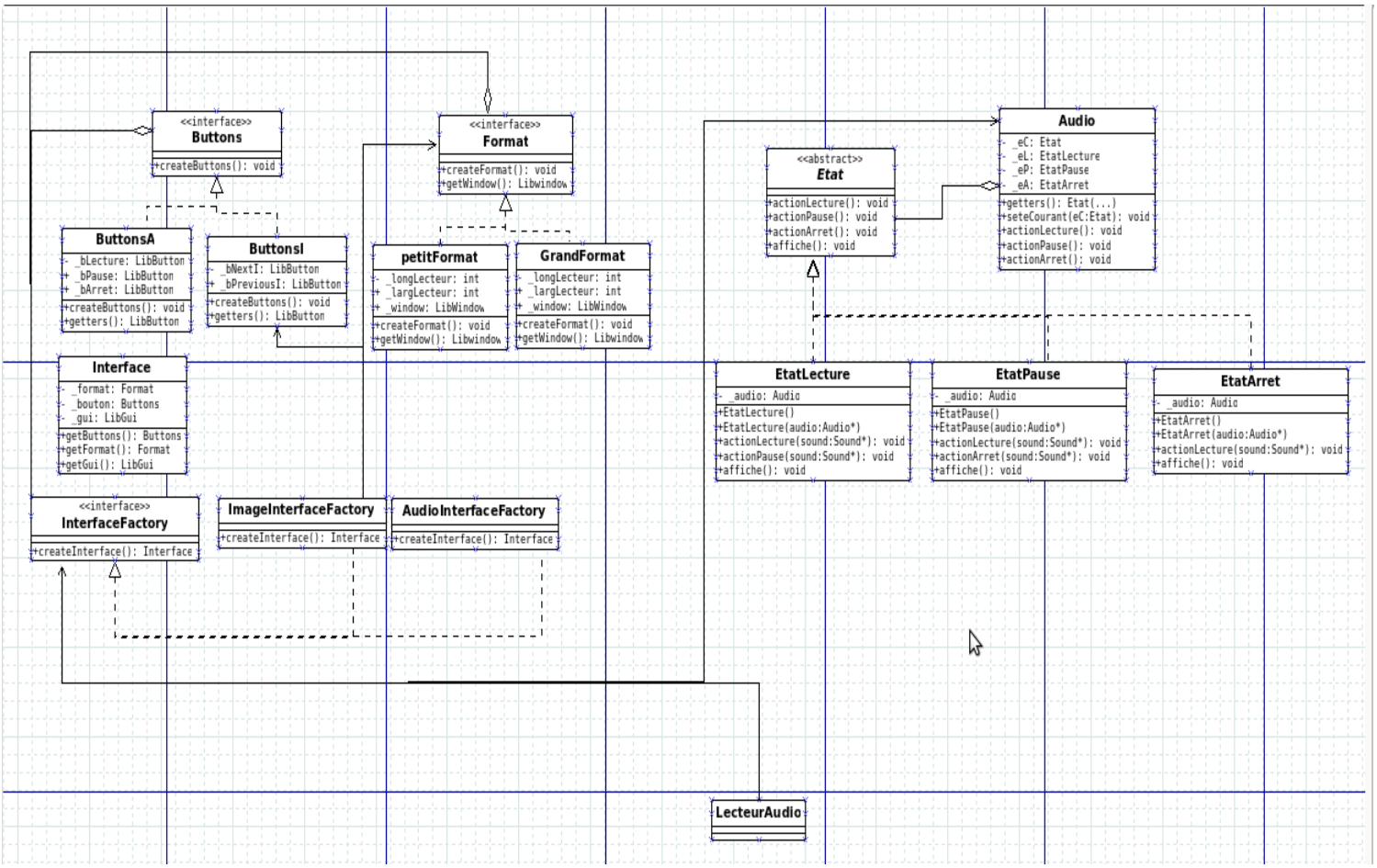

Figure 3 : UML du projet