



CLiMaX
Advanced Audio Shaker

Julien-Pierre AVÉROUS
Guillaume CALAS
Patrice DE SAINT STEBAN
Fabien DEBUIR



Table des matières

1	Le groupe : origine et constitution	1
1.1	Création du groupe	1
1.2	Constitution du bureau	1
1.3	Les membres du groupe	1
1.3.1	Julien-Pierre	1
1.3.2	Guillaume	2
1.3.3	Patrice	2
1.3.4	Fabien	2
1.4	Intérêt du projet	2
2	Objet de l'étude	4
2.1	Idée du Projet	4
2.2	Présentation du projet	4
3	Définition du projet	5
3.1	Découpage	5
3.2	Programme principal	6
3.3	Modularité	6
3.4	Répartition	8
3.4.1	1 ^{re} soutenance	8
3.4.2	2 ^e soutenance	8
3.4.3	3 ^e soutenance	9
3.4.4	Soutenance finale	9
4	Moyens mis en œuvre	10
4.1	Les logiciels	10
4.2	Au niveau humain	10
4.3	Le matériel	11
5	Conclusion	12

Introduction

CliMaX est un logiciel de mixage Mp3 évolué. Il sera multi-plateforme et composé d'une multitude de modules pour intégrer toutes les fonctions que nous jugerons nécessaires. Il sera prioritairement développé sous Unix mais sera aussi porté sous Mac et peut être sous Windows.

Ce logiciel sera modulaire, portable, avec un coût de développement peu important (utilisation de l'Open Source) et développé dans un temps restreint (8 mois).

Ce cahier des charges présentera le groupe, les prévisions des différentes phases du projet, ainsi que le matériel dont nous aurons besoin.

Chapitre 1

Le groupe : origine et constitution

1.1 Création du groupe

Cette année nous avons constitué une équipe d'ex SUP-B2 : Patrice DE SAINT STEBAN et Julien-Pierre AVÉROUS (C1) d'une part, et Fabien DEBUIRE et Guillaume CALAS (A1) de l'autre.

1.2 Constitution du bureau

Chef de projet : nous avons décidé que nous ferions une rotation de présidence tout au long de l'année, à chaque soutenance, de façon à ce que chacun puisse avoir une expérience de (pseudo-)management avant d'intégrer le cycle d'ingénierie.

L'ordre de passage est le suivant : AVÉROUS, CALAS, DEBUIRE et DE SAINT STEBAN.

1.3 Les membres du groupe

1.3.1 Julien-Pierre

J'ai 20 ans, Aquitain, j'ai commencé à développer en BASIC en Primaire (ordinateur-clavier), en RealBasic (équivalent Mac du VisualBasic) au collège et je développe en C/C++ depuis ma seconde au lycée. J'ai fait pas mal de projets tout seul pour des commandes, ou des projets libres avec des groupes d'utilisateurs sur un forum Mac où je suis modérateur de la section programmation (Objective-C, RealBasic, AppleScript, et autres langages). Je développe couramment de petits "Freeware" (en Realbasic ou Objective-C) et des "Plug-In" (en C++) ou des bouts

de code "utilitaires". Je connais bien PHP et MySQL (mon site est développé avec).

1.3.2 Guillaume

Âgé d'une vingtaine d'année, je suis originaire du Sud de la France (34). Très tôt passionné par le concept même de l'informatique, j'ai commencé avant même de toucher à mon premier ordinateur, c'était il y a dix ans. J'ai touché à peu près à tous les domaines, j'ai commencé à coder en QBasic avant de m'intéresser à Flash. J'ai ensuite créé et géré un site (www.mi-x2.net) durant deux ans.

J'ai été plutôt éccœuré par les évènements qui se sont déroulés dans mon groupe de projet de l'année dernière et j'attends, pour celui de cette année, quelque chose de positif et de résolument professionnel. Sans quoi ce sera une pure perte de temps pour tout le monde.

1.3.3 Patrice

J'ai 19 ans, et j'habitais Chartres (28), mais la famille vient de déménager à ST MALO. J'aime beaucoup la musique, je fais du piano et de l'orgue depuis déjà un certain nombre d'années. J'ai commencé l'informatique en créant des sites internet, qui se sont développés. J'ai ensuite appris le PHP. Puis, je me suis un peu intéressé à d'autres langages. L'année dernière, j'ai développé un jeu en Delphi avec les règles du jeu de société Risk mais dans l'univers de Star Wars, en Delphi et Directdraw.

1.3.4 Fabien

Je viens de TOULOUSE et suis passionné par l'informatique depuis tout petit. Je suis aussi Disc-Jockey dans mes heures perdues (qui se font de plus en plus rare) Sinon dans un contexte plus "informatique" j'aime bien développer de gros projets à l'image de ceux proposés par l'école.

1.4 Intérêt du projet

Pour nous, c'est notre deuxième projet dans le cadre de l'EPITA. Ce projet sera réalisé en C++ qui est une version objet du C, le langage que nous étudions en cours et qui sera important l'année prochaine en première année d'ingénierie. Ce projet va aussi nous apporter algorithmiquement, la manière de traiter le son sur un ordinateur, et surtout l'utilisation d'algorithmes avancés.

Comme nous voulons faire un programme modulaire, ce projet va nous apporter beaucoup dans la programmation, car nous devons créer et respecter un certain nombre de règles pour créer les différents modules ainsi que le programme principal qui gèrera tout cela.

Comme le projet sera important et devra être réactif, et rapide, cela va nous apprendre à optimiser nos algorithmes au maximum.

Chapitre 2

Objet de l'étude

2.1 Idée du Projet

Lors de notre première réunion, nous ne sommes pas arrivés à nous mettre d'accord sur le projet en lui même.

Par ailleurs, nous avons mis en place une liste de projets potentiels. Les idées étaient les suivantes :

- Mixage MP3
- Retouche d'image
- OCR
- Management d'un serveur distant
- Compilateur de langage algo
- Dessin Vectorielle
- Logiciel de gestion de donnée
- Gestion d'emplois du temps

2.2 Présentation du projet

Liste des fonctions du logiciel :

- Lecture de fichier audio (Mp3, Wav, Ogg, ...)
- Bibliothèque de fichiers, avec moteur de recherche évolué
- Playlist
- Équaliseur
- Fonctions spéciales (écho, réverbération ...)

Chapitre 3

Définition du projet

3.1 Découpage

De par la nature très modulaire de notre projet, un découpage très net des tâches à faire est facilement envisageable. Voici le découpage du projet :

- **Programme principale** – Le programme principale du projet qui va se charger de charger en mémoire les plug-in, gérer la communication entre eux, de passer les événements, de gérer l’affichage, etc.
- **Moteur graphique** – C’est la partie du programme principale qui va permettre d’afficher les interfaces graphiques des plug-in qui définiront une interface.
- **SDK** – C’est l’ensemble protocole, fichier de définition et fonctions qui va permettre de créer un Plug-In pour notre application. C’est avec ceci que nous allons développer ensuite nos Plug-In et c’est avec cela que n’importe qui pourra en créer.
- **Plug-In** – L’ensemble des Plug-In à faire.
- **Documentation** – La documentation d’utilisation de notre logiciel et de l’utilisation du SDK permettant de créer un Plug-In.
- **Site web** – Le site web du projet.

3.2 Programme principal

L'application principale va être la base du projet. Malgré que ce soit l'élément central de notre projet, ce sera lui qui va être le plus rapide à écrire. En effet, il ne va se charger que de gérer un affichage graphique simple via un moteur graphique (QT ou OpenGL), de charger en mémoire les Plug-In présents dans le dossier adéquate, mettre a disposition des autres Plug-In les Plug-In chargés en mémoire, gérer la communication entre les Plug-In par système d'identifiant, gérer les évènements système (clavier, souris, ...), et enfin appeler les différentes fonctions des Plug-In (par exemple décodage de flux, etc.) dans la boucle événementielle de l'application.

3.3 Modularité

Le programme sera entièrement modulaire. Chaque module (ou Plug-In) aura une fonction propre dans l'application, comme par exemple la gestion de la bibliothèque, le décodage de MP3, l'afficheur, les effets audio, etc.

Chaque Plug-In va être codé par un SDK, qui sera, avec l'application principale, la première chose écrite. Un Plug-In devra dans son code source définir au moins 3 fonctions de base qu'utilisera directement l'application :

– **void clInit()** – Cette fonction sera appelé quand le plug-in sera chargé en mémoire. Dans cette fonction devra être appelé la fonction `void RegisterModule(clMod *module) clMod` étant une structure définit dans les "headers" du SDK qui contient les variables suivantes :

- `string identifiant` l'identifiant textuel unique du Plug-In.
- `int kind` l'entier définissant le type du Plug-In (voir plus loin)
- `string comment` le commentaire sur le Plug-In (crédits rapide)
- `string name` le nom du Plug-In
- `pict icon` l'icône, ou imagerie du Plug-In.

– **void clClear()** – Cette fonction sera appelé juste avant que le Plug-In soit libéré de la mémoire, pour permettre une libération des instances internes au Plug-In.

– **void clAbout()** – Cette fonction sera appelée quand l'utilisateur du logiciel demandera l'"A Propos" du Plug-In. Libre au développeur du Plug-In de faire ce qu'il veut dans cette fonction. Mais le SDK fournira tout de même une fonction

permettant d'afficher une fenêtre basique avec le texte passé en paramètre de la fonction.

Comme vu précédemment, chaque Plug-In lors de son enregistrement devra définir son type. Voici les types possibles de Plug-In envisageables :

– **Entrée=0** – Plug-In de type entrée. Se seront les plug-in chargés de décoder un stream binaire en un format directement exploitable (le format en question sera propre à notre application). Il est défini dans notre SDK de cette manière :

```
typedef unsigned char** clFlow
```

On va surtout envisager dans un premier temps comment entrer un fichier, mais l'entrée pourra être de type quelconque par exemple stream Internet, Micro, etc.

– **Sortie=1** – Plug-In de type sortie. Se seront les plug-in chargés d'émettre un `clFlow` sur une sortie quelconque. On va surtout envisager les haut-parleurs dans un premier temps, mais on peut imaginer du "broadcast", etc.

– **Filtre=2** – Plug-In de type filtre. Il prendra en paramètre un `clFlow` et devra retourner un `clFlow`. Ceci regroupe tous les Plug-In devant modifier en temps réel un flux musical, pour, par exemple, faire de l'écho, baisser le volume, etc.

– **Utilitaire=3** – Plug-In de type utilitaire. Plug-In chargés de mettre à disposition des autres Plug-In des utilitaires de tout ordre, comme par exemple de gestion mémoire de flux audio, etc.

– **Bibliothèque=4** – Plug-In de type bibliothèque. Ils seront chargés de gérer un ensemble de flux audio. Par exemples des fichiers, des adresses de stream Internet, etc.

– **Visualiseur=5** – Plug-In de type visualiseur. Ils seront chargés d'afficher un visuel en fonction d'un `clFlow`. Par exemple, des effets abstraits avec ligne se déformant sur l'amplitude des fréquences de la musique bien connus sur les logiciels de lecture audio, affichages des amplitudes de fréquence, affichages d'une ligne de musique, etc.

– **Commande=6** – Plug-In de type commande. Ils seront chargés d'implémenter par exemple les boutons **Play**, **Pause**, **Stop**, etc.

En plus des fonctions de base, communes à chaque Plug-In (vu en début de section), chaque type de Plug-In devra définir un certain nombre de fonctions. Voici quelques exemples :

Type entrée

- `clFlow read(unsigned int size)` devra renvoyer un `clFlow` contenant *size* octets de donnée audio décodé dans le flux interne géré par le Plug-In
- `int error()` renvoi un code d'erreur si une erreur c'est produite. L'ensemble des codes erreur possibles seront listés dans un "header" du SDK.
- `bool Ended()` renvoi *true* s'il n'y a plus rien à décoder (fin de fichier par exemple), *false* sinon.

Type visualiseur

- `void parametrage(clVisualiseurParametre parm)` sera appelé lorsqu'il est temps de paramétrer le visualiseur en fonction d'une structure pouvant par exemple contenir les résolutions et définitions de l'écran, l'utilisation maximal des ressources processeur, etc.
- `void rendus(clFlow flow)` sera appelé pour mettre à jour l'affichage en fonction d'un nouveau bloc de donnée `flow`.

3.4 Répartition

La répartition, comme le découpage est assez facilement réalisable par la nature modulaire du projet. Chacun peut coder un Plug-In sans attendre le Plug-In d'un autre, du moment que la structure générale de l'application est bien définie.

3.4.1 1^{re} soutenance

	Julien-Pierre	Guillaume	Fabien	Patrice
Programme principal	+	+	+	+
Moteur graphique			+	
Site Web	+	+	+	+

3.4.2 2^e soutenance

	Julien-Pierre	Guillaume	Fabien	Patrice
Moteur graphique		+	+	
SDK	+			+

3.4.3 3^e soutenance

	Julien-Pierre	Guillaume	Fabien	Patrice
Plug-In	+	+	+	+

3.4.4 Soutenance finale

	Julien-Pierre	Guillaume	Fabien	Patrice
Plug-In	+	+	+	+
Documentation	+	+	+	+
Installation	+	+	+	+

Chapitre 4

Moyens mis en œuvre

4.1 Les logiciels

- QT DESIGNER, GCC (gratuit)
- Librairie graphique : QT (et peut-être OPENGL) (gratuit)
- GIMP (gratuit)
- Logiciel audio

Pour le site :

- (PHPCODER WIN / SUBETHAEDIT MAC) (gratuit)
- (EASYPHP WIN / SUBETHAEDIT MAC) (gratuit)

Le coût total de ces logiciels est de 0,00 €...

4.2 Au niveau humain

Plusieurs décisions prises par le groupe permettront, nous l'espérons, de fonctionner de manière régulière. Tout d'abord, une réunion du groupe, effectuée chaque semaine (le vendredi soir) permettra de faire un bref résumé des avancées de chacun, cela permettra de travailler ensemble et de s'aider mutuellement et dans la bonne humeur. Un court rapport sera tapé, résumant ce qui a été dit et fait lors de ces séances, mais aussi toutes nos impressions. Ce rapport sera affiché sur notre site Web, permettant de suivre l'avancée du groupe. Cela se fera après la création du site.

De plus, un moyen a été mis en place pour pouvoir suivre l'avancée des travaux, car Fabien va mettre en place un serveur CVS qui permettra de toujours avoir nos sources à jour et de gérer les modifications simultanées du code source, ainsi que les anciennes versions.

Enfin, il y a bien entendu nos chers outils bien aimés, GAIM (Windows & Li-

nux), AIM (Mac & Windows), NetSoul (Mac & Windows), mail et le téléphone...

4.3 Le materiel

- Centrino 1,5 | GForce FX (DEBIAN *Sarge*)
- PowerBook Ti G4 1GHz 15" | 64 Mo Vram 512 Mo RAM (OsX)
- Centrino 1,73 GHz | ATI X600 128 MO | 512 Mo (DEBIAN *Sarge*)
- P-IV 3,2E GHz | FX5700-Le 128 Mo | 1024 Mo RAM (DEBIAN *Sarge*)
- Centrino 1,73GHz | 1024 Mo RAM (OPENBSD)

Chapitre 5

Conclusion

Voici désormais le début d'une grande aventure pour nous tous. Des habitudes nouvelles vont devoir être prises, telles que la régularité, l'obstination et la volonté de travailler en groupe, le temps qui nous est imparti excluant tout bonnement le plus petit retard. Déjà, nous avons pu constater que les idées de chacun pouvaient très fortement diverger selon les thèmes abordés. Il faut de plus prendre en compte les difficultés ou les facilités de chacun, ce qui peut être parfois profondément irritant. Il est certain que les échéances seront primordiales, mais savoir relever un membre de l'équipe lorsqu'il est en panne de confiance (et cela arrivera fatalement) est tout aussi crucial à la bonne tenue de notre projet. C'est pourquoi l'aspect humain constitue le véritable défi de l'équipe.

Nous espérons que les objectifs et les options qui ont été définies dans ce rapport nous permettront de réaliser un projet plus que satisfaisant.