Université Abou Bekr Belkaid
Tlemcen Algérie
جامعة أبي بكر بلقايد
تلمسان الجزائر

République Algérienne Démocratique et Populaire
Université Abou Bakr Belkaid– Tlemcen
Faculté des Sciences
Département d'Informatique

Mémoire de fin d'études

pour l'obtention du diplôme de Master en Informatique

Option: *Système d'Information et de Connaissances (S.I.C) .*

Thème

Compression des fichiers son de type wave

Réalisé par :

- M^elle^ BAKLI Meriem
- M^elle^ TALHAOUI Nora

Présenté le 22 octobre 2012 devant le jury composé de MM.

- BENMAAMAR Bader *(Président)*
- BENZIANE Mohammed Yaghmorasan *(Encadreur)*
- BENAISSA Mohamed *(co-encadreur)*
- BENAMAR Abdelkrim *(Examinateur)*

Année universitaire: 2011-2012

Remerciements

Tout d'abord nous remercions Dieu Tout Puissant de nous avoir donné la force, la volonté, et le privilège d'étudier et de suivre le chemin de la science; ensuite nous remercions les membres de jury, d'avoir accepté de porter un jugement sur ce travail

Nous tenons également à remercier plus particulièrement :

Notre encadreurs Mr.BENZIANE et Mr.BENAISSA Mohamed , tous les enseignants du département d'informatique à université Abou bekr belkaid Tlemcen,

Nous adressons nos plus sincères remerciements à tous nos collègues et nos amis qui partagent avec nous les bons moments d''étude.

Nous exprimons nos gratitude a tous nos proches qui sont m'encouragée au cours de la réalisation de ce mémoire, Sans oublier tout le personnel d'administration du département d'informatique.

Nos remerciements vont également à toutes les personnes qui ont contribué de près ou de loin à l'élaboration de ce mémoire.

Merci à tous

Melle. BAKLI Meriem & Melle.TALHAOUI Nora

DEDICACES

Je dédie ce modeste travail à :

Mes parents pour leur soutien tout au long de mon cursus universitaire.

Notre encadreur Me

Mes frères et sœurs,

Ma famille,

Mes amis,

Et à tous mes camarades de même promotion.

Melle.BakliMeriem

DEDICACES

Je dédie ce travail à :

****En premier lieu à mes parents qui ont consenti beaucoup de sacrifices pour me permettre de réaliser mes objectifs.***

****A mes frères ABDELKADER,MANSOUR,AHMED et SOULIMANE. A mes sœurs. Et à toute la famille TALHAOUI sans exception.***

****A très chère amies, particulièrement, mon binôme BAKLI Meriem et toute sa famille.***

****A toute la promotion deuxième année Master en informatique SIC 2011-2012 de l'université de Tlemcen.***

TALHAOUI Nora

Table de matière

Liste des figures……v
Liste des tableaux……....vii
Liste des abréviations…….....viii
Introduction générale……1

Chapitre I : Introduction aux notions de base du son

I.1. Introduction……4
I.2. Définition……4
I.3. Les caractéristiques du son……7
I.3.1. De l'analogique au numérique……...7
I.3.2. Le son en numérique: un signal discontinu……7
I.4. L'acquisition d'un son……...11
I.5.Conclusion……...…12

Chapitre II : Les algorithmes de compression

II.1. Introduction……..14
II.2. La compression du son…….14
II.3. Principe général de la compression……...…14
II.4. Compression de type statistique……...15
II.4.1. codage de Huffman……..15
II.4.1.1. Principe de l'algorithm……16
II.4.1.1.2. Construction de l'arbre de Huffman……...….17
II.4.1.1.3. Compression / Décompression…….21
II.5. Compression de type Dictionnaire……...22
II.5.1. Exemples de compressions intuitives de type dictionnaire…….....…22

II.5.1.1. Compression de texte multi-dictionnaires.....…...22

II.5.1.2 Compression de texte multi-langages……..22

II.5.2 L'algorithme LZW……...…22

II.5.2.1 Le principe……...23

II.5.2.2. L'algorithme de compression…….....23

II.5.2.3. L'algorithme de décompression……..26

II.6. Compression par MP3……..……28

II.6.1. Les étapes de la compression MP3……...…28

II.6.2. Psycho-acoustique et sons masqués……29

II.7. Conclusion…….....31

Chapitre III : Les formats des fichiers audio

III.1. Introduction…….....33

III.2. Les formats audio……...…33

III.2.1. Les formats audio compressés avec perte……..33

III.2.1.1 MP3: MPEG1 audio layer 3……...33

III.2.1.2. WMA: Windows Media Audio…….34

III.2.1.3. Ogg: Ogg Vorbis……34

III.2.1.4. AAC: Advanced Audio Coding…….34

III.2.1.5. Le mp3pro……..34

III.2.2. Les formats audio compressés sans perte……...35

III.2.2.1. Free Lossless Audio Codec.....35

III.2.2.2. ALAC: Apple Lossless Audio Codec……35

III.2.2.3. L'Atrac 3……...….35

III.2.2.4.FLAC…….....36

III.2.3. Les formats audio sans compression…….....36

III.2.3.1 AIFF: Audio Interchange File Format……36

III.2.3.2. WAV: waveform (forme d'onde)……...36

III.2.3.3. CDA: Compact Disc Audio……...37

III.2.3.4. BWF……...37

III.3. Le format Wav……37

III.3.1. Format (bloc RIFF)……38

III.3.2. Descriptif du son (bloc fmt)……...38

III.3.3. DATA : partie données de fichier wav (bloc data)……..……..40

III.3.4. Structure générale d'un fichier wav……...41

III.4. Caractéristique de fichier wav……41

III.4.1. L'amplitude…….....41

III.4.2. La fréquence……...…42

III.4.3. Le débit……...42

III.4.4. L'ordre des données…….....42

III.4.5. Fichier wav en hexadécimal……...44

III.5. Conclusion……..…44

Chapitre IV : Implémentation et résultats

V.1. Introduction……..…46

V.2. Présentation de l'application……46

V.3 .Les résultats obtenus……48

V.4. Présentation de logiciel GoldWave…….50

V.4.1 .Résultats de la qualité de compression de son par les différents formats de codage de son par GOLDWAVE…….51

V.4.2. L'amplitude de chaque fichier Compressé……52

V.4.2.1. L'amplitude de fichier codé en pcm-unsigned-8bit-mono Compressé à MP3……..... 52

V.4.2.2. L'amplitude de fichier codé en pcm-signed-16bit-mono compressé à MP3……..…….53

V.4.2.3. L'amplitude de fichier codé en pcm-signed-24bit-mono compressé à MP3……...…….53

V .4.2.4. L'amplitude de fichier codé en pcm-unsigned-8bit-stéréo compressé à MP3…….....53

V.4.2.5. L'amplitude de fichier codé en pcm-signed-16bit-stéréo compressé à MP3….....54

V.4.2.6. L'amplitude de fichier codé en pcm-signed-24bit-stéréo Compressé à MP3.....54

V.4.2.7. L'amplitude de fichier codé en pcm-unsigned-8bit-mono converti à MP3par GoldWave……...……...54

V.2.8. L'amplitude de fichier codé en pcm-signed-16bit-mono converti à MP3 par GoldWave……...…….....55

V.4.2.9. L'amplitude de fichier codé en pcm-unsigned-24bit-mono converti à MP3 par GoldWave.....55

V.5. Conclusion…….....56

Conclusion générale…….58

Référence bibliographiques.....59

Annexe ...…….61

Liste de tableaux

Tableau 3.1 : comparatif entre la taille/qualité des principaux formats audio utilisés...37
Tableau 4.1 : temps de compression des différents fichiers son. ……..…....48
Tableau 4.2 : Taille de compression du son par les différents formats de codages…...49
Tableau 4.3 : taille de compression du son par les différents formats de codage. ….....51

Liste de Figure

Figure 1.1 : L'évolution de l'amplitude sonore dans le temps……..…5
Figure 1.2: les types de fréquence…….....6
Figure 1.3: L'exemple d'une chaîne analogique……...……7
Figure 1.4 : Exemple d'une chaîne numérique…….……...8
Figure 1.5: Echantillonnage d'un signal audio…….……9
Figure 1.6: signal échantillonné avant et après quantification……...10
Figure 1.7 : Acquisition d'un son……12
Figure 2.1 : Principe de la compression (cas d'une compression sans pertes)……...15
Figure 2.2 : compression huffman……..19
Figure 2.3 : Psycho acoustique et sons……..29
Figure 2.4 : masque frequentiel…….30
Figure 2.5 : masque temporel……31
Figure 3.1. Amplitude, fréquence……...42
Figure 3.2. Fichier wav en hexadecimal…….44
Figure 4.1 : interface de l'application……47
Figure 4.2 : ouvrir un fichier son……47
Figure 4.3 : jouer un fichier son. ……...48
Figure 4.4 : le temps de compression de différents fichiers son……49
Figure 4.5 : la taille d'origine et la taille de compression d'un fichier son codé en différents formats…….....50
Figure 4.6 : l'interface de logiciel GOLDEWARE……51
Figure 4.7 : la taille d'origine et la taille de compression d'un fichier son codé en différents formats par Goldwave…… ……52

Figure 1 :l'interface de Ocenaudio…….64

Figure 2 :l'interface de Audacity……...……65
Figure 3: interface de Free Audio Editor……...65
Figure 4: interface de Music Editor Free……....66
Figure 5 : interface de Free MP3 Cutter and Editor……...66
Figure 6: interface de Power Sound Editor Free……67
Figure 7 : interface de Shuangs Audio Editor.....67
Figure 8 : interface de Wavosaur…… ..…….68

Figure 9 :l’interface de Mp3DirectCut……. …….……....68
Figure 10 : interface de Wave Editor……..69

.

Liste des abréviations

PCM	**Pulse Code Modulation**
MPEG	**Moving Picture Experts Group**
WMA	**Windows Media Audio**
FLAC	**Free Lossless Audio Codec**
AAC	**Advanced Audio Coding**
VBR	**Variable Bit Rate**
ALAC	**Apple Lossless Audio Codec**
PC	**Personnel Computer**
WAV	**WAVE**
AIFF	**Audio Interchange File Format**
CDA	**Compact Disc Audio**
BWF	**Broadcast Wave Format**
MIT	**Massachusetts Institute of Technology**
TIFF	**Tagged Image Format File**
LZW	**Lempel , Ziv et Welch**
RLC, RLE	**Run Length Coding, Run Length Encoding**
GHz	**Giga Hertz**
RAM	**Ram Access Memory**
MP3	**MPEG1 audio layer 3**
dB	**DéciBels**
CD	**Compact Disque**

I.1. Introduction

Le son est une chose familière dans notre vie quotidienne que l'on en oublie souvent la signification physique qui est loin d'être facile à comprendre.

D'un point de vue physique, un son est une énergie qui se propage sous forme de vibrations dans un milieu compressible (dans l'eau, dans l'air, dans les matériaux solides).

Le traitement du son est la branche du traitement du signal qui s'applique aux signaux audio, dans le but notamment d'améliorer la qualité, de les compresser, ou d'extraire de l'information.

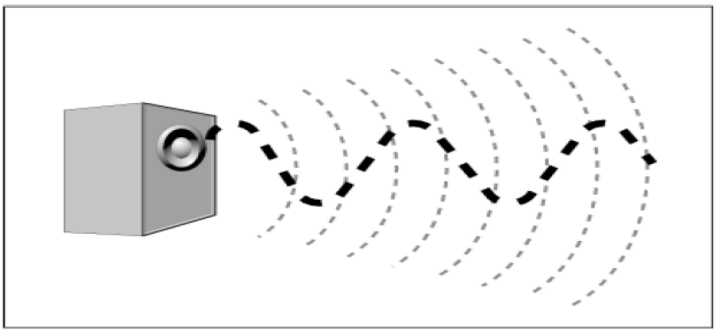
Le traitement du signal est la discipline qui développe et étudie les techniques de traitement, d'analyse et d'interprétation des signaux. Parmi les types d'opérations possibles sur ces signaux, on peut dénoter le filtrage, la compression de données, la numérisation**,** le codage, le chiffrement et la transmission de données.

La science qui étudie les sons s'appelle l'acoustique. La psychoacoustique combine l'acoustique avec la physiologie et la psychologie, pour déterminer la manière dont les sons sont perçus et interprétés par le cerveau.

Dans ce chapitre, nous présentons les notions de base sur le son.

I.2. Définition

Le son est une somme de vibrations, produites par des cordes vocales, un haut-parleur, etc…



Ces vibrations ont une fréquence, mesurée en Hertz. L'oreille humaine est un récepteur ne percevant que certaines fréquences : la bande 20Hz – 20Khz. **[1]**

I.3. Les caractéristiques du son

Comme tout phénomène vibratoire, le son peut être analysé comme un signal qui varie dans le temps. Les caractéristiques essentielles sont l'amplitude et la fréquence.

L'amplitude

La première caractéristique d'un son est son amplitude. Appelée aussi intensité ou volume sonore, c'est l'expression de la pression de l'air qui se mesure en décibels (dB).

0 dB correspondent au minimum que l'oreille humaine puisse percevoir (seuil d'audibilité).

Attention, une augmentation de 3db multiplie la puissance par deux!

Faible

Fort

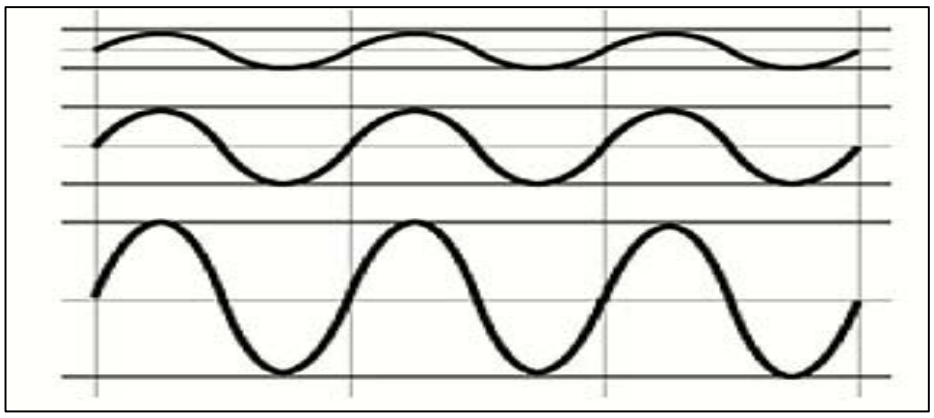


Figure 1.1 : L'évolution de l'amplitude sonore dans le temps

Exemple concret:

- De 0 à 10 dB : Seuil d'audibilité, Désert
- De 30 à 40 dB : forêt
- De 60 à 70 dB : sonnerie de téléphone
- De 80 à 90 dB : tondeuse à gazon, klaxon de voiture
- 120 dB : seuil de la douleur, avion au décollage
- 180 dB : décollage de la fusée Ariane, lancement d'une roquette

Fréquence

La fréquence, exprimée en Hertz (Hz), est le nombre de répétition d'une période par seconde.

Plus elle est élevée et plus le son paraîtra « aiguë », à l'inverse, il paraîtra « grave ». En musique, la fréquence définit donc la hauteur d'un son, soit, la note. (Ex: la note « LA » correspond à 440Hz, soit 440 vibration en une seconde). **[2]**

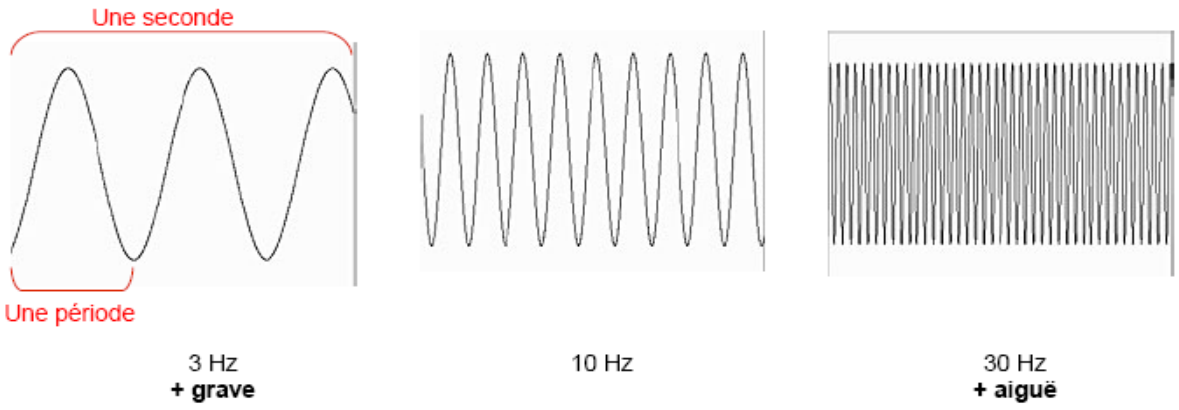


Figure 1.2: les types de fréquence

I.3.1. De l'analogique au numérique

Le son analogique : un signal continu

Lorsqu'on capte un son à partir d'un microphone, ce dernier transforme l'énergie mécanique (la pression de l'air exercée sur sa membrane), en une variation de tension électrique continue.

Ce signal électrique dit « analogique » pourra ensuite être amplifié, et envoyé vers un haut parleur dont la fonction est inverse: transformer à nouveau le signal électrique en une énergie mécanique (on peut observer le déplacement de la membrane d'un haut parleur en marche).

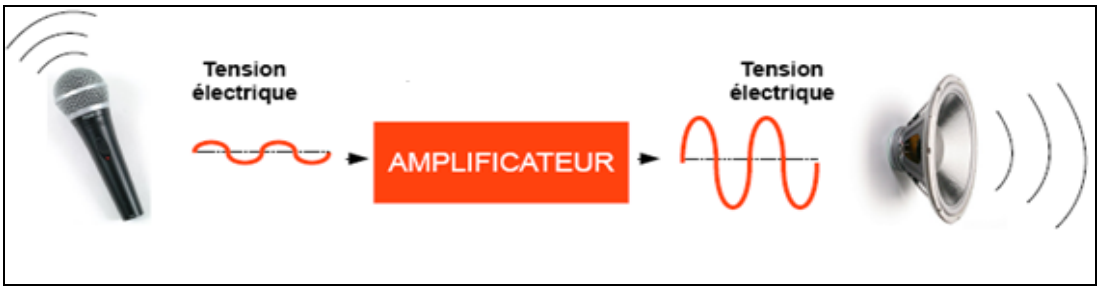


Figure 1.3: L'exemple d'une chaîne analogique

Le son analogique est généralement fixé sur des supports comme les bandes magnétiques, disques vinyles etc. Le problème rencontré par ces supports réside dans l'usure physique des informations au cours de leur utilisation (lecture/écriture). A terme, le signal est affaibli et peut disparaitre. **[2]**

I.3.2. Le son en numérique: un signal discontinu

Avec l'informatique, lorsque ce même signal électrique est capturé à partir du micro, il est converti en une suite de nombre, on parle alors de numérisation du signal. C'est la carte son qui s'en charge, elle contient des entrées (convertisseurs analogique vers numérique) et des sorties (convertisseurs numérique vers analogique).

La première phase appelée numérisation consiste donc à passer d'un signal continu (une variation de tension électrique) en une suite de valeurs mesurées à intervalles réguliers, donc discontinu.

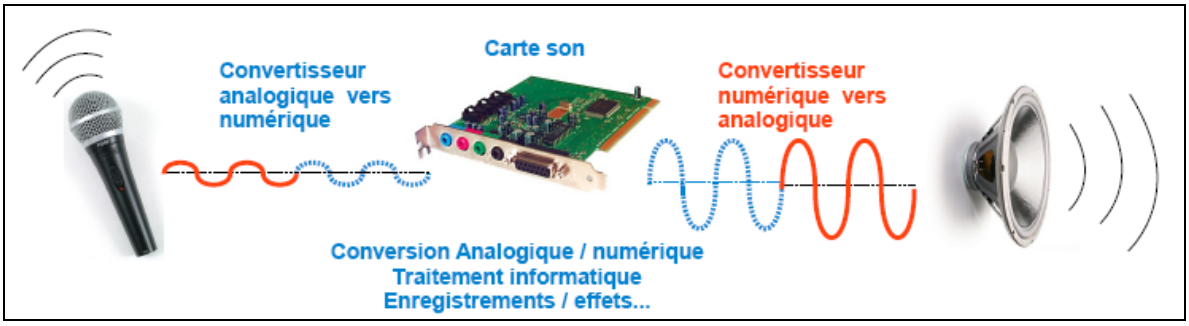


Figure 1.4 : Exemple d'une chaîne numérique

L'avantage du numérique, est la possibilité de lire et de dupliquer autant de fois ce signal sans aucune détérioration, puisqu'il a été réduit en une suite de nombres stockée dans un fichier informatique! Cela dit, la compression audio comme le MP3 peut provoquer une perte volontaire du signal afin d'économiser de l'espace de stockage. **[2]**

◆ **La fréquence d'échantillonnage**

Échantillonner un signal audio analogique revient à prélever ses valeurs de tension électrique un certain nombre de fois par seconde. La fréquence de ces prélèvements est appelée fréquence d'échantillonnage. La fréquence d'échantillonnage est fixée avant l'opération de numérisation et ne varie pas pendant la numérisation.

Les fréquences d'échantillonnage couramment utilisées en audio sont 44100Hz et 48000Hz. Elles sont souvent imposées par des contraintes technologiques. Par exemple, la norme du disque compact audio (CD audio) impose une fréquence d'échantillonnage de 44100Hz.

L'échantillonnage est effectué par découpage temporel du signal audio analogique. Ce découpage temporel permet de reconstruire en données chiffrées la forme d'onde du signal numérisé. La numérisation ne repose que sur des séries de 0 et de 1 : il s'agit d'un codage binaire. **[3]**

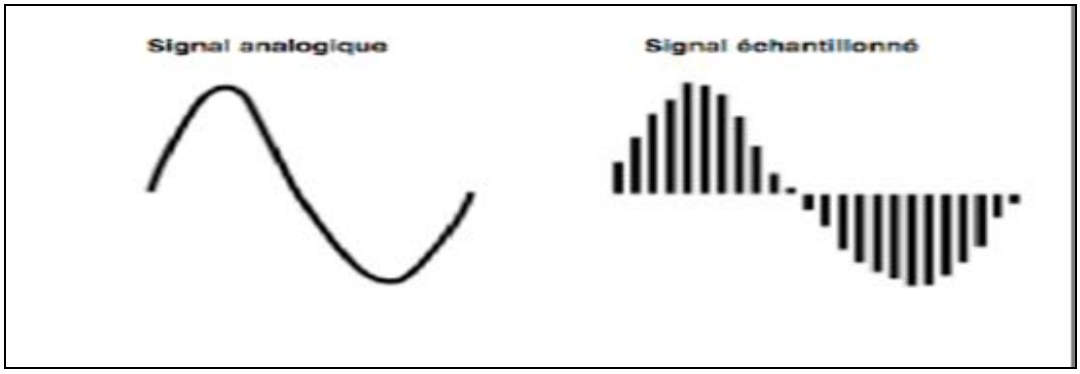


Figure 1.5: Echantillonnage d'un signal audio

- **La quantification**

Alors que l'échantillonnage opère un découpage temporel, l'opération de quantification crée une échelle de valeurs discrètes permettant d'attribuer à chaque échantillon une valeur d'amplitude. La quantification s'exprime en « bit » (un acronyme de binary digit). Les valeurs couramment utilisées en audio sont 16bit et 24bit.

L'amplitude de chaque échantillon doit impérativement prendre l'une des valeurs définies par l'échelle de quantification. Si la valeur d'amplitude de l'échantillon se situe entre deux paliers de l'échelle de quantification, elle est approximée au palier le plus proche. Cette approximation induit une erreur que l'on nomme « erreur de quantification ».

Par suite, plus le nombre de bits est élevé, plus le nombre de paliers est important et l'erreur de quantification faible. Autrement dit, les petites variations d'amplitude du signal échantillonné sont d'autant mieux approximées que la résolution de la quantification est élevée.

La fidélité de la forme d'onde numérisée à la forme d'onde du signal analogique dépend donc de la résolution (exprimée en bit) et de la fréquence d'échantillonnage (exprimée en kHz).

De même que pour la fréquence d'échantillonnage, le choix de la résolution de la quantification est soumis à des contraintes techniques. **[3]**

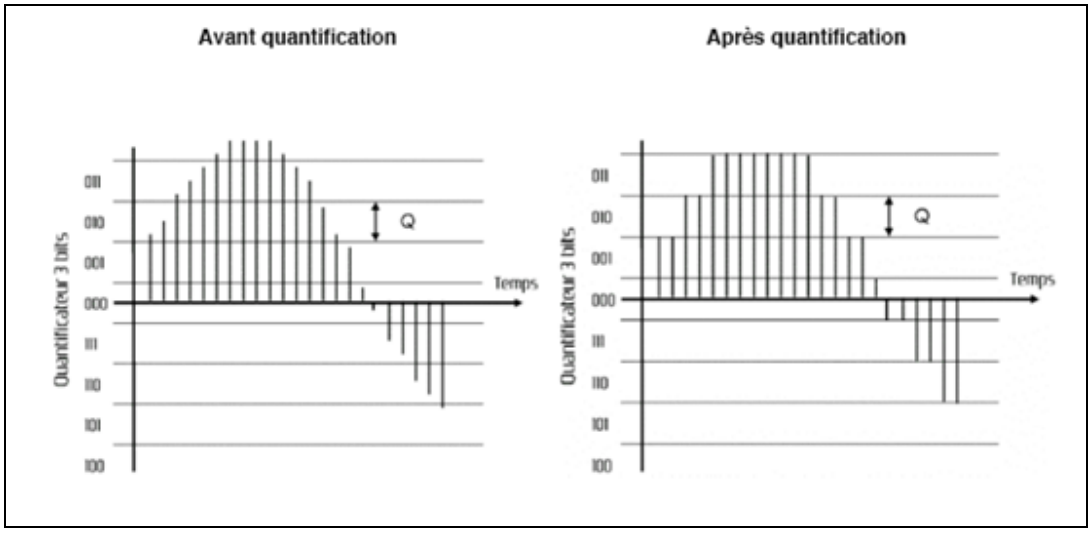


Figure 1.6: signal échantillonné avant et après quantification.

❖ **La compression du signal audio numérisé**

Un signal audio numérisé est stocké sur des disques durs, des disques compacts, des DVD… La nature de l'information qu'ils contiennent rend ces fichiers relativement volumineux. L'intérêt de la compression de stockage limitée (ex : baladeur mp3).

Les techniques de réduction de débit sont déjà très largement employées dans les domaines du cinéma et de la radio, via le câble, le satellite. **[3]**

✓ **Les algorithmes de compression**

Un algorithme est l'énoncé d'une suite d'opérations permettant de donner la réponse à un problème.

Dans le cas de la compression, l'algorithme a pour fonction de réduire la taille d'un fichier selon un certain nombre de contraintes que le programmeur spécifie. Par exemple, une des contraintes peut être de conserver toutes les fréquences inférieures à 20kHz afin de limiter les pertes de qualité sonore dans la zone audible du spectre.

Lors de l'étape de compression et de décompression d'un flux audio ou vidéo, on utilisera des algorithmes spécifiques rassemblés sous le terme commun de « CoDec ». Un codec est constitué de deux éléments :

• le COdeur contient un algorithme destiné à coder l'information. Dans le cas de la compression ce sera pour effectuer une réduction du poids des données ;
• le DECodeur contient un algorithme destiné à décoder l'information. Dans le cas de la compression ce sera pour reconstruire un signal audionumérique. **[3]**

✓ **Le taux de compression**

Compresser revient à réduire le débit du flux audio et/ou vidéo. Les algorithmes sont adaptés en fonction des applications (diffusion internet, télévision, cinéma) pour répondre aux besoins de chacun des médias. La réduction de débit (ou compression) s'exprime généralement sous la forme d'un taux dit « taux de compression ». Le taux de compression peut s'énoncer comme suit:
- soit comme le rapport entre le volume initial des données et le volume après réduction. Si le volume de données est deux fois plus faible après réduction (passant de 10Mo à 5Mo par exemple), on écrira qu'il s'agit d'un taux de 2:1 ;
- soit en pourcentage du volume après réduction par rapport au volume initial. Si le volume de données est deux fois plus faible après réduction, on écrira qu'il s'agit d'un taux de 50%.
Il existe par ailleurs deux types de compressions : la compression « destructive » et la compression« non destructive ». **[3]**

I.4. L'acquisition d'un son

L'acquisition se fait grâce à un microphone connecté à l'ordinateur.
Les technologies d'acquisition du son ont été longtemps analogiques : le son était représenté par les variations d'une grandeur physique, une tension électrique par exemple. Les techniques actuelles permettent d'obtenir directement un son numérisé : c'est notamment le cas des
magnétophones produisant un enregistrement sur cassettes D.A.T. (Digital Audio Tapes, digitalisation du son à une fréquence de 48 KHz) .**[4]**

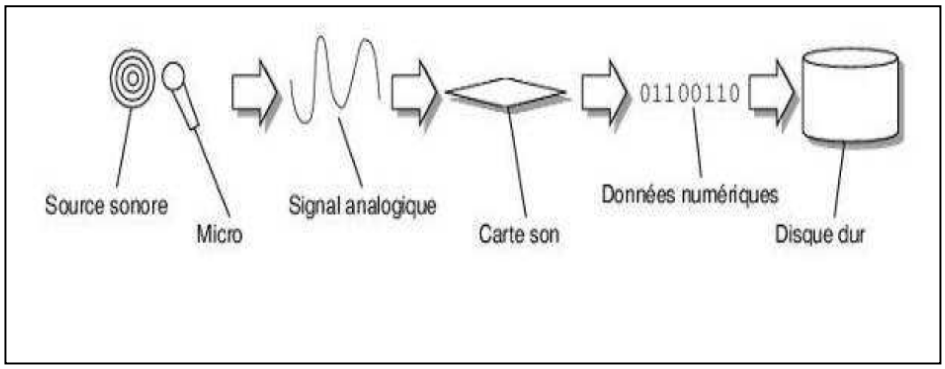


Figure 1.7 : Acquisition d'un son.**[5]**

I.5. Conclusion

Depuis la découverte de la synthèse numérique des sons, et avec l'arrivée d'ordinateurs personnels qui contient des cartes son, et qui permettre d'enregistrer et de traiter le son.

Dans ce chapitre, nous avons présentés les notions de base du son. Le chapitre suivant sera consacré sur les différents formats des fichiers audio.

II.1. Introduction

La compression de données est une partie de l'encodage de données au même titre que le cryptage de données (cryptographie) et la transmission de données. Les modems actuels utilisent systématiquement la compression pour atteindre des performances.

Presque chaque format de fichier incorpore l'une ou l'autre méthode de compression. Dans ce chapitre nous pressentons les notions de base de la compression et les algorithmes de la compression.

II.2. La compression du son

La compression consiste à réduire le temps et l'espace mémoire occupé par un fichier son

Pour compresser le son, plusieurs méthodes sont possibles :

- Supprimer les hautes fréquences quasiment inaudibles
- Supprimer les vibrations parasites
- Diminuer la fréquence d'échantillonnage (Attention, cela diminue fortement le rendu sonore).

• La solution : le VBR (Variable Bit Rate) la fréquence d'échantillonnage s'adapte au son.

En fonction de la fréquence d'échantillonnage, de la résolution sonore et du mode mono ou stéréo, on obtient un débit, mesuré en Kbit/s, représentatif de la qualité sonore. En VBR, le débit est différent à chaque instant**. [6]**

Valeurs typiques

• Un CD : 1440 Kbit/s

• Un fichier MP3 compressé au minimum : 320 Kbit/s

• Qualité correcte au format MP3 : 128 ou 192 Kbit/s

• Un fichier MP3 compressé au maximum : 64 Kbit/s

• Un son de qualité téléphonique : 32 Kbit/s.

II.3. Principe général de la compression

Les données (texte, son, images, vidéo, programmes ou autres fichiers) ont souvent besoin d'être compressées pour des raisons d'espace de stockage et/ou de vitesse de transfert.

Il s'agit donc de convertir des données d'un format d'origine vers un format compressé, ce qui peut se faire avec perte ou sans perte d'information.

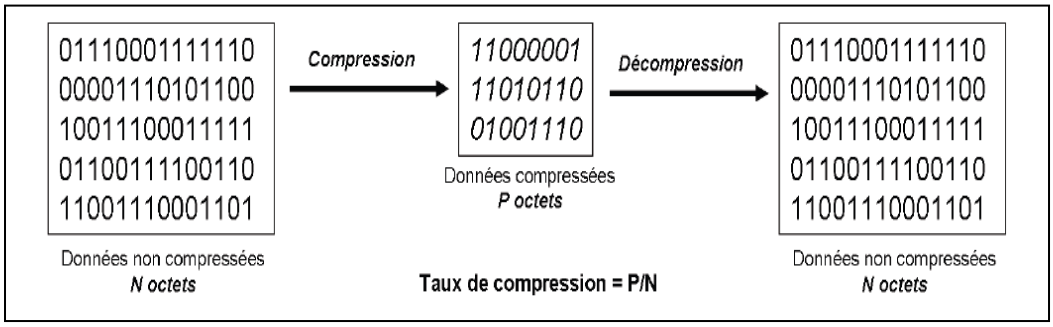


Figure 2.1 : Principe de la compression (cas d'une compression sans pertes).

L'efficacité de la compression d'un fichier est mesurée par le **taux de compression**, défini comme le rapport entre la taille du fichier compressé et celle du fichier initial. Avec cette définition, plus le taux de compression est faible, meilleure est la compression.**[7]**

II.4. Compression de type statistique

II.4.1. codage de Huffman

Le codage de Huffman est un algorithme de compression des données basé sur les fréquences d'apparition des caractères apparaissant dans le document initial. Il a été développé par un étudiant de la MIT (Massachusetts Institute of Technology), David A. Huffman en 1952. Cette technique est largement utilisée car elle est très efficace et on observe selon le type de données des taux de compression allant de 20% à 90% mais plus généralement entre 30% et 60%. La dernière partie de cette section traitera des performances. Le principe de compression est utilisé dans le codage d'image TIFF (Tagged Image Format File) spécifié par Microsoft Corporation et Aldus Corporation. La méthode JPEG (Join Photographic Experts Group) utilise aussi la compression de type Huffman pour coder les informations d'une image. (Elle utilise d'ailleurs des tables prédéfinies).

Ce procédé fait partie des méthodes de compression de type dites statistiques. Cela repose sur le principe sur l'attribution de codes plus courts pour des valeurs fréquentes

et de codes plus longs pour les valeurs moins fréquentes. Cela est plus efficace que la représentation actuelle qui consiste, quant à elle, à utiliser une longueur fixe pour chaque symbole (exemple : un octet par caractère, code ASCII.

II.4.1.1. Principe de l'algorithme

L'algorithme de Huffman utilise une table contenant les fréquences d'apparition de chaque caractère pour établir une manière optimale de les représenter par une chaîne binaire (cela reprend le principe du morse qui tend à minimiser le nombre de symboles à utiliser pour les lettres les plus fréquemment employées).

On peut décomposer la procédure en plusieurs parties :

– Tout d'abord, la création de la table de fréquence d'apparition des caractères dans le texte initial.

– Ensuite la création d'un arbre binaire (usuellement dénommé arbre de Huffman) suivant la table précédemment calculée. (Remarque : on devrait parler plutôt de l'arborescence de Huffman.)

– Enfin coder les symboles en représentation binaire optimale.

II.4.1.1.1. Table de fréquence d'apparition

Afin de construire cette table, il suffit simplement de dénombrer le nombre d'occurrences de chaque symbole s puis de calculer la fréquence fs de chacun d'entre eux grâce à la formule suivante :

$$fs = \frac{\text{nombre d'occurrences de } s}{\text{nombre de symboles}}$$

Ensuite on trie le tableau en fonction de la fréquence d'apparition (de façon croissante) puis suivant le symbole suivant. Supposons par exemple que nous ayons le texte suivant composé des symboles pris dans le code ASCII :

"L'algorithme de Huffman est une méthode qui permet de compresser les données".

Pour chaque symbole on calcule tout d'abord le nombre d'occurrences de chacun puis sa fréquence d'apparition suivant la formule citée précédemment.

La table de fréquences T triée est (Afin de simplifier l'exemple, on a négligé la casse) :

Symbole	Nombre d'occurrences	Fréquence d'apparition
c	1	$1/76 = 1.32\%$
g	1	$1/76 = 1.32\%$
q	1	$1/76 = 1.32\%$
'	1	$1/76 = 1.32\%$
a	2	$2/76 = 2.63\%$
f	2	$2/76 = 2.63\%$
i	2	$2/76 = 2.63\%$
p	2	$2/76 = 2.63\%$
u	3	$3/76 = 3.95\%$
h	3	$3/76 = 3.95\%$
l	3	$3/76 = 3.95\%$
d	4	$4/76 = 5.26\%$
n	4	$4/76 = 5.26\%$
o	4	$4/76 = 5,26\%$
r	4	$4/76 = 5.26\%$
t	4	$4/76 = 5.26\%$
m	5	$5/76 = 6.58\%$
s	5	$5/76 = 5.26\%$
	11	$11/76 = 14.47\%$
e	14	$14/76 = 18.42\%$

II.4.1.1.2. Construction de l'arbre de Huffman

L'arbre binaire de Huffman est la structure de données qui va nous permettre d'attribuer à chaque symbole une représentation binaire optimale. Afin de construire l'arbre, on utilise la table de fréquences précédemment construite qu'on appelle T et on applique l'algorithme suivant :

L'Algorithme : Construction de l'arbre

Données :

– T : la table de fréquence

– Q : Une file d'attente de noeuds de l'arbre binaire. Chaque feuille est étiquetée avec un symbole et son nombre d'occurrences. Chaque noeud interne est étiqueté avec la somme des occurrences des feuilles de sa sous arborescence.

– o : Une fonction qui à chaque noeud de l'arbre associe une valeur. Si le noeud est une feuille alors o renvoie le nombre d'occurrences du symbole, autrement o renvoie la somme des occurences des feuilles de la sous-arborescencedu noeud.

Résultat :

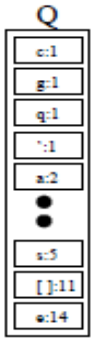
– A : L'arbre binaire résultant

```
begin
    Initialisation de Q tel que :
    Q contienne les feuilles représentant les symboles de la table T
    tant que (Q non vide) faire
        Créer un nouveau noeud z dans A tel que :
        gauche(z) = x = extraire-min (Q)
        droite(z) = y = extraire-min (Q)
        o(z) = o(x) + o(y)
        Insérer (z,Q)
    fin
end
```

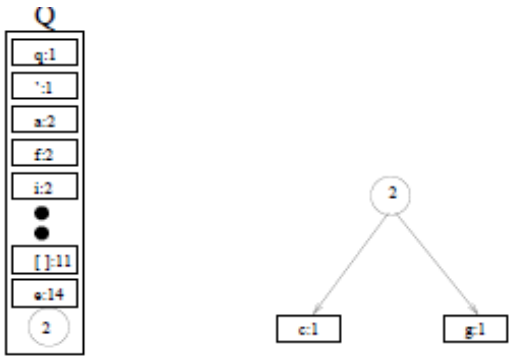
De façon informelle, on utilise une file d'attente Q dans laquelle on place les noeuds correspondants au couple [symbole : nombre d'occurrences du symbole] de tous les symboles. Ensuite on extrait de la file d'attente les 2 noeuds ayant la valeur minimale puis on crée un nouveau noeud dans l'arbre de Huffman ayant pour fils les 2 deux sélectionnés, on rajoute ensuite le noeud nouvellement crée dans la file d'attente , et on réitère jusqu'à ce que la file soit vide.

Reprenons l'exemple précédent cité et appliquons l'algorithme de construction de l'arbre étape par étape.

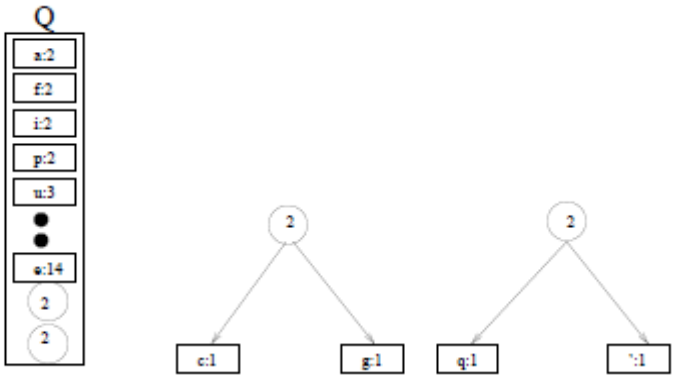
Initialisation



Etape 1



Etape 2



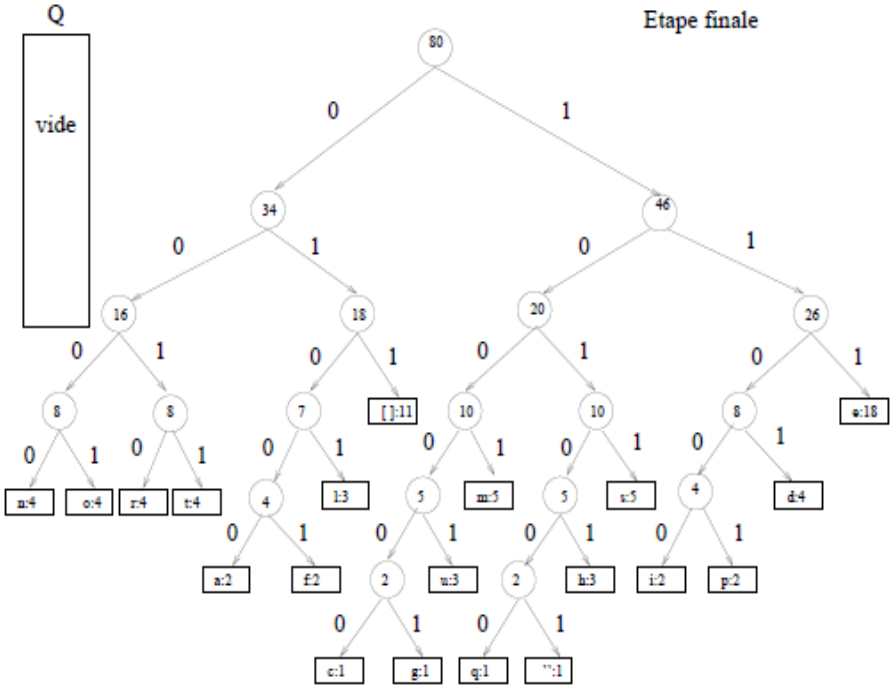


Figure 2.2 : compression huffman

L'arborescence de Huffman finalement construite, on étiquette les arcs de la façon suivante :

– Les arcs reliant un noeud à son fils gauche sont étiquetés par '0'.

– Les arcs reliant un noeud a son fils droit sont étiquetés par '1'.

De cette manière, chaque feuille représentant un symbole peut être redéfinie par un nombre binaire correspondant au chemin entre la racine et la feuille de l'arborescence.

Ainsi, les symboles les plus utilisés ont une représentation binaire moins importante (en terme de taille) que les symboles les moins utilisés. Ceci permet de représenter chaque symbole de façon optimale et permet de réaliser une compression des données efficacement.

Si on reprend notre exemple, on obtient la table de correspondance suivante :

Symbole	Représentation binaire	Taille(en bits)	Gain(en bits)
c	100000	6	2
g	100001	6	2
q	101000	6	2
'	101001	6	2
a	01000	5	3
f	01001	5	3
i	11000	5	3
p	11001	5	3
u	10001	5	3
h	10101	5	3
l	0101	4	4
d	1101	4	4
n	0000	4	4
o	0001	4	4
r	0010	4	4
t	0011	4	4
m	1001	4	4
s	1011	4	4
	011	3	5
e	111	3	5

De cette manière, notre phrase exemple devient :

"L'algorithme de Huffman est une méthode qui permet de compresser les données"

l = 0101

' = 101001

a = 01000

g = 100001

...

Message compressé = 0101101001010000101100001 etc...

(Á noter que le message compressé est une suite binaire de bits alors que le message original est une suite de symboles qui peut, comme nous l'avons vu en cours, être ramenée à une suite de bits).

II.4.1.1.3. Compression / Décompression

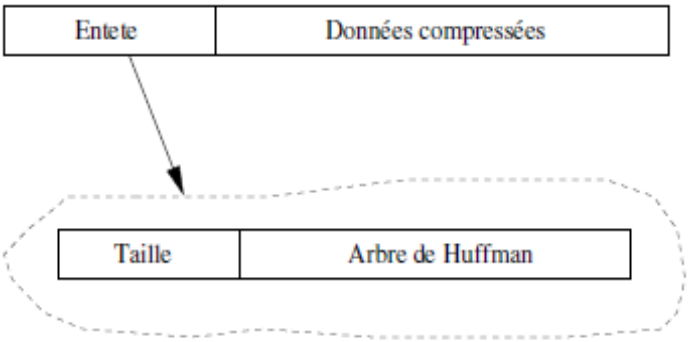
- **Compression** :

Pour la compression des données, on applique l'algorithme de Huffman comme indiqué précédemment afin de calculer l'arbre binaire de Huffman.

Ensuite, lorsqu'on désire transmettre des données à un destinataire, l'expéditeur émet un fichier contenant 2 parties :

– Un entête : Celui ci contient la taille originale du fichier, l'arbre binaire de Huffman calculé précédemment (Cette structure de données peut être aisément représenter par un tableau simple).

– Les données : Chaque symbole des données originales est substitué par sa représentation binaire correspondante.



Message à transmettre compressé

- **Décompression** :

Lors de la réception d'un message, le destinataire récupère dans l'entête l'arbre de Huffman et substitue chaque suite de bits correspondant à un chemin dans l'arborescence de Huffman par le symbole équivalent. On remarque qu'il ne peut pas avoir d'ambiguïté sur l'interprétation d'une séquence de bits à cause de la structure intrinsèque de l'arborescence.**[8]**

II.5. Compression de type Dictionnaire

L'algorithme de Huffman a longtemps dominé le monde de la compression, et c'est en 1978 où sont apparues les méthodes de compression de Abraham Lempel et Jacob Ziv, méthodes dites de type dictionnaire aussi appelés algorithmes à substitution de facteurs.

II.5.1 Exemples de compressions intuitives de type dictionnaire

II.5.1.1 Compression de texte multi-dictionnaires

Cette méthode utilise un dictionnaire pour chaque taille de mot. L'idée est de remplacer un mot par la taille de celui ci et son index dans le dictionnaire. Une telle compression entraîne des gains très significatifs.

Mais cette technique n'est pas utilisée car elle nécessite des astuces pour repérer les conjugaisons, les pluriels ou encore les majuscules. De plus elle nécessite qu'un dictionnaire soit présent pour compresser et pour décompresser, ce qui représente une contrainte en espace mémoire.

II.5.1.2 Compression de texte multi-langages

Le principe d'une méthode de compression de texte multi-langages est de remplacer certains mots du dictionnaire par leur traduction dans une autre langue, de façon à diminuer la longueur de la séquence de mots dans une phrase. On peut ainsi obtenir de bons taux de compression.

En effet si nous prenons une phrase française telle que 'nous mangeons', elle pourrait être remplacée par 'we eat', qui est une traduction franco-anglaise et qui permet à la fois un décodage simple et un gain en taille(7 caractères dans ce cas).

Néanmoins l'utilisation d'une telle méthode ne peut fonctionner que s'il n'existe aucune hésitation sur l'interprétation, telle que les phrases à interdépendance Syntaxe sémantique (par exemple La petite brise la glace).

II.5.2 L'algorithme LZW

C'est en 1977 que Jacob Ziv et Abraham Lempel fournissent une technique de compression différente de l'algorithme de Huffman, et capable de donner de meilleurs taux de compression. Ils mettent ainsi en place l'algorithme LZ77.

Puis vient LZSS, version amélioré de LZ77 par Storer et Szymanski puisque la recherche des séquences dans le dictionnaire est réduite logarithmique ment.

Enfin vient l'algorithme LZ78, plus connu sous le nom LZW, amélioration faite par Terry Welch en 1984 de LZSS de par le fait que les séquences sont rangées dans une arborescence. Il porte le nom de ses 3 inventeurs : Lempel, Ziv etWelch.

II.5.2.2 Le principe

Le principe est fondé sur le fait qu'une séquence de caractères peut apparaître plusieurs fois dans un fichier.

L'algorithme LZW de compression consiste à émettre à la place des séquences, les adresses des de ces séquences d'un dictionnaire généré à la volée. C'est un algorithme de compression nettement plus performant en moyenne que les algorithmes statistiques puisqu'il permet d'obtenir des gains plus élevés sur la majorité des fichiers.

L'algorithme LZW se distingue des méthodes statistiques pour plusieurs raisons :

– Le fichier comprimé ne stocke pas le dictionnaire, ce dernier est automatiquement généré lors de la décompression. Il n'existe donc pas de table d'entête.

– Contrairement aux méthodes statistiques qui utilisent la probabilité de présence sur un ensemble de taille fixe de symboles, l'algorithme LZW représente un algorithme d'apprentissage, puisque les séquences répétitives de symboles sont dans un premier temps détectées puis compressées seulement lors de leurs prochaines occurrences. Le taux de compression est dépendant de la taille du fichier. Plus la taille est importante, et plus le taux de compression l'est aussi.

– Il permet le compactage à la volée, puisqu'il n'y a pas à lire le fichier au préalable, il compresse les séquences de symboles au fur et à mesure.

II.5.2.3 L'algorithme de compression

- **Le principe de l'algorithme LZW de compression**

L'objectif de l'algorithme de compression LZW est de construire un dictionnaire où chaque séquence sera désignée par une adresse dans celui ci. De plus chaque séquence ne s'y trouvant pas y est rajoutée. Au final, on se retrouve avec une suite d'entiers, des adresses pointant vers une séquence contenue dans le dictionnaire.

Le dictionnaire est donc un tableau dans lequel sont rangées des séquences de symboles de taille variable, repérées par leurs adresses (leur position dans le tableau). La taille de ce dictionnaire n'est pas fixe et les premières adresses de 0 à 255 du dictionnaire contiennent les codes ASCII. Les séquences ont donc des adresses supérieures à 255.

L'algorithme LZW de compression

Données :

– Le dictionnaire des symboles rencontrés : Dico

– Le fichier à compresser : Fichier

Résultat :

– Le fichier compresser : Fichier

début

```
    s=premier octet du fichier
  tant que le fichier n'est pas à sa fin
    t=octet suivant
    u=concaténation(s,t)
    si (u appartient Dico) s=u
    sinon
    ajouter (u) dans Dico
    écrire adresse de s
    s=t
    fin si
  fin tant que
  écrire adresse de s
```

fin

-Exemple d'utilisation de l'algorithme LZW de compression

A l'aide de l'algorithme LZW, nous voulons compresser la séquence suivante :

'SISI-ET-ISIS'

On lit donc le premier caractère soit 'S'. Puis on lit le suivant : 'I'. On forme la séquence u=s+t soit 'SI', puisque celle ci n'est pas présente dans le dictionnaire,on l'ajoute dans le dictionnaire. Cette séquence aura l'adresse 256, première séquence située dans le dictionnaire après la table ASCII. Enfin on attribue l'adresse de s (valeur ASCII de 'S') dans le fichier soit 83 ou (1010011en binaire). On continue ensuite avec le caractère suivant.

La compression effective a lieu lorsque qu'on rencontrera pour la seconde fois une paire ('SI' par exemple) déjà présente dans le dictionnaire. Dans ce cas on émettra l'adresse de la séquence 'SI' et non l'adresse de la séquence 'S' et de la séquence 'I', on ajoutera

par la suite dans le dictionnaire la séquence de 3 caractères (soit 'SIS' dans notre exemple).

Voici un tableau résumant les opérations effectuées sur l'exemple lors du déroulement de l'algorithme LZW de compression :

	Phase 1	Phase 2	Phase 3	phase 4	Phase 5	Phase 6
Valeur de s	S	I	S	SI	-	E
Valeur de t	I	S	I	-	E	T
Valeur de u	SI	IS	SI	SI-	-E	ET
u appartient au Dico ?	Non	Non	**Oui**	Non	Non	Non
Ajout dans le Dico de	SI	IS		SI-	-E	ET
Ecrire adresse de	S soit **83**	I soit **73**		SI soit **256**	- soit **45**	E soit **69**

	Phase 7	Phase 8	Phase 9	phase 10	Phase 11	Phase 12
Valeur de s	T	-	I	IS	I	IS
Valeur de t	-	I	S	I	S	
Valeur de u	T-	-I	IS	ISI	IS	
u appartient au Dico ?	Non	Non	**Oui**	Non	**Oui**	
Ajout dans le Dico de	T-	-I		ISI		
Ecrire adresse de	T soit **84**	- soit **45**		IS soit **257**		IS soit **257**

Voici le dictionnaire résultant de l'algorithme :

Adresse	Séquence
0..255	ASCII
256	SI
257	IS
258	SI-
259	-E
260	ET
261	T-
262	-I
263	ISI

Enfin voici la suite d'adresse contenu dans le fichier compressé :

83.73.256.45.69.84.45.257.257

On peut ajouter que les adresses ne sont pas de taille maximale dès le départ de l'algorithme. Elles sont incrémentées d'1 bit dès que l'on atteint une nouvelle puissance de 2.

Dans notre exemple lorsqu'on obtient une adresse à écrire de taille plus grande que 255, les adresses sont incrémentées d'1 bit.

Plusieurs méthodes sont possibles pour permettre de prévenir le décompresseur du changement de la taille des adresses.

Premièrement, l'utilisation d'un code spécial est nécessaire qui a pour adresse par exemple 255 et signifie que les adresses seront incrémentées de 1 bits à partir de sa lecture. Ainsi le décompresseur est prévenu lors de la lecture d'un telle code et sait alors que le nombre de bits à lire est incrémenté (dans ce cas ci soit 9 bits). Ce processus fonctionne si l'on condamne certaines adresses du dictionnaire (255, 511,1023 ...,) en n'y plaçant pas de séquences d'octets.

Remarque : Un problème se pose, c'est la façon dont on peut différencier la vraie valeur '255' de l'indicateur d'augmentation de bits. Il suffit pour résoudre ce problème d'émettre l'adresse 255 sur 8 bits suivi immédiatement du code '255' mais sur 9 bits soit 011111111.

Une autre méthode, moins élégante, qui utilise une entête pour informer l'algorithme de décompression des positions où dans le fichier l'on à augmenter la taille des adresses.

Le code 'SP' peut ne pas être le seul code de communication avec l'algorithme de décompression. D'autres codes peuvent par exemple servir à vider des dictionnaires périodiquement, à créer d'autres dictionnaires : il n'y a pas de limites.

II.5.2.4 L'algorithme de décompression

- **Le principe de l'algorithme LZW de décompression**

L'algorithme de décompression LZW reconstruit le dictionnaire au fur et à mesure de la lecture du fichier compressé. Son processus est très proche de l'algorithme de compression.

On commence par lire les codes du fichier en partant d'une taille de 8 bits.

Puis suivant la méthode utilisée, lorsque le décompresseur rencontre un code spécial il augmente la taille de son tampon pour lire les adresses. Il continue à lire les adresses jusqu'à la fin du fichier et il inscrit la séquence correspondant à l'adresse dans le fichier décompresser.

Voici l'algorithme de décompression :

L'algorithme LZW de décompression

Données :

– Le dictionnaire des symboles rencontrés : Dico

– Le fichier à décompresser : Fichier

Résultat :

– Le fichier décompresser : Fichier

début

a=Séquence(première adresse contenu dans le fichier)
écrire a
tant que le fichier n'est pas à sa fin
b=adresse suivante
si (b appartient Dico) s=Séquence(b)
sinon s=concaténation(a,t)
fin si
écrire séquence(s)
t=s[0]
ajouter (Séquence(a)+t) dans Dico
a=b
fin tantque

fin

Exemple d'utilisation de l'algorithme LZW de décompression

A l'aide de l'algorithme LZW, nous voulons décompresser la séquence suivante: '83.73.256.45.69.84.45.257.257' On lit donc la première adresse soit 83. On écrit la séquence associée à cette adresse du dictionnaire. Ensuite on lit l'adresse suivante. On s'assure que l'adresse lue appartient au dictionnaire et que ce n'est pas un code spécial de contrôle, ensuite on écrit la séquence associée à cette adresse. On ajoute dans le dictionnaire la concaténation de l'ancienne adresse lue et la première lettre de la séquence.

On continue ensuite avec les adresses suivantes jusqu'à la fin du fichier.

Remarque : nous ne nous préoccuperons pas de la taille des adresses inscrites dans le fichier à décompresser.

Voici un tableau résumant les opérations effectuées sur l'exemple lors du déroulement de l'algorithme LZW de compression :**[8]**

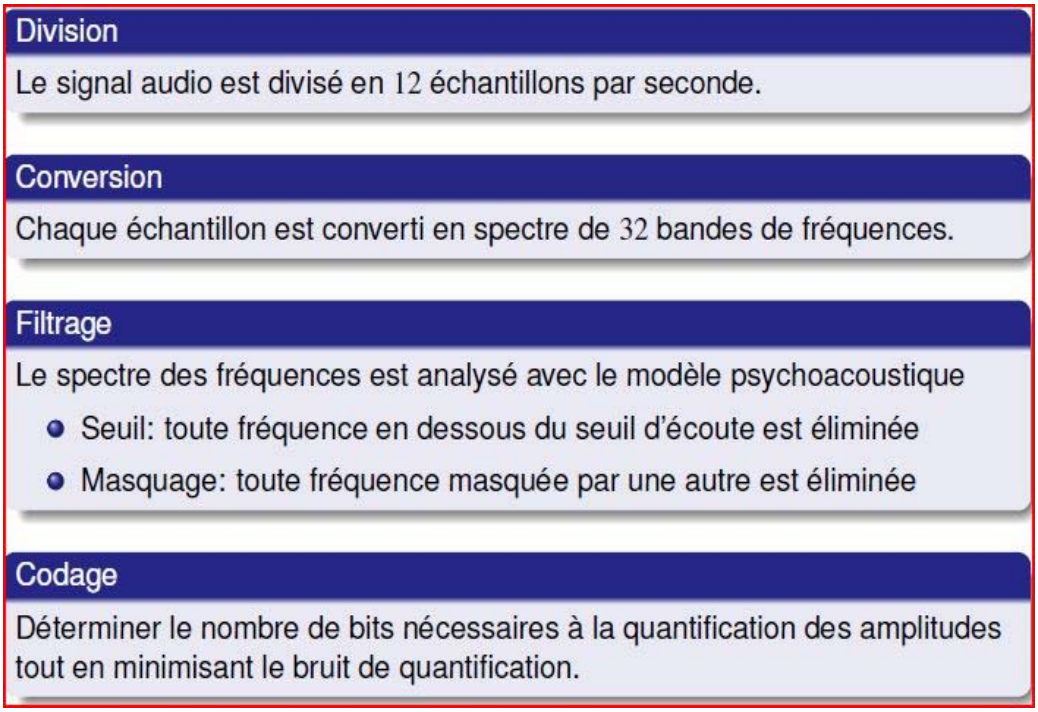
	Phase 1	Phase 2	Phase 3	phase 4
Valeur de a	S=*Séquence*(83)	I=*Séquence*(73)	SI=*Séquence*(256)	-=*Séquence*(45)
Valeur de b	73	256	45	69
b appartient au Dico ?	Oui	Oui	Oui	Oui
Valeur de s	I=*Séquence*(73)	SI=*Séquence*(256)	-=*Séquence*(45)	E=*Séquence*(69)
Valeur de t	I	S	-	E
Ajout dans le Dico de	Si	IS	SI-	-E
Ecrire séquence	**S** *et* **I**	**SI**	-	**E**

	Phase 5	Phase 6	Phase 7	phase 8
Valeur de a	S=*Séquence*(69)	I=*Séquence*(84)	SI=*Séquence*(45)	-=*Séquence*(257)
Valeur de b	84	45	257	257
b appartient au Dico ?	Oui	Oui	Oui	Oui
Valeur de s	T=*Séquence*(84)	-=*Séquence*(45)	-=*Séquence*(45)	IS=*Séquence*(257)
Valeur de t	T	-	I	I
Ajout dans le Dico de	ET	T-	-I	ISI
Ecrire séquence	**T**	**-**	**IS**	**IS**

II.6. Compression par MP3

II.6.1. Les étapes de la compression MP3

Il y a 4 étapes de la compression MP3 décrit comme suit :



-Le signal est découpé en petites sections appelées frames.

-Ces chiffres sont ensuite comparés à des tables de données propres au Codec, qui contiennent des informations sur les modèles psycho-acoustiques.

-Dans le codec mp3, ces modèles sont très avancés, et une grosse partie de la modélisation se fonde sur un principe du nom de Masking.

-Toute information qui correspond au modèle psycho-acoustique est conservée et le reste est rejeté. Ce sont les bases de la compression audio.

-En fonction du bit-rate (nombre de bits par seconde), le codec utilise la taille allouée pour stocker ces données.

-Ceci étant fait, le résultat passe par la compression sans pertes Huffman, qui réduit encore la taille de 10%.

-La technique principale du codec mp3 pour enlever de l'information est en détectant quels sons ne sont pas détectables, ou 'masqués', et qui du coup ne peuvent pas être entendus.

-Ces sons sont ensuite enlevés sans perte audible dans le signal.**[9]**

II.6.2. Psycho-acoustique et sons masqués

L'oreille humaine peut percevoir en théorie toutes les fréquences comprises entre 20 et 20 000 Hz

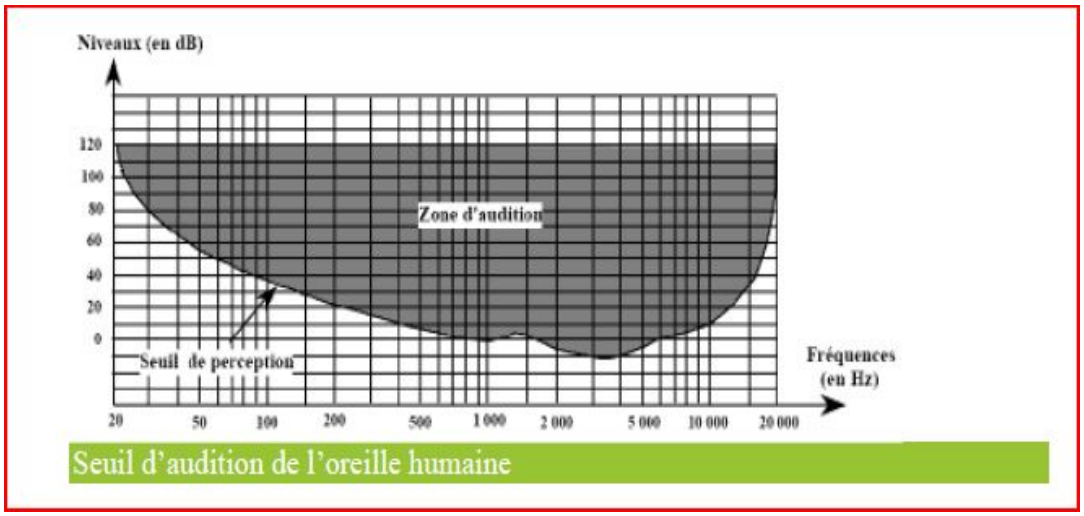


Figure 2.3 : Psycho acoustique et sons

Il y a deux principaux types d'effets de masquage - le masquage fréquentiel et le masquage temporel.

- **Masquage de Fréquences**

On parle de masquage fréquentiel lorsqu'un son faible - qui serait parfaitement audible s'il était émis seul - est masqué parce qu'il se trouve accompagné simultanément par un son fort de fréquence voisine (son « masquant).

Il est inutile de coder les signaux qui sont situés en dessous. Cette courbe de « masquage »
(les variations du seuil d'audition donc) variant à chaque instant en fonction du contenu spectral du signal, c'est donc une véritable analyse en temps réel qui doit être réalisée par les circuits de codage.

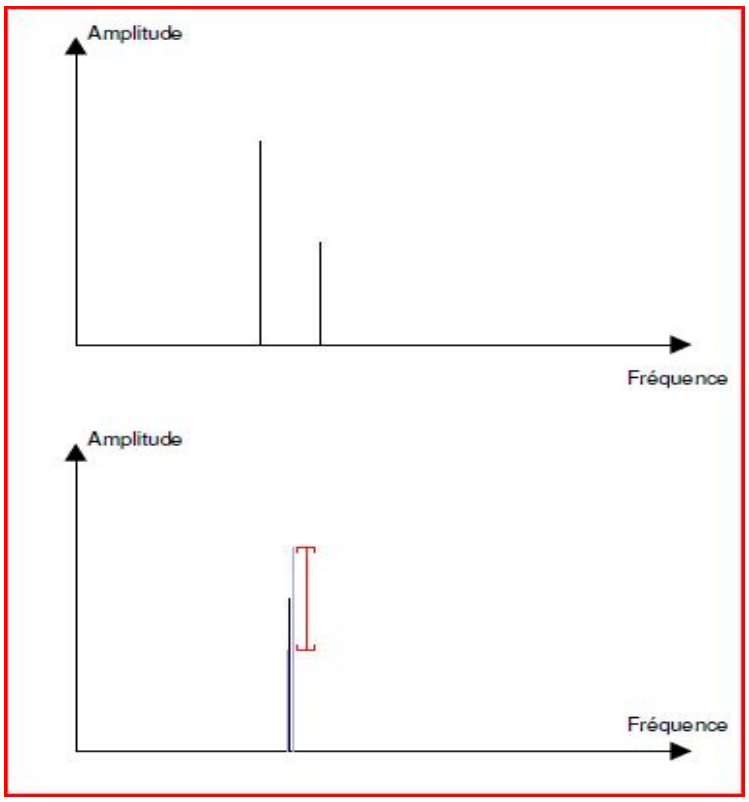


Figure 2.4 : masque frequentiel.

- **Masquage Temporel**

Un son de forte amplitude va être perçu plus longtemps et masquer les sons qui suivent.

Le terme de masquage temporel fait référence au masquage réalisé après l'apparition d'un son masquant de forte intensité, mais également, au masquage

Le terme de temporel indique que le son masquant et le son masqué sont décalés dans le temps par opposition au masquage fréquentiel qui ne concerne que des sons simultanément présents.**[9]**

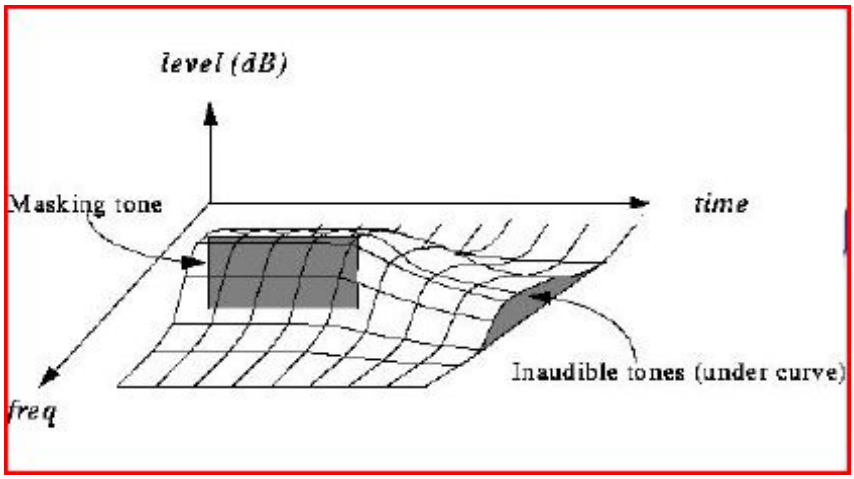


Figure 2.5 : masque temporel

II.7. Conclusion

Dans ce chapitre nous avons présenté la définition de la compression, nous avons distingué deux classe importante des méthodes de compression, c'est la compression de type statistique, la compression de type dictionnaire et la compression MP3.

III.1. Introduction

On recense environ 50 formats audio différents dans le monde, plus ou moins utilisés. Dans un format donné, les fichiers sont déclinés en plusieurs taux de compression ou format de données avec différentes fréquences d'échantillonnage, qui induisent des niveaux de qualité sonore et des tailles de fichier très différents. De manière générale, pour un codec donné, plus le taux de compression est bas, plus le fichier est volumineux, mais plus la qualité sonore du fichier comprimé s'approche de la celle du fichier non comprimé. Par contre, plus le taux de compression est haut, plus le rendu sonore perd de sa qualité (ceci est valable uniquement pour les codecs avec pertes). Dans ce chapitre, On va mentionner quelques extensions de fichiers audio (les plus courantes).

III.2. Les formats audio

III.2.1. Les formats audio compressés avec perte

La compression audio avec perte *(lossy)* se base sur des algorithmes spécialisés pour déterminer quelles transformations simplifient la représentation du son tout en étant perçue quasiment de la même manière par l'oreille humaine. Elle diminue la taille du fichier en éliminant les nuances perçues comme les moins utiles. L'élimination est définitive, créer un fichier dans un format de haute qualité à partir d'un fichier compressé avec perte ne sert strictement à rien.

Le format le plus connu est le MPEG-1/2 Audio Layer 3, dont le suffixe est **.mp3**. Ce format propose une qualité sonore très correcte pour un débit de 128 kbits/s. C'est ce format qui a été massivement utilisé pour transférer les musiques via internet dès la fin des années 1990. Rapidement, des baladeurs avec une mémoire réenregistrable et capables de lire directement ce format sont apparus.**[10]**

III.2.1.1 MP3: MPEG1 audio layer 3

Le MPEG-1/2 Audio Layer 3, plus connu sous son abréviation de MP3, est la spécification sonore du standard MPEG-1, du Moving Picture Experts Group (MPEG). C'est un algorithme de compression audio capable de réduire drastiquement la quantité

de données nécessaire pour restituer de l'audio, mais qui, pour l'auditeur, ressemble à une reproduction du son original non compressé, c'est-à-dire avec perte de qualité sonore significative mais acceptable pour l'oreille humaine.
Il s'agit du format le plus répandu actuellement et il est compatible avec presque tous les logiciels existants.**[11]**

III.2.1.2. WMA: Windows Media Audio

WMA est un format propriétaire de compression audio développé par Microsoft. Le format WMA offre la possibilité de protéger dès l'encodage les fichiers de sortie contre la copie illégale par une technique nommée gestion numérique des droits.
WMA est une alternative au MP3, plus souple mais moins répandu et est uniquement compatible avec les logiciels Microsoft. **[11]**

III.2.1.3. Ogg: Ogg Vorbis

Vorbis est un algorithme de compression et de décompression (codec) audio numérique, sans brevet, ouvert et libre, plus performant en termes de qualité et taux de compression que le format MP3, mais moins populaire que ce dernier. Il reste compatible avec peu d'applications pour le moment. **[11]**

III.2.1.4. AAC: Advanced Audio Coding

Le but de ce format à été d'obtenir un meilleur rapport qualité/compression que le mp3. Adopté par Apple et Real Networks, ce format est lisible sur la plupart des lecteurs audio numériques.Il possède un bon codec destructif et présente une qualité de son meilleure pour un poids plus léger que son équivalent en mp3.
C'est un format concurrent direct du WMA, principalement utilisé pour iPod et iTunes.**[11]**

III.2.1.5. Le mp3pro

La compression mp3Pro se présente comme un successeur avantageux au mp3. En effet, les fichiers générés par un encodeur mp3PRO sont compatibles avec tous les lecteurs mp3. Cependant seuls les lecteurs certifiés mp3PRO bénéficient des avantages supplémentaires de cet encodage.

III.2.2. Les formats audio compressés sans perte

La compression sans perte *(lossless)* signifie qu'on utilise un algorithme tel qu'on peut toujours retrouver les données d'origine. Dans l'absolu, il existe toujours un fichier d'origine tel que l'algorithme ne ferait pas gagner d'espace disque.

Typiquement, la compression sans perte permet de diviser la taille des fichiers par deux ou trois. Elle est relativement peu utilisée, car ce gain est très faible en comparaison de ceux permis par la compression avec perte (ce qui est un gros handicap pour les échanges de fichiers), et assez gourmande en temps de calcul. Aucun standard n'a donc suffisamment convaincu pour devenir universellement lisible. **[10]**

III.2.2.1. Free Lossless Audio Codec

FLAC est un codec libre de compression audio, sans perte. À l'inverse de codecs tels que MP3 ou Vorbis, il n'enlève aucune information du flux audio.
FLAC est approprié pour tous les archivages de données audio, avec le support des métadonnées, image de couverture, ainsi que pour la recherche rapide. FLAC est libre et open source; ne nécessitant pas le paiement de redevances, il est bien supporté par de nombreux logiciels. **[1]**

III.2.2.2. ALAC: Apple Lossless Audio Codec

ALAC est un format d'encodage sans perte (lossless) apparu tardivement (2004). Le format a été développé par Apple pour combler une lacune de la norme MPEG-4, dont la partie consacrée à l'encodage sans perte tarde encore à être finalisée. ALAC est un format propriétaire et qui s'adresse majoritairement aux utilisateurs d'iTunes et de l'iPod.**[1]**

III.2.2.3 L'Atrac 3

Le format Atrac3, dont les fichiers portent l'extension *.omg, est un format propriétaire de Sony. Celui-ci a été créé dans le but de multiplier par deux (LP2) ou par quatre (LP4) la capacité d'un MD. Ce format est uniquement lisible sur les lecteurs MD compatibles MDLP. Aujourd'hui, Sony (et quelques autres marques) commercialisent des nouveaux lecteurs appelés "NetMD". Le protocole NetMD permet de transférer plus rapidement ses fichiers Atrac3 à partir de son PC.
L'atrac3 impose beaucoup de contraintes:

-Impossible de lire les fichiers *.OMG (donc atrac3) sur un autre ordinateur que celui qui l'a encodé.
-Impossibilité d'effectuer plus de 3 transferts d'un même fichier sur un MD (quelque soit le MD).
-L'Atrac3 encrypté est un format entièrement clos.**[8]**

III.2.2.4.FLAC

Le FLAC est un codec de compression audio qui présente l'avantage d'offrir une compression non-destructive (Lossless). Il utilise une méthode de compression qui diminue le débit sonore ainsi que la capacité de stockage sans retirer de données du flux audio originel.
Le taux de compression dépend des informations à traiter mais s'échellonne entre 30 et 60% de réduction. Il possède un rapport de « compression / qualité audio » très intéressant mais manque cruellement de popularité.**[11]**

III.2.3. Les formats audio sans compression

Il existe un format audio non compressé, PCM, qui est généralement stocké sous forme de WAV sur Windows ou sous .Aiff sur Mac OS. WAV et AIFF sont des formats de fichiers flexibles conçus pour stocker plus ou moins n'importe quelle combinaison de taux d'échantillonnage ou de bitrates (bit rate). Ce sont les formats de fichier appropriés pour le stockage et la réalisation d'enregistrements originaux.

III.2.3.1 AIFF: Audio Interchange File Format

AIFF est un format de fichier audionumérique développé par Apple pour stocker les sons sur les ordinateurs de la marque.
Les fichiers AIFF portent généralement l'extension .aif, ou .aiff.
Les données sont codées en PCM big-endian sans compression. Ainsi, une piste CD Audio, codée en 16 bits, stéréo 44,1 kHz aura un bitrate de 1411,2 kbit/s. Il existe néanmoins un format compressé (AIFF-C ou AIFC) qui supporte une compression pouvant aller jusqu'à un rapport 1/6.**[11]**

III.2.3.2. WAV: waveform (forme d'onde)

C'est un format audio «basique» développé par Microsoft et IBM. L'encodage et le décodage sont immédiats car il n'y a aucune compression. Le son peut être mono ou stéréo.

Les fichiers WAV offrent une qualité sonore incomparable et sont compatibles avec tous les lecteurs audio mais la taille des fichiers est très importante. On ne peut pas les utiliser pour la diffusion par Internet.**[11]**

III.2.3.3. CDA: Compact Disc Audio

Le CDA est le format des contenus audio se trouvant sur un CD Audio. C'est un format naturellement non compressé de qualité CD, présentant le désavantage d'être extrêmement lourd. Ce format est le plus souvent compressé afin de le stocker dans un baladeur numérique par exemple. .**[11]**

III.2.3.4. BWF

Le BWF (*Broadcast Wave Format*) est un format audio standard créé par l'European Broadcasting Union en tant que successeur du WAV. Le BWF permet de stocker des métadonnées dans le fichier. Voir *European Broadcasting Union : Spécification du Broadcast Wave Format* (EBU Technical document 3285, juillet 1997). Il s'agit du format d'enregistrement usuel utilisé dans de nombreuses stations de travail audio professionnel de la télévision et du cinéma. Les Fichiers BWF incluent une référence standardisée Timestamp qui permet et facilite la synchronisation avec un élément d'image distincte.

Voici un tableau comparatif entre les différents formats de son :

Format	Taille	Qualité
AAC	3,99 Mo	128 kbps
Vorbis	3,75 Mo	128 kbps
WMA	3,9 Mo	128 kbps
MP3	5,2 Mo	128 kbps
AIFF	43,5 Mo	1411 kbps (CD)
Apple Lossless	25,4 Mo	841 kbps
FLAC	26 Mo	CD
WAV	42,50 Mo	CD

Tableau 3.1 : comparatif entre la taille/qualité des principaux formats audio utilisés.**[1]**

III.3. Le format Wav

Le format PCM : C'est le format de fichier "standard" pour les samples (les enregistrements de sons), car les données sont brutes, c'est-à-dire qu'elles ne sont ni modifiées, ni compressées. Le fichier possède une en-tête de 44 octets (en tout cas en général, voir ci-dessous), permettant de connaître le type du sample: Son format, sa fréquence, le nombre de voies, etc...

Cet en-tête (« header » en anglais) peut, dans certaines variantes du format PCM, avoir une taille supérieure à 44 octets. Mais voyons d'abord le format le plus standard :

Un fichier WAVE est composé de 3 blocs distincts : RIFF, fmt et data…

- Le **bloc RIFF** comporte les informations concernant le type d'en-tête, la taille du fichier et le format du fichier.
- Le **bloc fmt** comporte les spécifications audio : le format audio, le nombre de canaux, la fréquence, le byte rate, le nombre de bits par échantillons…
- Le **bloc data** contient la taille du bloc de données et les données. **[12]**

III.3.1. Format (bloc RIFF)

• On trouve d'abord la mention « RIFF » dans les 4 premiers octets du fichier ($52, $49, $42, $42 en hexadécimal. Ce sont les codes ASCII des lettres R,I,F,F)

• On trouve ensuite la taille TOTALE du fichier codée sous la forme d'un entier long (4 octets, donc)

• On trouve ensuite la mention « WAVE» soit 4 caractères ce qui nous donne en hexadécimal : $57,$41,$56,$45 **[12].**

III.3.2. Descriptif du son (bloc fmt)

Fréquence d'échantillonnage, nombre d'octets par échantillons, etc. Les données qu'il contient ainsi que leurs types et leur ordre correspondent exactement à la structure WAVEFORMAT (cette structure, définie par Microsoft, se retrouve dans pas mal de docs. Elle est aujourd'hui officiellement remplacée par la structure WAVEFORMATEX qui est exactement la même pour les 6 premiers champs, c'est-à-dire tout les champs que l'on retrouve dans l'en-tête du fichier WAVE/PCM).**[12]**

• **Indicateur de zone** : « fmt » soit 4 caractères (le dernier caractère est un espace) ce qui nous donne en hexadécimal : $66,$6D,$74,$20. « fmt » est l'abréviation de « format ». Cet indicateur nous prévient que les informations qui vont suivre concernent le format du son.

• **Taille de la structure WAVEFORMAT** sous la forme d'un entier long (soit 4 octets). La structure WAVEFORMAT comporte 16 octets. On trouve donc, logiquement, la valeur 16 (ou $10 en hexadécimal) stockée à cet endroit.

• **wFormatTag.w** (2 octets) : ce champ contient un code correspondant au format exact de codage des données. Pour les fichiers de type PCM, ce champ contiendra la valeur 1.

• **nChannels.w** (2 octets) : nombre de canaux. On aura la valeur 1 pour les sons mono, 2 pour les sons stéréo, et éventuellement plus pour les sons moins standard.

• **nSamplesPerSec.l** (4 octets) : nombre d'échantillons par seconde.

• **nAvgBytesPerSec.l** (4 octets) : nombre d'octets par secondes. Cette valeur fait double emploi avec les autres valeurs enregistrées mais vous devez la compléter correctement pour être certains que votre fichier sera compatible avec tous les programmes de son. Le nombre d'octets par seconde dépend :

• Du nombre d'échantillons pas seconds

• Du nombre d'octets par échantillon

• Du nombre de canaux

Il se calcul comme suit :

nAvgBytesPerSec= nSamplesPerSec* nBitsPerSample/8* nChannels

(nBitsPerSample est le nombre de BITS par seconde, un octet comporte 8 bits)

• **nBlockAlign.w** (2 octets) : contient la taille totale (en octets) d'un échantillon. Cette valeur fait également double emploi avec les autres valeurs enregistrées mais vous devez aussi la compléter correctement pour être certains que votre fichier sera compatible avec tous les programmes de son. Elle dépend :

• Du nombre d'octets par échantillon

• Du nombre de canaux

Elle se calcul comme suit :

nBlockAlign = nBitsPerSample/8* nChannels*

(nBitsPerSample est le nombre de BITS par seconde, un octet comporte 8 bits)

• **nBitsPerSample.w** (2 octets) : contient le nombre de bits par échantillon (voir note concernant l'amplitude)

Pour finir, le fichier contient les données proprement dites, que l'on appelle, en anglais, les data.

III.3.3. DATA : partie données de fichier wav (bloc data)

• **Indicateur de zone** : «data» soit 4 caractères ce qui nous donne en hexadécimal : $64,$61,$74,$61. Cet indicateur nous prévient que les informations qui vont suivre sont les données proprement dites.

• **Taille des datas** sous la forme d'un entier long (soit 4 octets). Cette taille est le nombre total d'octets des datas. Elle nous permet, par exemple, de calculer la durée du sample en appliquant la formule :

Durée = Taille des datas/ nAvgBytesPerSec

La taille des données doit TOUJOURS être un multiple de nBlockAlign. Lorsqu'une modification d'un son amène une modification de la taille des données, on peut « arrondir » la taille à un multiple de nBlockAlign à l'aide des deux opérations suivantes : **[12]**

NouvelleTaille = NouvelleTaille/ nBlockAlign

NouvelleTaille = NouvelleTaille* nBlockAlign

III.3.4. Structure générale d'un fichier wav

Offset (décimal)	offset (hexa)	nom	longueur (oct.)	**description**
0	00h	rID	4	contient "RIFF"
4	04h	rLen	4	longueur du fichier
8	08h	wID	4	contient "WAVE"

Le **Format** Chunk:

Offset (décimal)	offset (hexa)	nom	longueur (octet)	**description**
12	0Ch	fId	4	contient "fmt " ("fmt espace")
16	10h	fLen	4	Longueur du Chunck
20	14h	wFormatTag	2	**format** (1 = Microsoft Pulse Code Modulation PCM)
22	16h	nChannels	2	nombre de canaux (1=mono, 2=stéréo)
24	18h	nSamplesPerSec	4	fréquence d'échantillonage (en Hz)
28	1Ch	nAvgBytesPerSec	4	= nChannels * nSamplesPerSec * (nBitsPerSample/8)
32	20h	nBlockAlign	2	= nChannels * (nBitsPerSample / 8)
34	22h	nBitsPerSample	2	longueur d'un échantillon en bits (8, 16, 24 ou 32)

Le WAVE Data Chunk:

Offset (décimal)	offset (hexa)	nom	longueur (octet)	**description**
36	24h	dId	4	contient "data"
40	28h	dLen	4	longueur du chunck dData (en octets)
44 et plus	2Ch	dData	dLen	les données du son échantillonné

III.4. Caractéristique de fichier wav

III.4.1. L'amplitude

Un sample est composé d'une courbe continue ayant une valeur bi-polaire (signée) . Le 1er élément d'un son est l'amplitude: C'est le point le plus élevé (et le plus bas) de la courbe. Plus l'amplitude est élevée, plus le son est fort, bruyant. L'unité de grandeur de l'amplitude est le décibel (dB). "C'est une mesure logarithmique donnant le degré d'amplification d'une vibration." Nous n'irons pas plus loin dans la description du décibel, puisque ce n'est pas indispensable dans la programmation. L'amplitude est digitalisée avec l'ADC de la carte son. Par exemple, en 8 bits, l'amplitude possède une résolution de 256 valeurs. En 16 bits, 65536 valeurs, etc... Il existe aussi le 24 et 32 bits. Pour le moment, toutefois, SoudEditor ne supporte pas les données 24 bits qui sont très embêtantes à manipuler et qui sont très rarement utilisées. Plus la résolution est élevée, plus l'échantillon est proche du son original. Dans la figure 4.1, l'amplitude digitalisée est illustrée en vert. En 8 bits, la valeur de l'amplitude est non signée, et en 16 bits, l'amplitude est signée. **[12]**

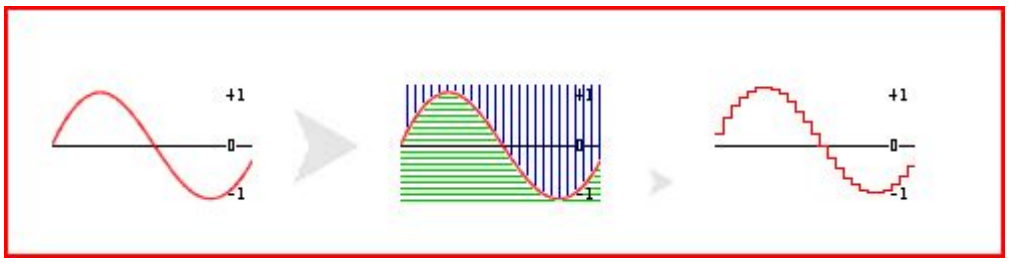


Figure 3.1. Amplitude, fréquence

III.4.2. La fréquence

En bleu (figure 3.1), c'est la fréquence d'échantillonnage, le nombre de valeurs définissant l'amplitude pour une seconde d'enregistrement. Ainsi 44100 Hz signifie 44100 échantillons pour une seconde de son mémorisé. Plus la fréquence d'échantillonnage est élevée, meilleure est la qualité du sample, plus les données digitales sont proches de l'original.

III.4.3. Le débit

On peut calculer le "débit" (ko/s) d'un sample avec ces paramètres: L'amplitude (format 8 ou 16 bits), le mode (mono ou stéréo) et la fréquence. Par exemple, en 8 bits mono 44100Hz, cela nous donne pour une seconde d'enregistrement: 44100 octets (1 octet=8 bits). En stéréo, c'est le double. En 16 bits stéréo, c'est le quadruple. Ainsi, avec la qualité du CD audio (16 bits stéréo 44,1 kHz), on obtient 176400 octets par seconde. A partir de la taille du fichier (donnée par fileSize), et les caractéristiques du sample, il est facile de calculer le temps total en secondes (ou ms) de lecture d'un sample.**[12]**

III.4.4. L'ordre des données

Après les 44 octets de l'en-tête, viennent les données. Les données ont un ordre bien défini. Dans le cas d'un sample **8 bits mono,** c'est **1** seul octet par sample, donc les données se suivent normalement. Pour un sample **8 bits stéréo** , ce sont **2** octets par sample, l'octet de la voie de gauche, puis l'octet de la voie de droite: L,R, L,R, L,R, etc... Dans le cas d'un sample **16 bits mono** , ce sont **2** octets par sample, l'octet de poids faible, puis l'octet de poids fort. Par exemple pour une donnée qui vaudrait 15000, nous aurions: $98 $3A (les octets sont toujours inversés). Dans le cas d'un sample **16 bits stéréo** , ce sont **4** octets par sample; Deux octets pour la voie de gauche, et deux pour la

voie de droite: L,L,R,R, L,L,R,R, L,L,R,R, etc...

Le format des données : Le sample **8 bits** (mono ou stéréo) possède des données non-signées, c'est-à-dire que le point (l'amplitude) le plus bas vaut **zéro** , le point du milieu vaut **127** , et le point le plus haut vaut **255** .

Pour pouvoir travailler sur ces données 8 bits, (par exemple pour modifier le volume du sample, ou le mixer avec un autre, etc...) les données 8 bits doivent subir une petite manipulation afin d'être présentées de la même façon que les données 16 ou 32 bits.**[12]**

Nom	Nombre d'octets	Type	Endian	Description
Bloc de déclaration d'un fichier au format WAVE				
Chunk ID	4	4 char	big	4 caractères constants RIFF : 0x52494646
Chunk Size	4	int 32	little	Taille du fichier - 8 octets des deux premiers blocs
Format	4	4 char	big	4 caractères constants WAVE : codé 0x57415645
Bloc décrivant le format audio				
SubChunk 1 ID	4	4 char	big	4 caractères constants "fmt " : codé 0x666d7420
SubChunk 1 Size	4	int 32	little	Taille du sous-bloc "SubCHunk 1" - 8 octets, vaut 16 pour le PCM
Audio Format	2	int 16	little	Type de compression audio, 1 pour PCM
Num Channels	2	int 16	little	Nombre de canaux (1 = Mono, 2 = Stéréo, ...)
Sample Rate (Frequence)	4	int 32	little	Fréquence d'échantillonnage (nombre d'échantillons par seconde)
Byte Rate (BytePerSec)	4	int 32	little	Nombre d'octets par seconde = Frequence * BitsPerSample/8 * NumChannels
Block Align (BytePerSample)	2	int 16	little	Nombre d'octets par échantillon (tous canaux confondus) = Nombre de canaux * Nombre d'octets par échantillons
Bits per Sample	2	int 16	little	Nombre de bits par échantillon
Bloc des données				
SubChunk 2 ID	4	4 char	big	4 caractères constants : "data" : 0x64617461
SubChunk 2 Size	4	int 32	little	Taille des données = NbCannaux * NbSamples * BitsPerSample/8 = taille du fichier - 44 octets (taille de l'en-tête, rappelez-vous-en !)
Datas (données)	FileSize - 44	...	little	Les données !

Exemple détaillé de la structure des fichiers audio wav .

III.4.5. Fichier wav en hexadécimal

Voila une représentation détaillée d'un fichier wav avec les trois blocs (riff, fmt et data).

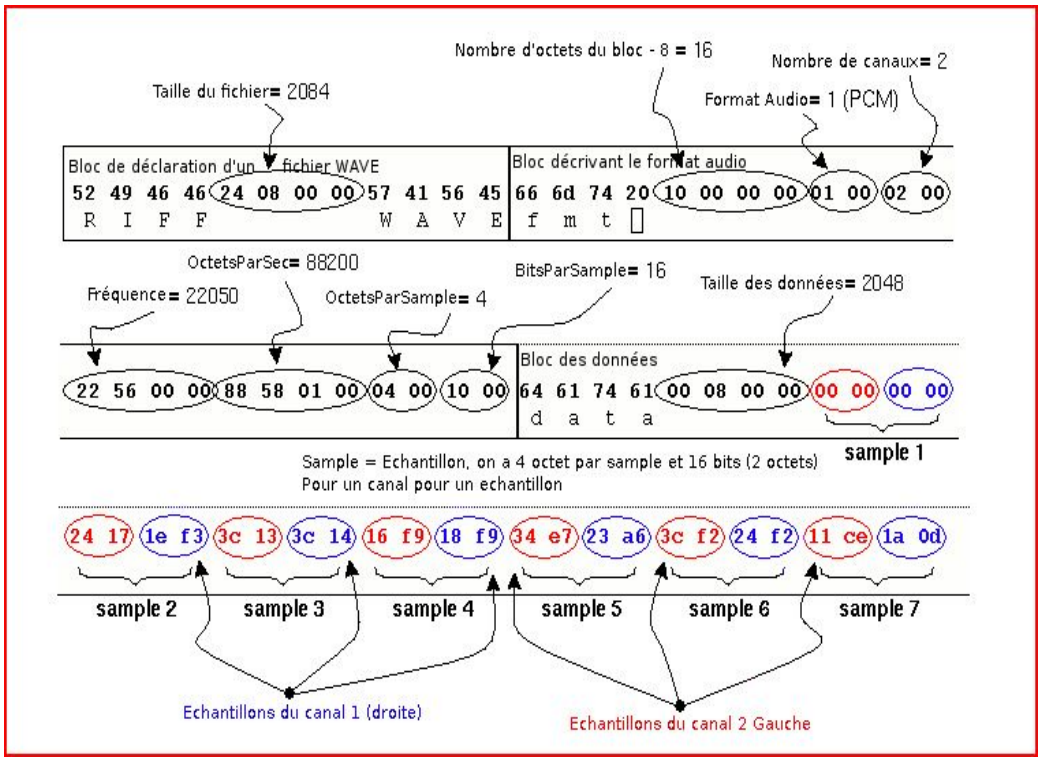


Figure 3.2. Fichier wav en hexadecimal

III.5. Conclusion

Dans ce chapitre, nous avons présenté une description des différents formats du fichier audio, et précisément une description détaillée sur la structure du fichier WAVE. L'étude et l'analyse de format de fichier Wave sera l'objectif de notre application de compression.

IV.1. Introduction

Dans ce chapitre, Nous avons réalisés une étude comparative en basant sur le taux et temps de compression des différents formats de codage des fichiers wave. Le résultat obtenu de la compression des fichiers wave sera aussi comparé avec les logiciels de compression standard comme goldwave pour estimer la performance et la qualité de notre résultat de compression.

Notre travail s'effectué sur un ordinateur Intel (R) cadencé à 1.67 GHz et sous une RAM de 2.00 Go et un système d'exploitation vista.

Paramètres de comparaison

La comparaison va se faire sur plusieurs points et suivant certains critères :

- Temps de compression
- Taux de compression
- L'amplitude

IV.2. Présentation de l'application

Langage utilisé

C++ Builder est le nouvel environnement de développement basé sur C++ proposé par Borland. Fort du succès de Delphi, Borland a repris la philosophie, l'interface et la bibliothèque de composants visuels de ce dernier pour l'adapter depuis le langage Pascal Orienté Objet vers C++ répondant ainsi à une large faction de programmeurs peu enclins à l'utilisation du Pascal qu'ils jugent quelque peu dépassé.**[15]**

L'interface de l'application

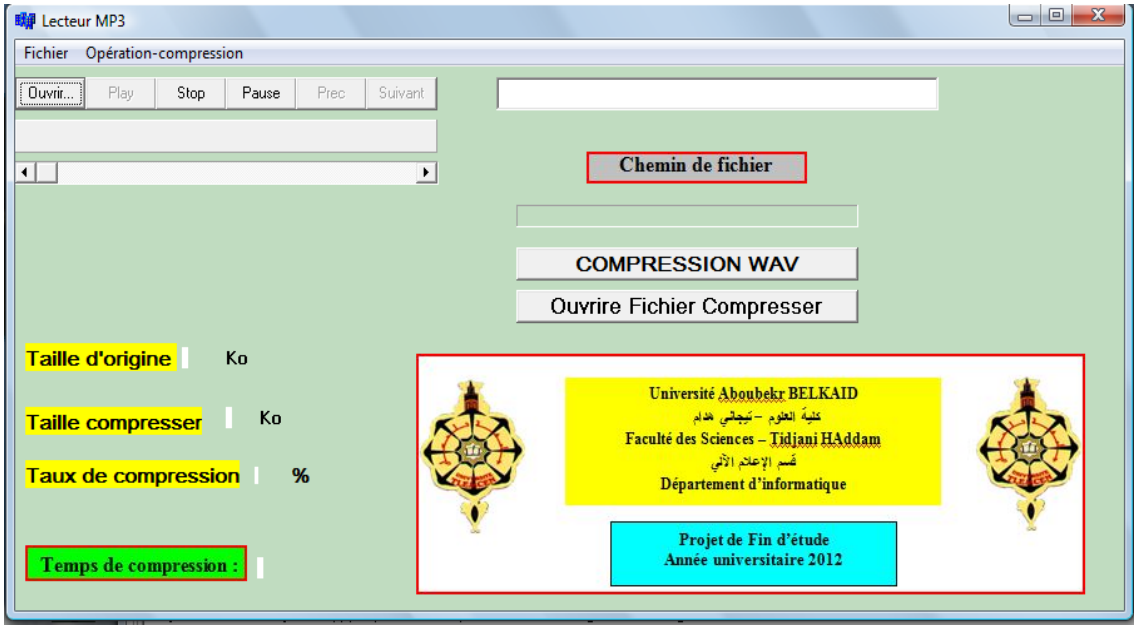


Figure 4.1 : interface de l'application

- Pour l'ouverture d'un fichier son : appuyer sur bouton Ouvrir :

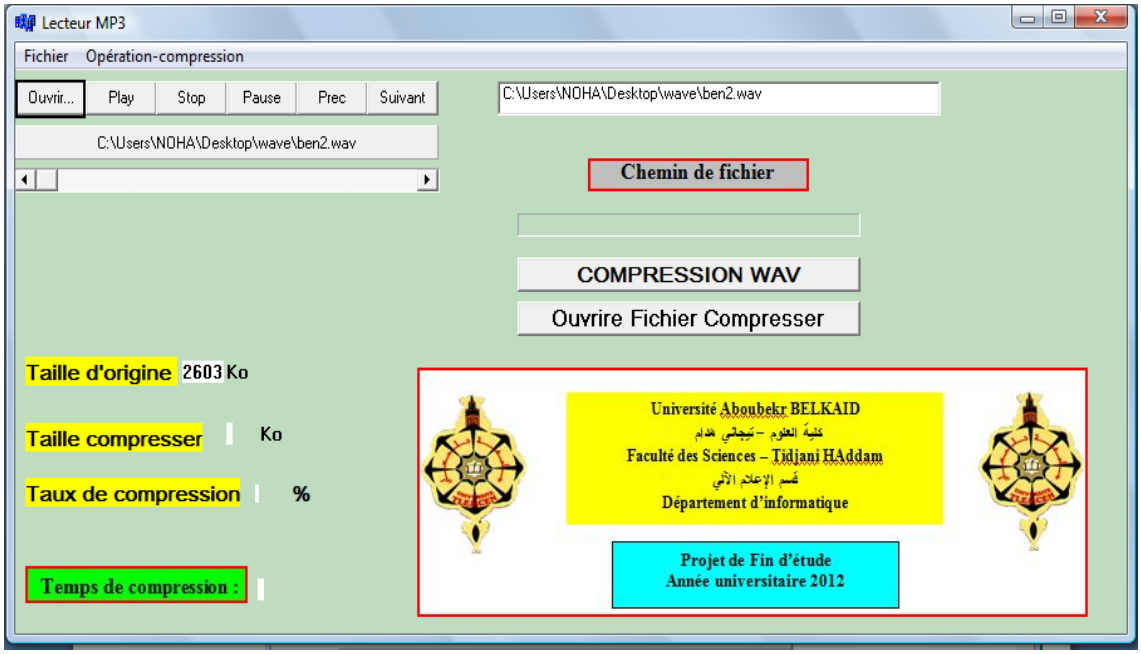


Figure 4.2 : ouvrir un fichier son

❖ Pour **jouer** le fichier son :

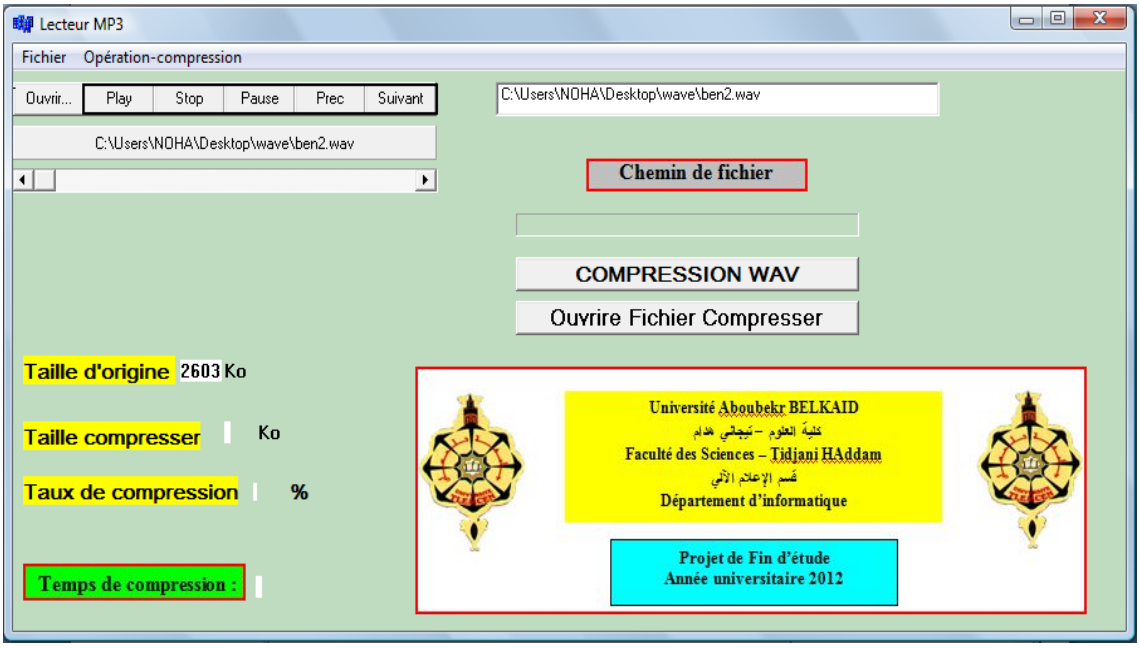


Figure 4.3 : jouer un fichier son.

IV.3 .Les résultats obtenus

A .Résultats du temps de compression de différents fichiers son :

	Temps de compression de chaque fichier		
	Fichier 1	**Fichier 2**	**Fichier 3**
Taille original(Ko)	1055	1889	2603
Taille compressé(Ko)	286	707	708
Temps (seconde)	**4.431**	8.86	10.686

Tableau 4.1 : temps de compression des différents fichiers son.

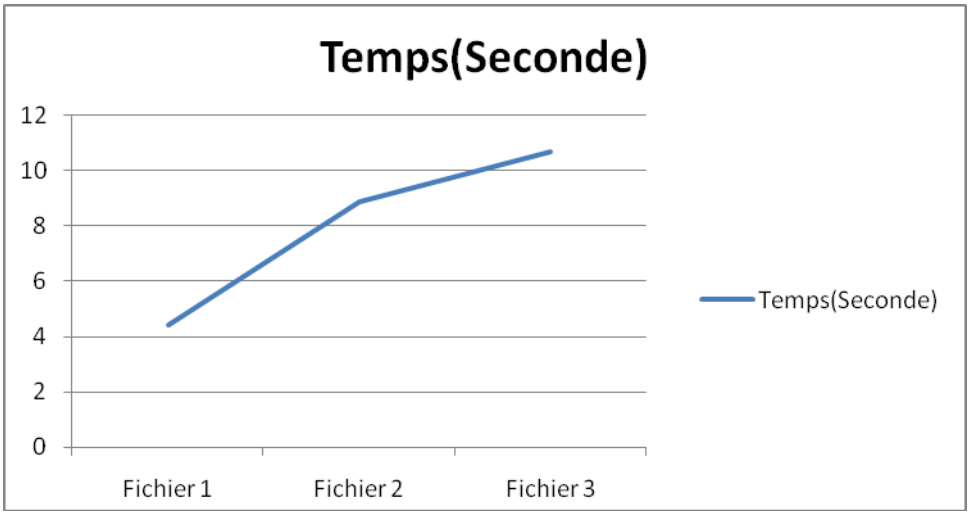


Figure 4.4 : le temps de compression de différents fichiers son.

Remarque :

- Le temps de compression est changé par le changement de la taille de fichier à compresser.
- Dans les trois fichiers, nous avons remarqués que lorsque en utilisant un fichier de petit taille (fichier 1) le temps de compression est plus rapide que les autres fichiers. Cela implique que le temps de compression est augmenté selon la taille de fichier à compréssé.

B .Résultats de taux de compression de son par les différents formats de codage

	Taille de fichier après la compression avec différents codages					
	pcm-8bit		**pcm-16bit**		**pcm--24bit**	
	Mono	**stéréo**	**mono**	**Stéréo**	**Mono**	**stéréo**
Taille original	1301 Ko	2603Ko	2603 Ko	5207 Ko	3905 Ko	7810 Ko
Taille compressé	353 Ko	353 Ko	708 Ko	708 Ko	106 Ko	**106 ko**
Taux compression	72%	86%	72%	86%	72%	86%

Tableau 4.2 : Taille de compression du son par les différents formats de codages.

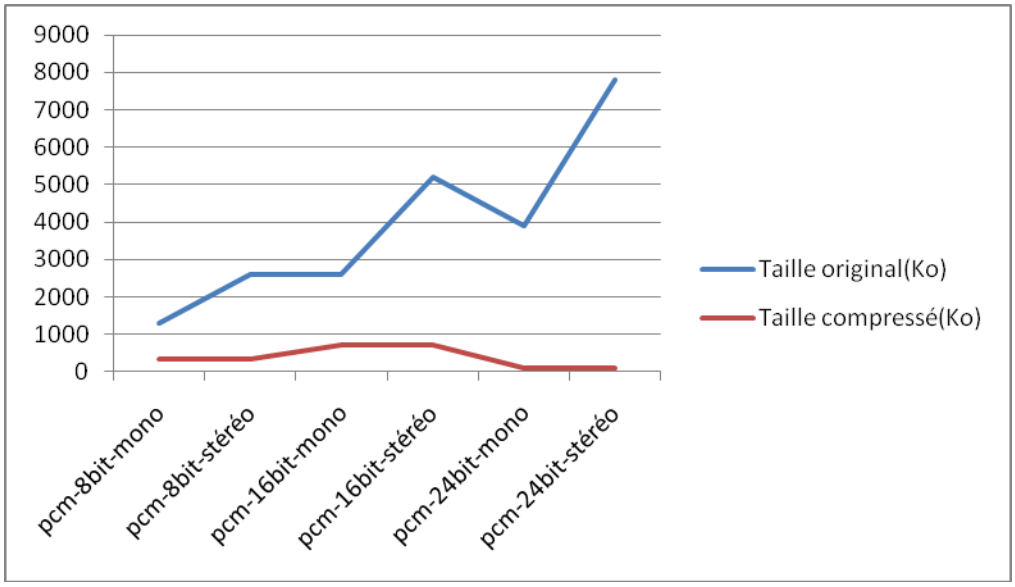


Figure 4.5 : la taille d'origine et la taille de compression d'un fichier son codé en différents formats.

Remarque :

- Nous avons observés d'après les résultats obtenus que le taux de compression des fichiers son wave avec le codage stéréo (8bits, 16 bits et 24 bits) est meilleur que le taux de compression des fichiers son codé en mono (cette résultat de compression est claire dans les trois fichiers)

IV.4. Présentation de logiciel GoldWave

GoldWave est un logiciel spécialement conçu pour l'édition des fichiers audio. Il dispose de tous les outils qu'il faut pour couper, copier ou coller des fichiers audio. On peut aussi effectuer des enregistrements en toute simplicité.**[16]**

Principales fonctionnalités

-Édition : Goldwave permet de couper, coller, insérer, remplacer ou écraser des fichiers audio. Il est possible de créér une transition entre deux fichiers collés pour éviter les changements brusques. Toutes les manipulations peuvent se faire en quelques clics.Il est compatible avec la majorité des formats audio.

- Lecture et enregistrement : la lecture audio est facilitée. On peut avoir une vue en tempsréel du fichier lu sous forme d'onde ou d'analyse spectrale. L'enregistrement est possible du moment que le son passe par la carte appropriée. On peut par exemple

enregistrer à partir d'un microphone, d'un appariel branché sur l'entrée ligne et bien plus encore.

- **Effets :** pour améliorer ou modifier le son, une douzaine d'effets est proposée. On trouve un équaliseur, une réverbération, un "flanger" et bien plus encore.Il embarque aussi des réducteurs de bruits pour les enregistrements sur vinyle et cassette. Des outils pour l'analyse des fréquences sont aussi disponibles. **[16]**

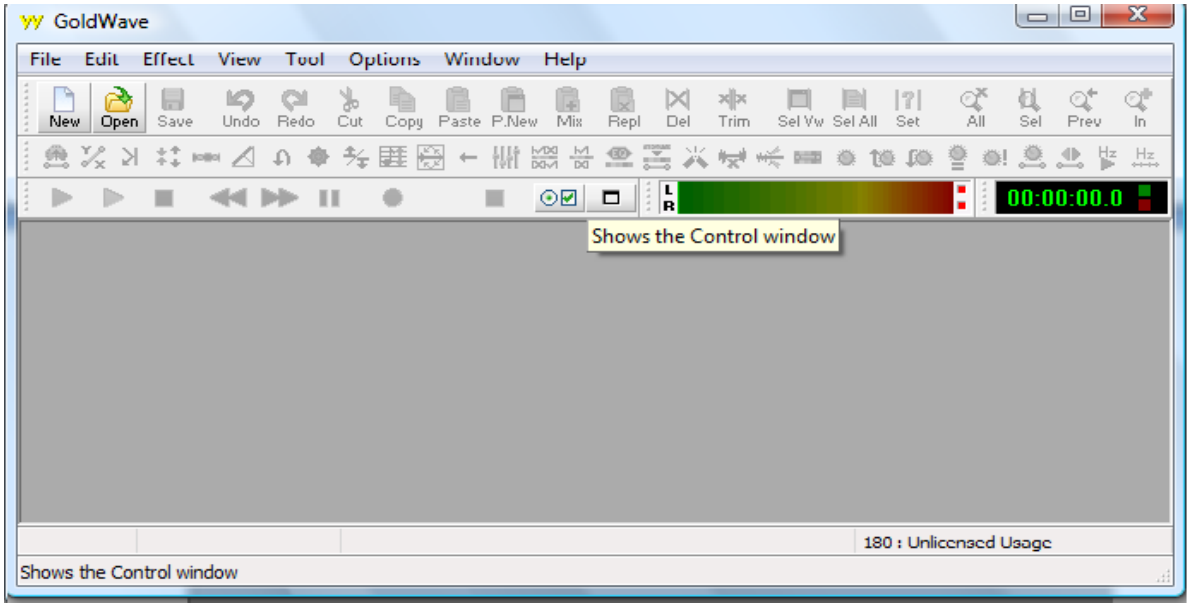


Figure 4.6 : l'interface de logiciel GOLDEWAVE.

IV.4.1 .Résultats de la qualité de compression de son par les différents formats de codage de son par GOLDWAVE :

	Taille de fichier après la compression avec différents codages					
	pcm-8bit		**pcm-16bit**		**pcm--24bit**	
	Mono	**stéréo**	**mono**	**Stéréo**	**Mono**	**stéréo**
Taille original	1301 Ko	2603Ko	2603 Ko	5207 Ko	3905 Ko	7810 Ko
Taille compressé	471 Ko	471 Ko	471 Ko	471 Ko	471 Ko	471 ko
Taux compression	63%	81%	81%	90%	87%	93%

Tableau 4.3 : taille de compression du son par les différents formats de codage.

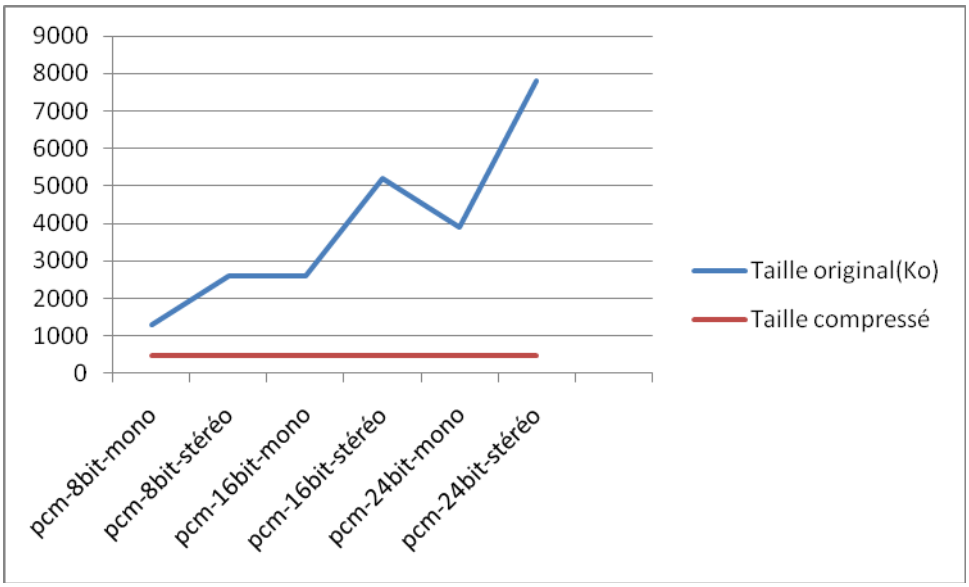


Figure 4.7 : la taille d'origine et la taille de compression d'un fichier son codé en différents formats par Goldwave. .

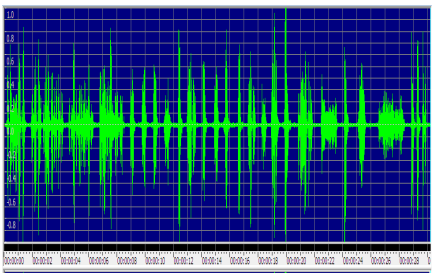
Remarque :

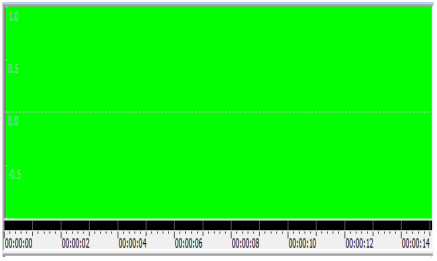
- lorsque on fait la conversion d'un son de type wave à un fichier son de type MP3 par goldwave, nous remarquant que la taille de ce fichier avec différents attributs a été réduits et équivalente dans tout les cas (quelques soit le codage utilisé).

- A partir de la comparaison des résultats obtenus de Goldwave et notre logiciel nous concluons que la qualité de compression par Goldwave est bonne et meilleur par rapport à notre résultat obtenu par notre application surtout pour les fichiers de grand de taille

IV.4.2. L'amplitude de chaque fichier Compressé

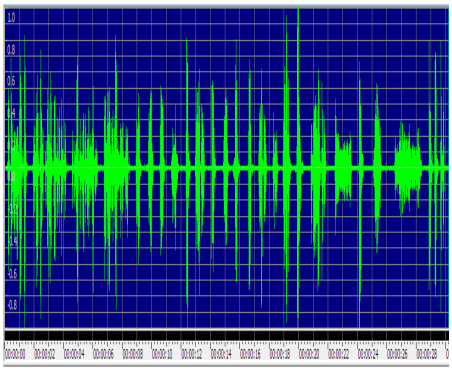
IV.4.2.1. L'amplitude de fichier codé en pcm-unsigned-8bit-mono compressé à MP3 :

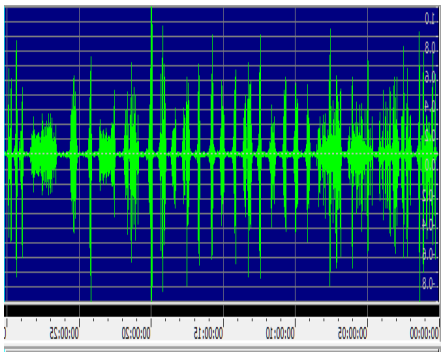




Avant la compression ***après la compression***

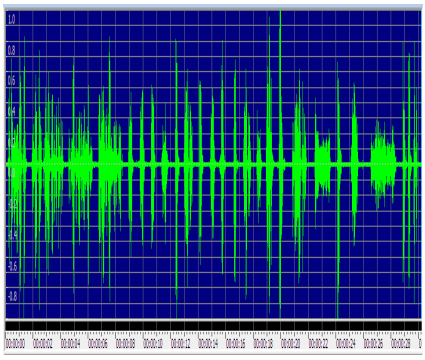
IV.4.2.2. L'amplitude de fichier codé en pcm-signed-16bit-mono compressé à MP3 :

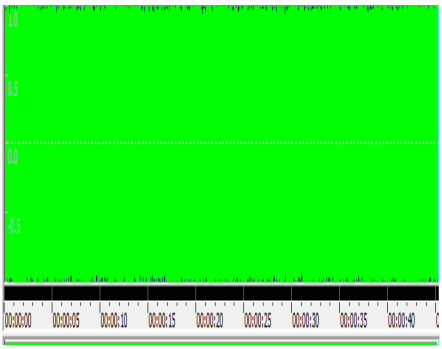




Avant la compression ***après la compression***

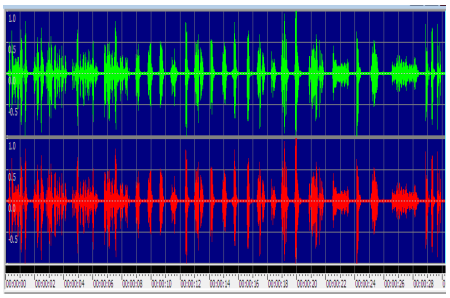
IV.4.2.3. L'amplitude de fichier codé en pcm-signed-24bit-mono compressé à MP3 :

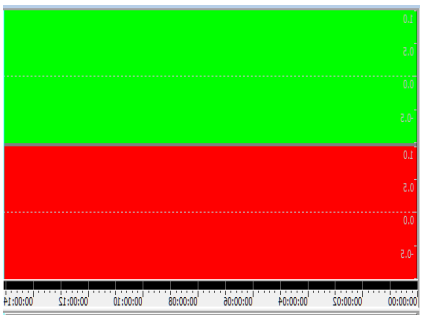




Avant la compression
après la compression

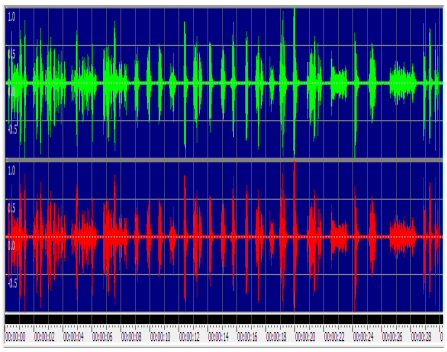
IV .4.2.4. L'amplitude de fichier codé en pcm-unsigned-8bit-stéréo compressé à MP3 :

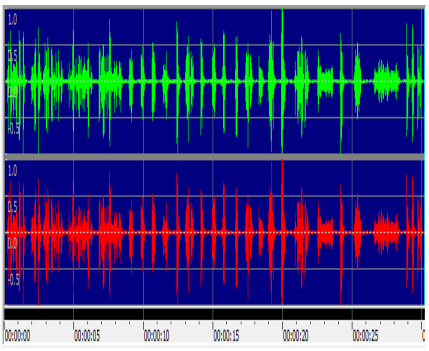




Avant la compression *après la compression*

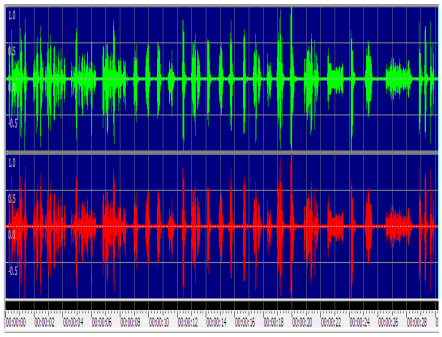
IV.4.2.5. L'amplitude de fichier codé en pcm-signed-16bit-stéréo compressé à MP3 :

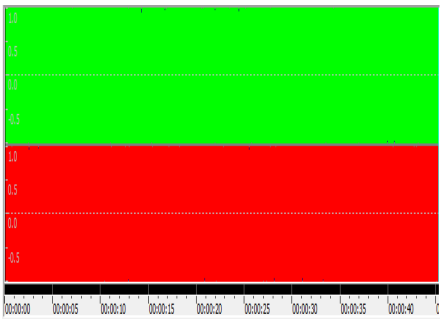




Avant la compression *après la compression*

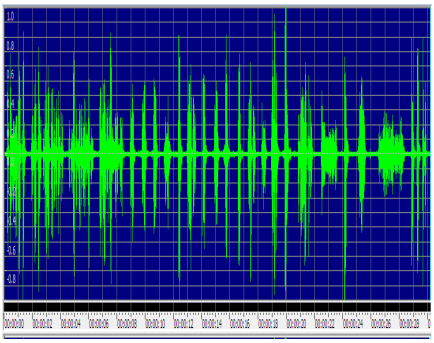
IV.4.2.6. L'amplitude de fichier codé en pcm-signed-24bit-stéréo compressé à MP3 :

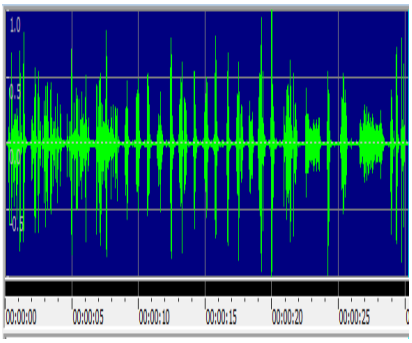




Avant la compression *après la compression*

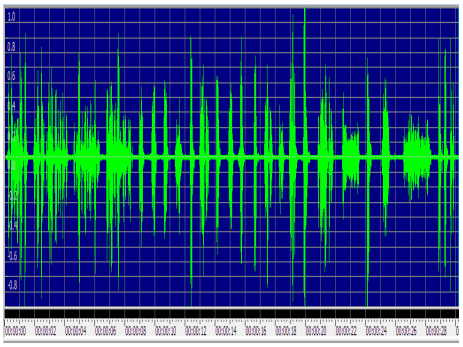
IV.4.2.7. L'amplitude de fichier codé en pcm-unsigned-8bit-mono converti à MP3 par GoldWave **:**

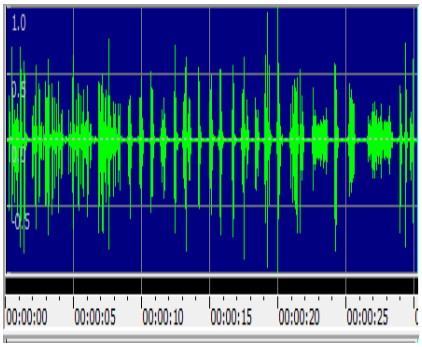




Avant la compression ***après la compression***

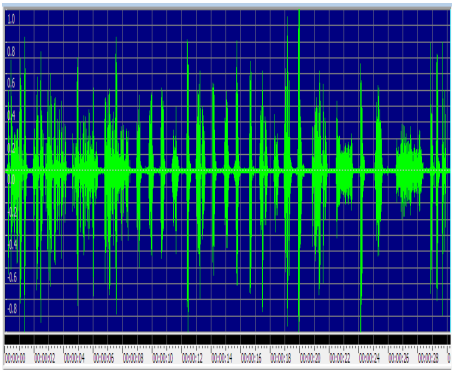
IV.2.8. L'amplitude de fichier codé en pcm-signed-16bit-mono converti à MP3 par GoldWave **:**

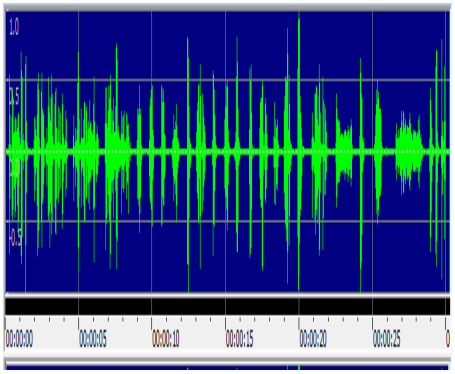




Avant la compression ***après la compression***

IV.4.2.9. L'amplitude de fichier codé en pcm-unsigned-24bit-mono converti à MP3 par GoldWave **:**





Avant la compression ***après la compression***

Remarque :

Nous avons remarqués que l'amplitude de chaque fichier compressé avec différents attributs est changés dans notre logiciel de compression et même l'amplitude de chaque fichier converti par le logiciel GoldWave est changé.

IV.5. Conclusion

Dans ce chapitre, nous avons présentés les résultats obtenus de notre application qui est basé sur la compression des fichiers son de type wave avec les différents format de codage en fichier MP3 selon masquage fréquentiel et temporel.

D'après les résultats obtenus et la comparaison selon le taux de compression, que les fichiers son en codage stéréo donne une meilleur réduction de la taille de fichier compressé par rapport au codage mono.

Plus, nous remarquons aussi que le goldwave (application de traitement de son) donne une meilleure qualité de compression avec un taux élevé par rapport à notre application.

Conclusion générale

Dans ce mémoire, nous avons présenté les notions de base sur le traitement du son ensuit nous avons présentés quelque formats des fichiers audio et surtout le format wave avec ses caractéristiques, puis nous avons décrit les différents algorithmes de compression des fichiers son. Nous avons aussi parlé sur les différents formats audio existés sur l'Internet.

Dans le domaine du traitement de son, il existe plusieurs travaux et recherche pour la compression des fichiers audio.

Notre objectif principal dans ce projet de fin d'étude consiste à la compression d'un fichier en format wave à un fichier en format MP3 basé sur le masquage fréquentiel et temporelle, et en particulier la compression d'un fichier wave codé avec différents attributs. Nous avons réalisés une étude comparative en basant sur le taux et temps de compression des différents formats de codage des fichiers wave. Le résultat obtenu de la compression des fichiers wave sera aussi comparé avec les logiciels de compression standard comme goldwave pour estimer la performance et la qualité de notre résultat de compression.

Comme perspectives de notre travail, nous souhaitons d'implémenté d'autre algorithme de compression comme RLE et complété les étapes de l'algorithme compression MP3 pour avoir une meilleur taux de compression par rapport aux logiciels standard de compression de son.

Finalement, nous voudrons réitérer mon désir que ce projet soit utile pour l'avenir. Nous souhaitons qu'il y a une poursuite de notre travail dans le domaine de la compression de son.

[1] : MARCHAL Mickaël,Fichiers Audio et Vidéo, site web :http://www.lesitedemika.org .

[4] :J. TERRA Terrasson , Les outils du multimédia, éditions Armand Colin, 1992.

[5] :Guénael Launay, Les bases de l'ordinateur, Planète Radio,page=5

[6] :Affès Amel et All , La multimédia,4 ème lettre.

[8] : Romain Matuszak ,Clément Follet , etude et comparatif codecs audios avec pertes , compte Rendu,2007-2008.

[10] : Frederic Koriche , Cours Théorie de l'Information, Compression: partie IV, Université Montpellier II, France.

[12] : AMARA Khadija, BENZENINI Hadjira, La cryptographie à clé sécrète appliqué sur des fichiers sons, PFE MASTER, 2012.

[14] : HERNÁNDEZ José, Algorithmes d' Acquisition, Compression et Restitution de la parole à vitesse variable. Étude et Mise en place, PFE, ENSEA (École Nationale Supérieure de l'Électronique et de ses Applications Cergy-Pontoise (Paris)), Avril 1995.

[15] : ISIMA , Introduction à C++ Builder ,OURS BLANC DES CARPATHES ™, 1988-1999 .

Sites web

[2] :http://raphael.isdant.free.fr/traitement_numerique/3traitement_numerique_du_son.pdf

[3] :http://www.pfast.fr/IMG/pdf/CriteresEvalFormatsCompressionCicmHd3d.pdf.

dernier mis a jour: 11/10/2007.

[7] :Compression:http://fr.wikipedia.org/wiki/Compression_de_donn%C3%A9es

Archive : http://fr.wikipedia.org/wiki/Archive_%28informatique%29

[9] : http://perso.usthb.dz/~slarabi/CoursSM/Chap3-Audio.pdf

[11] : http://www.unil.ch/webdav/site/guichet/users/fducrest/public/La_compression_audio.pdf

[13] : http://freewares-tutos.blogspot.fr/2011/09/theme-30-les-10-meilleurs-logiciels.html.

dernier mis a jour: 26 sept. 2011.

[16] : http://www.commentcamarche.net/download/telecharger-34066117-goldwave.HTML.

Dernière modification le mercredi 22 août 2012 à 14:27:29

Annexe A

1 .Bloc de déclaration d'un fichier au format WAVE

[Bloc de déclaration d'un fichier au format WAVE]

[Chunk RIFF début]

FileTypeBlocID (4 octets) : Constante «RIFF» (0x52,0x49,0x46,0x46)

FileSize (4 octets) : Taille du fichier moins 8 octets

FileFormatID (4 octets) : Format = «WAVE» (0x57,0x41,0x56,0x45)

[Chunk RIFF fin]

[Bloc décrivant le format audio]

[Chunk fmt début]

FormatBlocID (4 octets) : Identifiant «fmt » (0x66,0x6D, 0x74,0x20)

BlocSize (4 octets) : Nombre d'octets du bloc - 8 (0x10)

AudioFormat (2 octets) : Format du stockage dans le fichier (1: PCM, ...)

NbrCanaux (2 octets) : Nombre de canaux (de 1 à 6, cf. ci-dessous)

Frequence (4 octets) : Fréquence d'échantillonnage (en Hertz) [Valeurs standardisées : 11025, 22050 , 44100 et éventuellement 48000 et 96000]

BytePerSec (4 octets) : Nombre d'octets à lire par seconde (i.e., Frequence * BytePerBloc).

BytePerBloc (2 octets) : Nombre d'octets par bloc d'échantillonnage (i.e., tous canaux confondus : Nb rCanaux * BitsPerSample/8).

BitsPerSample (2 octets) : Nombre de bits utilisées pour le codage de chaque échantillon (8, 16, 24).

[Chunk fmt fin]

[Bloc des données]

[Chunk data début]

DataBlocID (4 octets) : Constante «data» (0x64,0x61,0x74,0x61)

DataSize (4 octets) : Nombre d'octets des données (i.e. "Data[]", i.e. taille_du_fichier - taille_de_l'entête (qui fait 44 octets normalement).

DATAS[] : [Octets du Sample 1 du Canal 1] [Octets du Sample 1 du Canal 2] [Octets du Sample 2 du C anal 1] [Octets du Sample 2 du Canal 2] (si on est en stéréo : 2 cannaux)

5 [Chunk data fin]

2.Structure de riff

```
1. struct RIFF {
2. char ChunkID[4];    // contient les lettres "RIFF" pour indiquer que le fichier
   est codé selon la norme RIFF
3. word ChunkSize;       // taille du fichier entier en octets (sans compter les 8 o
   ctets de ce champ et le champ précédent CunkID
4. char Format[4];       // correspond au format du fichier donc ici, contient les l
   ettres "WAVE" car fichier est au format wave
5  } RIFF;
```

3.Structure fmt

```
1. struct fmt {
2. char Subchunk1ID[4];    // contient les lettres "fmt " pour indiquer les données à suiv
   re décrivent le format des données audio
3. dword Subchunk1Size;    // taille en octet des données à suivre (qui suivent cette vari
   able) 16 Pour un fichier PCM
4. short AudioFormat;       // format de compression (une valeur autre que 1 indique un
   e compression)
5. short NumChannels;       // nombre de canaux: Mono = 1, Stereo = 2, etc..
6. dword SampleRate;       // fréquence d'échantillonage, ex 44100, 44800 (nombre d'é
   chantillons par secondes)
7. dword ByteRate;          // nombre d'octects par secondes
8. short Blockalign;       // nombre d'octects pour coder un échantillon
9. short BitsPerSample;    // nombre de bits pour coder un échantillon
0  } fmt;
```

4.Structure data

```
1. struct data {
2. char Subchunk2ID[4];    // contient les lettres "data" pour indiquer que les do
   nnées à suivre sont les données audio (les échantillons et)
3. dword Subchunk2Size;    // taille des données audio (nombre total d'octets co
   dant les données audio)
4. short *data;            // données audio... les échantillons
5  } data;
```

5.Lecture de fichier wave

```
long filesize(FILE* stream) {
   long curpos, length;
   curpos = ftell(stream);
   fseek(stream, 0L, SEEK_END);
   length = ftell(stream);
   fseek(stream, curpos, SEEK_SET);
   return length;
}

int check_file(FILE *fl, short *fstart) {
 unsigned long dlng;
 int mmd, lng, i;
 char ch;
 fseek(fl, 0, SEEK_SET);
 if (fgetc(fl)!='R') return(1);
 if (fgetc(fl)!='I') return(1);
 if (fgetc(fl)!='F') return(1);
 if (fgetc(fl)!='F') return(1);
 fseek(fl, 8, SEEK_SET);
 if (fgetc(fl)!='W') return(1);
 if (fgetc(fl)!='A') return(1);
 if (fgetc(fl)!='V') return(1);
 if (fgetc(fl)!='E') return(1);
 if (fgetc(fl)!='f') return(1);
 if (fgetc(fl)!='m') return(1);
 if (fgetc(fl)!='t') return(1);
 if (fgetc(fl)!=' ') return(1);
 fseek(fl, 34, SEEK_SET);
 fread(&lng, 2, 1, fl);

 printf("\nbps  : %d",lng);
//if (lng!=8) return (1);
 fseek(fl, 22, SEEK_SET);
 fread(&mmd, 2, 1, fl);
 printf("\nMode : %d",mmd);
 //if (mmd!=1) return (1);
 i=0;
 while((fgetc(fl)!='d')&&(!feof(fl))) i++;
 if (fgetc(fl)!='a') return (1);
 if (fgetc(fl)!='t') return (1);
 if (fgetc(fl)!='a') return (1);
 fread(&dlng, 4, 1, fl);
 *fstart = ftell(fl);
 return 0;
}
```

6 .Logiciels d’édition audio

Il existe plusieurs logiciels pour éditer facilement les fichiers audio. Ils sont classés sans ordre de préférence, à chacun de faire son choix. La plupart sont en anglais, mais sont relativement simples à utiliser. Dans ce chapitre nous avons mentionné quelques logiciels.

6.1. Logiciels d’édition du son

6.1.1. Ocenaudio

est un logiciel d'origine brésilienne (mais disponible en anglais) qui fonctionne aussi bien sur Windows (XP/Vista/7), que sur Mac OS X et Linux et qui permet d'éditer et d'analyser facilement des fichiers audio (Mp3, WAV, FLAC, OGG, etc) .**[13]**

Figure 1 :l’interface de **Ocenaudio**

6.1..2.Audacity

est sans doute l'éditeur audio gratuit le plus connu. Fonctionnant lui aussi sur Windows, Mac OS X, GNU/Linux, etc, il est disponible en français et permet de :

- Enregistrer en direct.
- Convertir vos disques et cassettes sur support numérique.

- Editer des fichiers audio Ogg Vorbis, MP3 et WAV
- Couper, copier, coller et assembler des extraits sonores
- Modifier la vitesse ou la hauteur d'un enregistrement

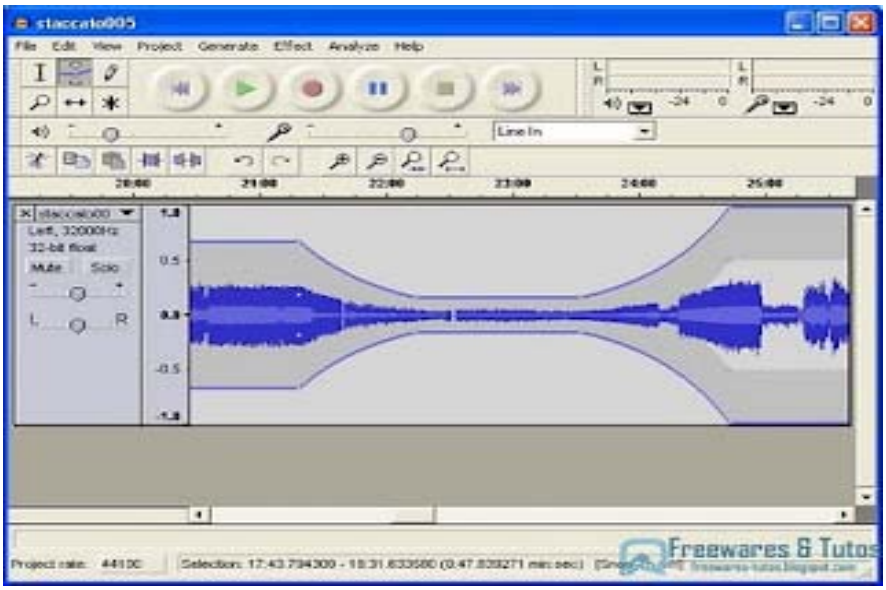


Figure 2 :l'interface de **Audacity**

6.1.3. Free Audio Editor

est un outil audio complet (disponible en français) qui permet de gérer toute votre musique à partir de la même interface. Vous pourrez enregistrer des sons provenant de votre lecteur CD, microphone, lecteur de vinyl, disque dur, etc. et les sauvegarder en fichiers audio et plus encore.**[13]**

Figure 3: interface de **Free Audio Editor**

6.1.4.Music Editor Free

est un outil puissant qui permet de modifier vos fichiers de musique comme si vous éditiez des fichiers texte !. Il dispose de nombreuses fonctions.

.

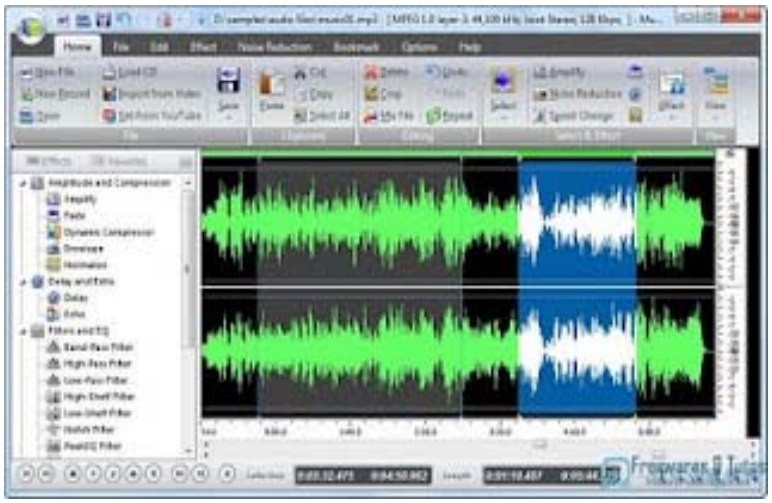


Figure 4: interface de **Music Editor Free**

6.1.5. Free MP3 Cutter and Editor

Est un outil extrêmement simple et utile pour des tâches courantes d'édition de fichiers MP3. Il affiche une représentation graphique de votre fichier audio pour sélectionner et éditer facilement une partie du MP3.

Figure 5 : interface de **Free MP3 Cutter and Editor**

6.1.6.Power Sound Editor Free

Est un logiciel d'édition audio qui intègre de nombreuses fonctions de retouche audio. Il permet, entre autres d'enregistrer toute source sonore, d'éditer, mélanger avec d'autres sources sonores, d'y ajouter des effets (Reverb, Chorus, écho), etc.

Figure 6: interface de **Power Sound Editor Free**

6.1.7.Shuangs Audio Editor

est un éditeur audio qui permet de couper des fichiers WAV, MP3 et WMA, mais aussi d'ajuster le volume du fichier audio entier, ou d'une partie seulement, et de lui appliquer des effets (fade in/out).

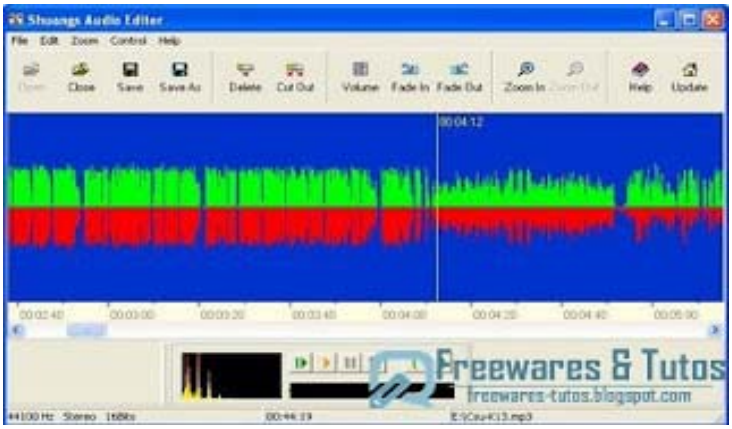


Figure 7 : interface de **Shuangs Audio Editor**

6.1.8.Wavosaur

Est un logiciel d'édition audio-numérique qui permet d'enregistrer et éditer les fichiers sons. Disponible en français, il fonctionne uniquement sous Windows 98, XP et Vista.

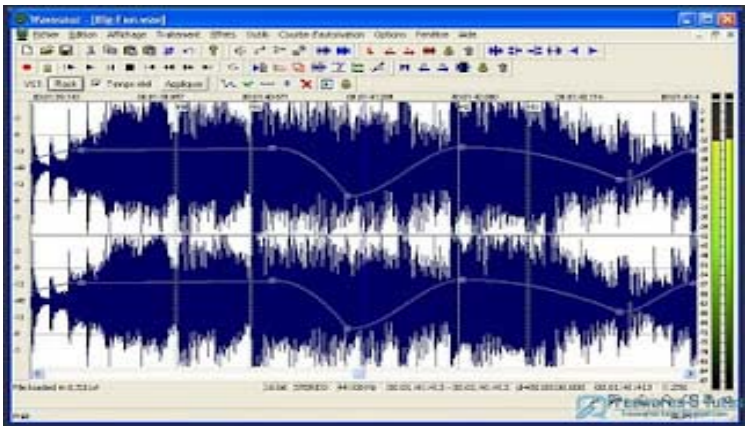


Figure 8 : interface de **Wavosaur**

6.1.9.Mp3DirectCut

Est un logiciel simple d'emploi qui permet de couper, copier, coller, modifier le volume des mp3 sans avoir besoin de les décompresser. Il peut aussi enregistrer n'importe quelle source sonore grâce à son enregistreur intégré.

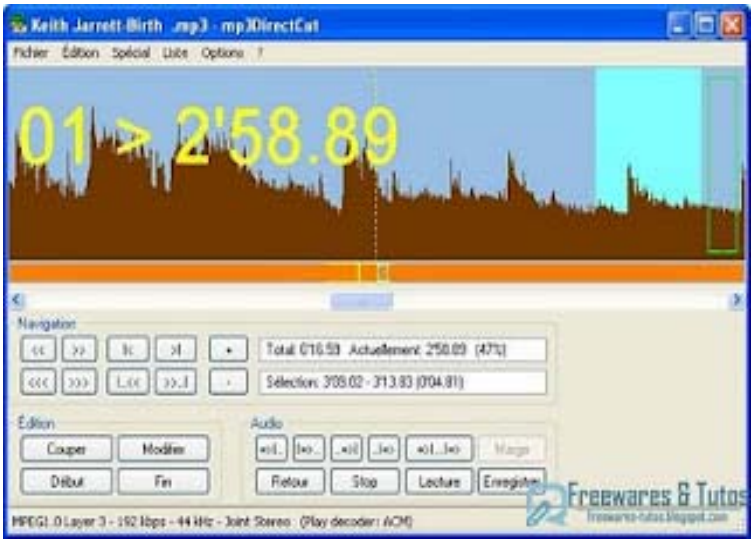


Figure 9 :l'interface de **Mp3DirectCut**

6.1.10. Wave Editor

est un logiciel de montage audio pour Windows qui intègre des fonctions d'édition puissantes et conviviales qui conviennent en particulier aux débutants pour exécuter les tâches basiques telles que : couper, copier, coller et supprimer des parties de l'enregistrement.**[13]**

Figure 10 : interface de **Wave Editor**

Résumé

Ce travail de projet de fin d'étude s'intéresse à une étude comparative sur la compression d'un fichier son. La compression est l'action utilisée pour réduire la taille physique d'un bloc d'information.. Il existe plusieurs algorithmes pour la compression comme HUFFMAN, …etc. Nous avons fait la compression d'un fichier son de format WAVE non compressé à un fichier MP3 compressé avec différent format de codage, différent frame et quelque soit le fichier mono où stéréo. A partir des résultas obtenus, on conclu que le taux de compression varie selon de la taille et les attributs de fichier compressés.

Mots clés : la compression, fichier son, WAVE, MP3, codage, taux de compression.

Abstract

Our job is about a comparative study of sound file compression. Compression it means reduce a physical volume of an information bloc.. A lot of algorithms are used for compression the file like HUFFMAN…etc. In this application, we do the compression of WAVE format of sound that it not compress to a MP3 format that is compressed with different attribute, different sample rate and if the file is mono or stereo. From different simulation, we conclude that the rate of compression is varied from the volume and the attributes of file that will be compress.

Key Words: compression, sound file, WAVE, MP3, attribute, rate of compression.

تـلخيص:

عملنا هذا يتضمن دراسة مقارنة بخصوص عملية ضغط ملف صوتي.هذه العملية تسمح بتقليص الحجم المادي للملف الصوتي. هناك العديد من الخوارزميات المتعلقة بالضغط مثل هوفمان، ... الخ. قمنـــا بضغط ملف صوتي غير مضغوط WAVEملف مضغوط MP3 بحيث ناخذ الملف صوتي بترامیز مختلفة، معدل العينة مختلفة وبحيث يكون ستريو او أحادي. من هذه المحاكاة، وجدنا أن نسبة الضغط تختلف تبعا للحجم وسمات الملف الذي نود ضغطه.

الكلمات المفتاحية: ضغط، ملف صوتي، WAVE، MP3، الترميز، معدل الضغط.