I- a l'attention de l'utilisateur : installation et execution
Ce logiciel est écrit en langage Java2 (version 1.4.2), et ne fonctionnera que si votre machine est equipee d'un environnement java2 (Java2 Runtime Environnment, jre...) correspondant a cette version, ou plus recente : j2re1.4.. ou plus recent (ca devrait etre le cas pour la plupart des systemes windows)
si vous n'en disposez pas, vous reporter en II
1.1-decompresser le fichier Dictee...zip sous un repertoire quelconque (pas forcement en mode administrateur), avec winzip(windows), ou Ark (linux)
1.2- lancement
sous windows => *executer cmd*, ca ouvre une console Dos
sous la console dos, vous placer sur le repertoire *dist* du repertoire que vous avez cree, a l'aide de la commande *cd* (pour chdir, ou change directory), qui permet de changer de repertoire
=> entrer
cd C:\choseBidule..\TrucMachin...\DicteeMusicale\dist

ensuite executer le programme en entrant
java -jar AnaSon.jar
, ca devrait fonctionner

sous linux ouvrir un terminal,
passer sous le repertoire dist
cd /home/TrucMachinChose/DicteeMusicale/dist
executer le code java avec la version 1.4.2 de java telechargee chez sun, genre
/usr/local/jre-1.4.2-version278.6.98/bin/jav -jar ./DicteeMusicale.jar

1.3- pour commencer
Preciser que vous bossez sous windows si c'est le cas (je suis un utilisateur de linux, et c'est le choix par defaut du sys. D'exploitation)

ensuite l'ancer l'accordeur, et choisir l'application accordeur (vous taire quand il evalue le niveaux de bruit), puis chanter, siffler, jouer d'un instrument (et verifier que le niveau de signal est coherent avec le bruit produit=> plus vous chantez fort, plus le niveau de signal est important...)
Si le niveau reste tres faible :regler le mixer de votre carte son, de facon a activer le micro...

Si le niveau est incoherent, ca vient probablement du codage des donnees=>
-arreter l'accordeur, en fermant la fenetre,
-dans la liste prevuee a cet effet(Codage des donnees), choisir *Little endian*, et relancer l'accordeur
si ca ne fonctionne toujours pas
-dans la liste prevuee a cet effet, choisir *laisser le sys d'exploitation decider*, et relancer l'accordeur
si ca ne fonctionne pas, je ne peux plus rien pour vous, supprimer le repertoire et au revoir...

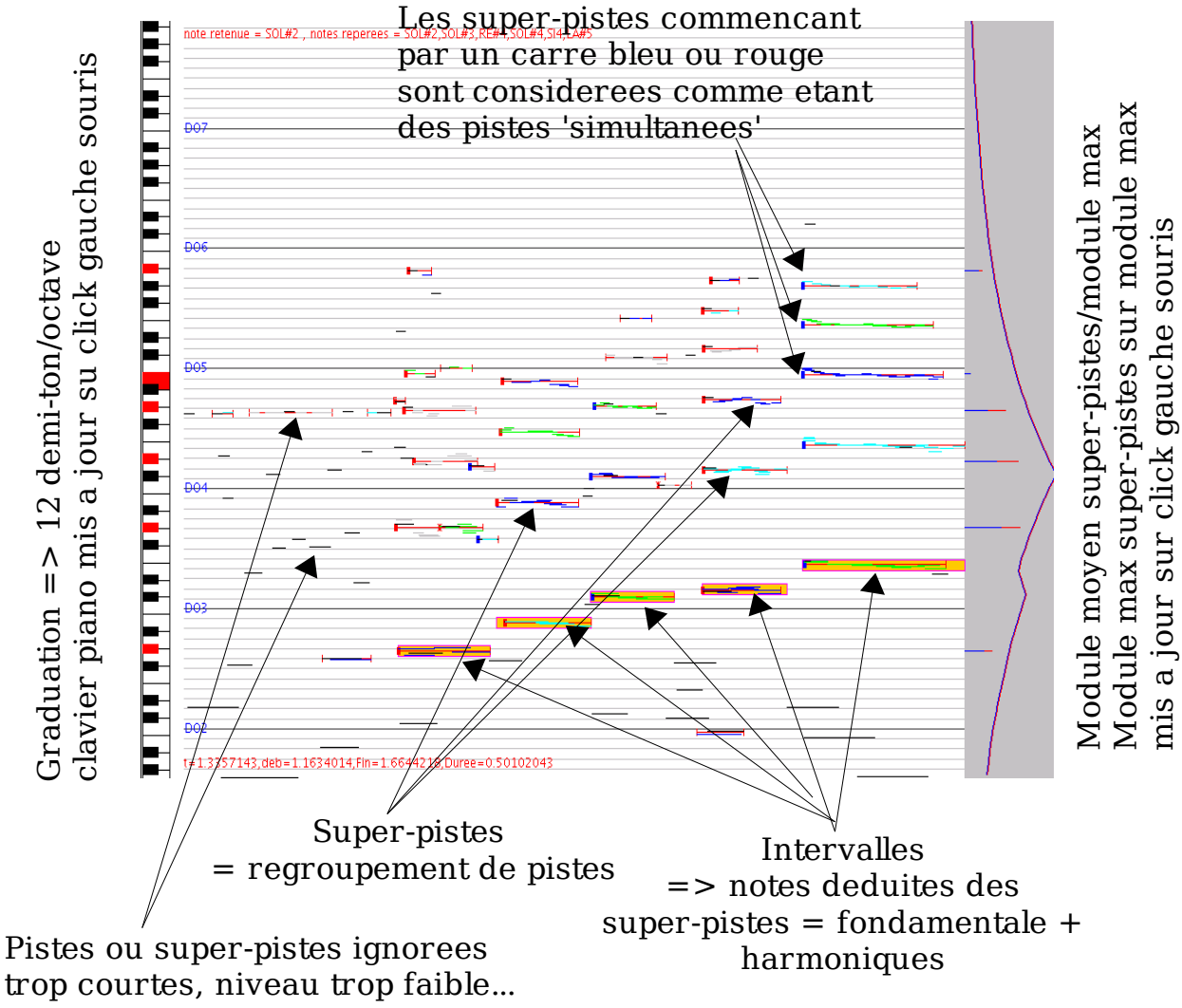
II- A L'ATTENTION DU DEVELOPPEUR POTENTIEL

Ce logiciel a ete developpe sous Netbeans version 5.0 pour java 1.4, systeme de developpement libre et gratuit, telechargeable chez SUN, de façon a ce que n'importe quel informaticien interesse par la musique puisse librement le reprendre pour y ajouter ses propres fonctionnalites....

pour telecharger cette version de netbeans , vous pouvez rechercher "j2sdk", sous google ou autre moteur de recherche, ca vous mettra directement sur la page de telechargement de Sun. Ensuite telecharger et installer j2sdk with netbeans pour java 1.4, correspondant a votre systeme d'exploitation, et l'installer.
Sous Netbeans=> Project=>open=> DicteeMusicale..., vous avez alors acces a tout le source du programme, en tant que developpeur....

Breve explication de la structure, et fonctions cle=> comment ca marche

la reconnaissance des notes est operee en 3 phases,
Phase 1 : detection de pistes frequentielles (partie traitement du signal)
Phase 2 et 3 => bidouille perso, pas de competences particulieres dans ce domaine
Phase 2 : regroupement des pistes en super pistes
Phase 3 : regroupement des super pistes et de leurs harmoniques en notes de musique
Voir dessin ci-dessous (resultat de traduit son-> notes, ou analyse fichier wav), en l'occurence j'ai chante DO RE MI FA SOL



selon l'endroit sur lequel vous voulez intervenir, les competences a mettre en oeuvre sont assez differentes, voir ci-apres

PHASE 1: TRAITEMENT DU SIGNAL ET DETECTION DES PISTES FREQUENTIELLES
(routine ContexteFlts.INIT_CONTEXTE() , routine Cle **ClcDetect.execute()**)
(*seul un developpeur ayant une connaissance suffisante du traitement de signal sera en mesure de comprendre le jargon suivant...*)
le traitement du signal a appliquer est fixe par la routine ContexteFlts.INIT_CONTEXTE() , qui definit une serie de traitements (filtrages, demodulations, decimation) a appliquer sur le buffer son, avec la routine **ContexteFlts.init_traitements()**
ces traitements sont ensuite appliques pour chaque nouveau buffer sont, en appelant la routine **ContexteFlts.serie_de_traitements()**

le buffer son(buffer de short= entier 16 bits) est analyse de la maniere suivante
(on note *sig*(*t*)= sortie du buffer a l'instant *t*)

On le passe au travers d'une succession de demodulateurs, centrees sur des frequences centrales f_i regulierement reparties sur une echelle logarithmique, et fixees par les parametres suivants :
Nombre de filtre / Octave => Contexte.NB_FILTRES
FrequenceMinimum => Contexte.F_MINI
Nombre de filtre / Octave => Contexte.F_MAXI
FrequenceD'echantillonnage => Contexte.F_ECH
Pour chacun des demodulateurs , de frequence fi:
-La sortie de chaque demodulateur est a valeur complexe(
-Re(t)= sig(t).cos(wi.t), Im(t)=sig(t).sin(wi.t)) . wi=2.pi.fi
-ce signal (complexe)est passe au travers d'un passe-bas, de sortie complexe y(t)
-On evalue ensuite l'amplitude instantanee, la frequence instantanee, le retard de groupe, et la derivee de la frequence instantanee / a la frequence de demodulation f0, du signal y(t)

- on lance la routine de detection : **ClcDetect.execute()**, qui est la **routine Cle de l'algorithme de detection** ...
  *ORIGINALITE : l'idee de base de la detection de frequences appliquee ici consiste a reperer les composantes **suffisamment periodiques** du signal, pour estimer que l'on peut les analyser par tr de Fourier <=> dans ce cas et uniquement dansce cas, l**es amplitudes et frequences instantanees sont alors considerees comme des informations valides,** et sont regroupees dans des "pistes", qui sont des objets de Type InfoTrack, qui indiquent*
  la frequence de la piste
  le debut et la fin de la piste
  le module au carre (amplitude au carre) de la piste
  ... tout ce qu'on peut avoir envie de connaître sur la piste ....
  Le reperage des pistes est base sur le postulat suivant :

UNE PISTE SERA CONSIDEREE SUFFISAMMENT PERIODIQUE A L'INSTANT t <=> LA DERIVEE DE LA FREQUENCE INSTANTANEE Finst(t) DE LA SORTIE DU FILTRE, PAR RAPPORT A LA FREQUENCE DE DEMODULATION FI, EST A PEU PRES EGALE A 1 A CETTE FREQUENCE (<=> si on avait demodule avec Fi+DeltaF au lieu de Fi , on aurait alors mesure une frequence instantanee Fins(t)+DeltaF, au lieu de Finst(t).c'est seulement a cette condition que l'on peut considerer que Finst(t) est une information exploitable).
CE POSTULAT SE TRADUIT PAR LE CRITERE SUIVANT DANS L'ALGORITME
il faut que 0,5 < Dfins(t)/ DFI<1,5 avant toute tentative d'exploitation de la sortie d'un filtre. Comme ce critere est tres simple a calculer, ca permet d'elaborer un algorithme d'extraction des pistes, qui fonctionne en temps reel sans difficulte (j'ai essaye le detecteur avec 24 filtres / Octaves sur 6 octaves, avec une frequence d'echantillonnage de 44 KHZ, sur ma machine a 800MHZ, et il fonctionne sans aucun probleme=> 0,2s pour traiter 1s de signal)

Conclusion : Les personnes qui voudraient modifier la partie '' Traitement du signal'' sont invitees a analyser, et a modifier la routine : ContexteFlts. ClcDetect.execute(), qui est la routine cle de detection

PHASE 2: REGROUPEMENT DES PISTES EN "SUPER PISTES"
(cette partie est accessible a toute personne qui a des notions sur ce qu'est une frequence, un temps, et une amplitude)

c'est la routine **ContexteFlts.regroupe_notes(LinkedList infos_track)** qui regroupe les pistes elementaires en pistes plus grandes

Nécessité du groupement

- plusieurs demodulateurs(filtres) ont pu detecter la meme piste frequentielle
-une piste reelle de frequence variable peut avoir cesse d'etre detectee par un filtre, pour etre detectee par le suivant

Remarque : la methode de regroupement releve plus de la recette du chou farci aux aromates que de la science (autant je suis satisfait par mes algos de base en traitement du signal, autant je suis conscient que l'on peut franchement ameliorer les choses a ce niveau, en bref c'est de la bidouille malsaine).

En entree, on dispose d'une liste d'objets de type InfoTrack, decrivant chacune des pistes elementaires(frequence, amplitude, debut,duree, filtre utilise), detectees par les routines de traitement de signal.

En sortie, certaines pistes ont ete regroupees .

Une piste regroupee se repere par le fait que le champ *is_a_super_track*
est alors egal a *true*
dans le cas ou ce champ est a *true*, les champs suivants contiennent l'information relative au groupe de pistes

f_reelle_super_track (frequence reelle moyenne du groupe de pistes)
sum_mod_square_super_track (somme des energies du groupe de pistes)
mean_mod_square_super_track(amplitude au carre moyenne du groupe de pistes)
n0_reel_super_track(numero d'echantillon de debut du groupe de pistes. Remarque: temps debut=n0/F_ECH)
n1_reel_super_track(numero d'echantillon de fin du groupe de pistes. Remarque: temps fin=n1/F_ECH)

Methodologie de groupement (j'ai un peu honte...)

REGLE 1 : Si deux pistes se recouvrent temporellement et ont des frequences suffisament proches, on les regroupe :
les parametres mis en jeu sont (valeurs par defaut):
SEUIL_FREQ_NOTES_SYNCHRONES_EN_DEMI_TON=0.5
COEFF_RECOUVREMENT_NOTES_SYNCHRONES=0.75

<=> si 2 pistes ont un ecart en frequence inferieur a 0.5 demi-tons, et qu'elles se recouvrent pendant plus de 75% de la duree de la plus courte, alors on les regroupe

REGLE 2 : Si deux pistes sont temporellement proches et ont des frequences suffisament proches, on les regroupe :
les parametres mis en jeu sont (valeurs par defaut):
SEUIL_FREQ_NOTES_ASYNCHRONES_EN_DEMI_TON 0.5
SEUIL_TEMPS_NOTES_ASYNCHRONES 0.05
<=> si 2 pistes ont un ecart en frequence inferieur a 0.5 demi-tons, et qu'elles se recouvrent pendant plus de0,05secondes, alors on les regroupe

MODIFICATIONS
vous pouvez a loisir remplacer ces regles par d'autres moins subjectives, en modifiant la routine **ContexteFlts.compute_super_tracks(LinkedList infos_tracks)**

PHASE 3: DETECTION DES NOTES DE MUSIQUE A PARTIR DES "SUPER PISTES"
Partie accessible aux musiciens ayant une bonne notion de frequence fondamentale et harmoniques....

dans cette partie, on utilise les informations groupees (les objets InfoTrack dont le champ is_a_super_track est egal a *true) pour en deduire les notes de musique...*
l'idee generale est de reconstruire la frequence fondamentale de la note, depuis les super-pistes qui contiennent simultanement le fondamental et les harmoniques, voire pas de fondamental et uniquement des harmoniques...

Avertissement : Difficultes de la detection (c'est vraiment un probleme pas facile)
1*- Une note jouee precedemment peut continuer a sonner lorsque vous en jouez une nouvelle, et vous tromper sur le fondamental*
2- A cause des interferences, Une piste precedemment jouee peut disparaître au moment ou vous jouez une nouvelle note, et reapparaitre juste apres...=> elle va apparaître comme une nouvelle piste, et vous tromper sur le fondamental.
3- si vous etes comme-moi, vous chantez et vous sifflez faux : c'est catastrophique, chaque fois que je chante, je pars un demi-ton en dessous ou au dessus de la note a jouer, et je reajuste au bout de 50 a 100ms=> ca fait a apparaître une note courte ½ ton en dessous ou au dessus de celle qui devrait etre detectee.(ca n'est pas un probleme du detecteur, une guitare ou un piano ne fait pas ca, et en plus le phenomene apparait sur le fondamental et toutes les harmoniques...)

ROUTINES MISE EN JEU
public Interval[] getIntervals(LinkedList evts,float duree_min_interval_s,float seuil_disparition_note_prec_s,float seuil_apparition_hms_s,int nb_tracks_min_interval,int nb_pistes_energy)
cree et regroupe les super_pistes, en notes, de la maniere suivante (pas simple, de piege en piege ca se complexifie...)

les entrees sont une liste d'evenements, qui indiquent l'apparition (type_evt=DEB_NOTE) ou la disparition(type_evt=DEB_NOTE) de pistes

Regle 1 : Si plusieurs evenements de creation arrivent quasi-simultanement (seuil fixe par

seuil_apparition_hms_s), on les regroupe , et si on a un nombre suffisant de creations simultanees(nb min fixe par nb_tracks_min_interval), on detectera la nouvelle note en oubliant toutes les pistes precedant la creation...
Par la suite, tant que la note ne sera pas terminee, on ignorera les evenements de creation qui ne relevent pas de la regle 1 (pas assez de pistes pour remettre en cause la note)
Remarque : les notes les plus sures sont celles qui sont crees par cette regle, pour qu'elle fonctionne bien, il faut au moins une dizaine de filtres / Octave...

Regle 2(independamment de la 1) : si nb_pistes_energy>0 , alors seul les nb_pistes_energy pistes les plus energetiques sont prises en compte (elle ne va pas tarder a disparaître, parce-qu'elle n'apporte pas grand chose au niveau des tests)

Regle 3 : Seules les notes (et non pas les pistes) qui, apres application des regles 1 et 2 ont une duree superieure a duree_min_interval_s, sont retenues.

Regle 4: L'identification d'une note a un instant t ignore les pistes qui finissent avant t+ seuil_disparition_note_prec_s .

voilà c'est tout pour les intervalles....

Information contenue dans les objet Interval

InfoTrack[] tab_infos_track => toutes les super-pistes qui ont ete retenues pour la creation des notes de cet intervalle
InfoNote[] notes; toutes les notes que l'on pourrait deduire de ces pistes (de la plus probable a monsens, a la moins probable). Dans le soft je n'utilise que notes[0]= la plus probable

---

details et routine cle : exploitation de tab_infos_track pour produire notes
=>routine **ContexteFlts.getMeilleuresNotes3()**

en gros
-classe les pistes par frequences croissantes,
-calcule les PGCD des pistes, a ¼ de ton pres, et le poids correspondant
- classe les PGCD par frequence croissantes
- retient comme fondamental la frequence la plus basse qui pese suffisamment lourd

PHASE 4: COMMENT ADAPTER LE SOFT POUR FABRIQUER VOS PROPRES FACE AVANT (DICTEE MUSICALE,....)

=> copier sous un autre nom tout ce qui releve de l'accordeur
modifier les parametres employes lors de l'appel (bouton Accordeur)

Modifier la routine PaintComponent du VotreAccordeurPlotPanel, en utilisant ce qui vous chante
=>les infos_track = toutes les pistes
les infos_track qui ont le champ is_super_track =true => lessuper_pistes
les intervalles qui contiennent les notes de musique retenues....