Université de Nice Sophia-Antipolis
Master 1 IFI & MBDS

TER : Évo–Music

00101 -> ♪♪

Adrover Marc, Caraffa Laurent, Diallo Mamadou Saliou,
Theroude Nicolas, Valdener Marc

Table des matières

Chapitre 1

Introduction

Le propos de ce rapport est d'expliquer comment générer aléatoirement et intelligemment de la musique grâce à un générateur basé sur la théorie musicale, et faire converger cette musique suivant le goût de l'utilisateur grâce à des algorithmes évolutionnaires.

Ce rapport commencera par définir la problématique, puis décrira ce qu'est un algorithme évolutionnaire et définira les bases de la théorie musicale. Les choix de conceptions du programme seront abordés par la suite, ainsi qu'une explication detaillé de l'implementation. Pour finir, une étude sera proposée à travers un questionnaire.

L'application fournie n'a pas la prétention de rivaliser avec des compositeurs, mais plutôt de proposer une solution inédite et surprenante de généreration de la musique.

Chapitre 2

Problématique

Cette partie expose les questions suivantes. Pourquoi ce projet a –t –il été développé ? Comment s'assurer que les morçeaux produits seront écoutables ? Quelles techniques d'optimisations peuvent être utilisées ? Quels problèmes seront rencontrés à l'usage ?

2.1 Pourquoi ce projet ?

La musique est de plus en plus utilisée pour des domaines variés, qui ne nécessitent pas des morçeaux élaborés[1]. Ces domaines vont de la musique proposée lors d'un trajet en ascenseur aux sonneries de portable[2]. Ce projet doit permettre un gain de temps lors de la génération de tels morçeaux.

2.2 Théorie musicale

La théorie musicale s'appuie sur la physique des ondes. Si un morçeau de musique est agréable à écouter, c'est qu'il respecte certaines propriétés physiques. Toutefois son étendue est telle, que seul le système tonal[3] sera utilisé. C'est la démarche adoptée pour produire des morçeaux de musiques dont l'écoute sera agréable.

Un autre problème surviendra lors du croisement de deux morçeaux de musique distincts, car il faudra s'assurer de ne pas perdre en qualité.

Un écueil de cette approche est le grand espace de solution fournie[4]. Un moyen de s'y soustraire est le recours à des algorithmes méta –heuristiques.

2.3 Techniques d'optimisation utilisées

Le problème est d'arriver à faire face à l'explosion combinatoire des possible solutions. Il faudrait un processus permettant l'évaluation et l'amélioration de solutions. De plus ce processus doit être suffissament adaptable pour travailler avec des morçeaux de musiques.

[1] Qui ne nécessitent pas la sensibilité artistique d'un être humain
[2] La qualité des morçeaux n'est cependant pas limité à de tels usages
[3] Le plus système répandu dans la culture occidentale
[4] Le nombre de morçeaux possible dépasse les 2^{50} (voir en annexe)

2.4 Problèmes liés à l'utilisation

2.4.1 Évaluation des morçeaux

Un algorithme n'est pas suffisament performant pour reproduire le jugement humain en terme d'écoute de morçeaux de musique. Il faudrait donc faire appel à un autre moyen d'évaluation. Une solution consiste à demander à des êtres humains de donner leurs avis[5].

2.4.2 Reconnaissance des morçeaux

Toutefois un être humain est lui aussi limité, principalement sa mémoire auditive. En effet il est difficile au cours de l'écoute d'un ensemble de morçeaux de musiques, de mémoriser ces derniers pour les comparer. Sauf si l'évaluateur décide de réécouter ce même ensemble.

2.4.3 Lassitude

Cependant la réécoute d'un même ensemble de morçeaux de musiques provoque rapidement une lassitude chez l'être humain. Cet état peut de plus être augmenter par la répétition d'action « monotone », ainsi le format de présentation devra résoudre ce problème.

[5]Sans tenir compte de l'aspect subjectif du jugement

Chapitre 3

Description de l'application

L'application se découpe en trois parties principales, le générateur musical, l'algorithme évolutionnaire et le site web, qui sert d'interface entre l'utilisateur et le contenu.

L'application fournit deux services différents :

1. « Musique personnalisée » : un utilisateur fait générer des morçeaux de musiques, sur son ordinateur.
2. « Musique du jour » : une communauté vote chaque jour pour deux morçeaux de musiques.

Une applet a été créée pour chacun des deux cas. Pour le premier cas au lancement de l'applet, le générateur crée une première génération de 8 musiques. Ensuite, et ce pour chaque nouvelle génération, l'utilisateur choisit les 2 musiques qu'il préfère et celles-ci serons envoyées à l'algorithme évolutionnaire afin que celui-ci croise leurs paramètres et les envoie au générateur pour qu'il génère de nouvelles musiques. Chaque nouvelle génération contient deux chansons en moins que la précédente. L'expérience se termine donc au bout de 4 générations (8 -> 6 -> 4 -> 2). Quand il le souhaite l'utilisateur peut demander de repartir à zéro avec huit nouveaux morceaux.

Le second cas est légèrement différent, la génération de la musique et l'interaction avec l'algorithme évolutionnaire se fait côté serveur. Au lancement de l'applet le serveur envoie les musiques du jours. L'utilisateur peut alors les écouter et voter pour ses deux morçeaux de musiques préférés. Le serveur stocke les votes et toutes les 24 heures sélectionne les deux morceaux ayant reçu le plus de votes pour la génération suivante. Il y a donc une génération par jour et comme l'algorithme se termine au bout de quatre générations le cycle redémarre tous les quatre jours.

Chapitre 4

Description fonctionnelles

Avant d'entrer dans les détails de la conception du projet nous allons décrire ici les outils que nous avons manipulés pour sa réalisation.

4.1 Algorithmes évolutionnaires

Principes généraux Les algorithmes génétiques sont des métaheuristiques inspirés de la théorie néo-darwinienne de l'évolution. Cette large classe d'algorithmes consiste à faire évoluer une population de solutions potentielles d'un problème d'optimisation donné. A chaque itération, l'algorithme sélectionne les meilleures solutions, puis les modifie aléatoirement afin de constituer une nouvelle population pour l'itération suivante. L'algorithme se termine suivant un critère lié à la convergence de la population ou au temps de calcul. L'efficacité de l'algorithme réside dans le fait que de légères variations aléatoires sur les solutions permettent de découvrir suffisamment souvent de meilleures solutions qui seront sélectionnées pour la suite. Le processus est progressivement guidé par la découverte de meilleures solutions. Cela rejoint les idées de Darwin et de la théorie de l'évolution des espèces naturelles qui postulent une constante et progressive adaptation d'une espèce à son environnement.

Individus et population Dans le monde des algorithmes génétiques, un individu est une solution plus ou moins performante à un problème donné. L'ensemble des individus traités par l'algorithme génétique, se nomme une population. Parmi les individus d'une population, on distingue les individus μ (parents) et les individus λqui sont leurs enfants.

Fonction d'évaluation La fonction objectif (fonction de fitness) est la fonction qui sert à mesurer la performance d'une solution.

Boucle générationnelle Dans cette section, nous allons commenter brièvement les différentes étapes qui constituent un algorithme génétique. La figure 4.1 montre les principales étapes de l'algorithme.

Initialisation de la population initiale Dans l'étape d'initialisation de la population, on prend μ individus. La manière de choisir les individus dépend alors du problème donné.

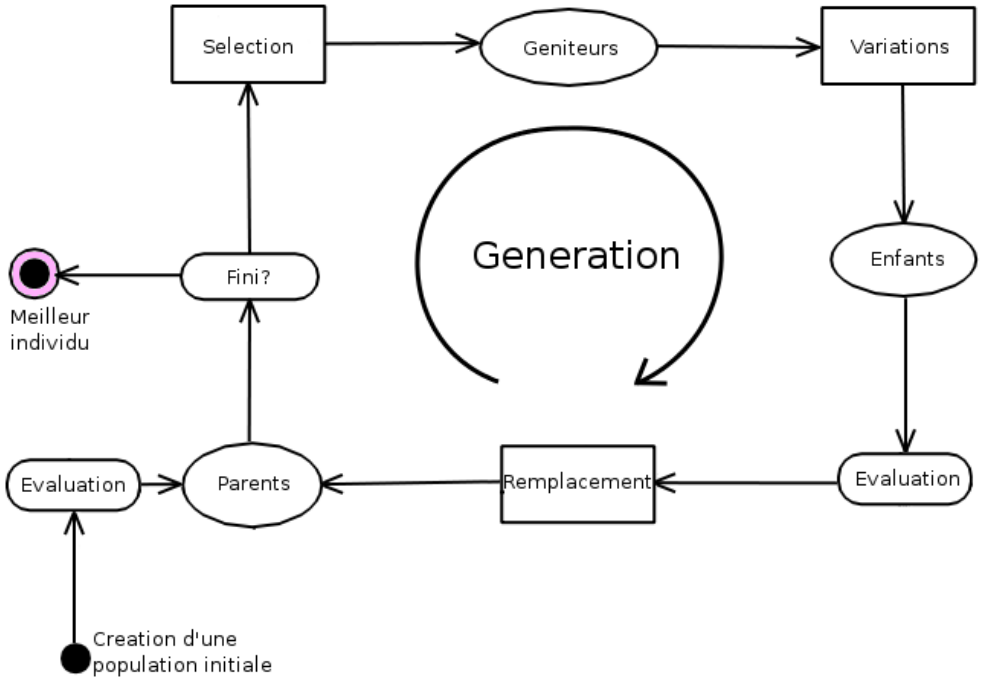


FIG. 4.1 – Boucle Evolutionnaire

Evaluation des μ individus Dans cette étape de la boucle, on évalue chaque individu μ de la population avec la fonction de fitness.

Sélection pour la reproduction Le principe de la sélection pour la reproduction est de choisir parmi les parents d'une génération ceux qui vont se reproduire. La sélection pour le remplacement suppose que parmi une population d'individus μ, on applique un certain taux de reproduction. Par exemple, si le taux de reproduction est de 80%, cela voudra dire que 80% des individus μ vont être sélectionnés pour se reproduire. Il éxiste différentes méthodes de sélection, stochastiques où non, mais qui favorisent toujours les individus ayant la meilleure fitness. Nous avons pour notre part choisis d'utiliser une sélection élitiste.

sélection élitiste Le principe de la sélection élitiste est très simple. Il suffit de choisir les m meilleurs individus parmi les individus μ. Cette méthode privilégie ainsi le phénomène d'intensification.

Opérateurs de variation Pour revenir à notre boucle de notre méta-heurisque, à l'issu de l'étape de sélection pour la reproduction, on a besoin d'opérateurs de variation pour obtenir de nouvelles solutions qu'on appellera individus λ (enfants). Ces opérateurs se nomment croisement et mutation.

Croisement On va prendre les individus μ (parents) précédemment sélectionnés, pour se reproduire deux à deux et donner des individus λ (enfants). L'idée du croisement est proche de celle d'un brassage génétique dans lequel on mélange les caractéristiques de chacun des individus. La figure ci-dessous montre un exemple de croisement en un point. Pour chacun des deux individus, on choisit le même point de croisement. Les caractéristiques qui se trouvent à gauche de ce point pour le premier individu et les caractéristiques qui se trouvent à droite de ce point pour le deuxième individu

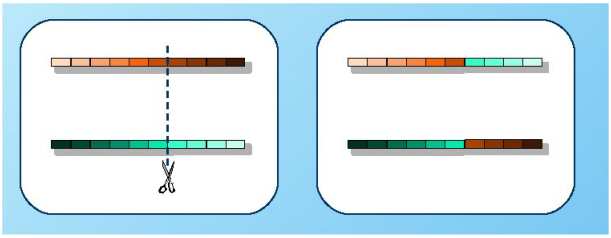


FIG. 4.2 – Un exemple de croisement en un point

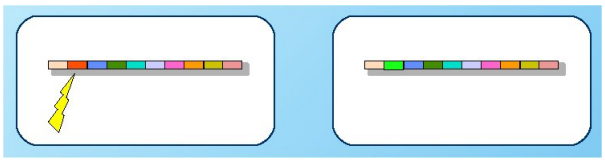


FIG. 4.3 – Un exemple de mutation

vont être raccollées pour donner naissance à un individu λ (figure de droite). On fait de même pour les caractéristiques qui se trouvent à droite du point pour le premier individu et les caractéristiques qui se trouvent à gauche du deuxième individu. On obtient ainsi un deuxième enfant. Il existe d'autres types de croisements possibles notamment les croisements en un point, deux points, n points, ou uniforme . . .Cet opérateur permet ainsi la diversification des solutions.

Mutation Le principe de l'opérateur de mutation est de faire muter un certain nombre d'individus (généralement défini par une probabilité de mutation) en changeant de légères caractéristiques dans le génotype de la solution. La figure 2.3 illustre un exemple de mutation. Sur la figure, si on considère qu'on traite par exemple un tableau de couleurs, l'opérateur de mutation consisterait ici à remplacer le deuxième élément du tableau par une autre couleur. Cet opérateur permet également de diversifier les solutions.

Evaluation des individus λ On évalue par la suite chacun des enfants produits, en faisant appel à la fonction de calcul de la fitness.

Remplacement Afin de réitérer le processus de la boucle, on a besoin de définir un mode de remplacement, afin de constituer la nouvelle génération. Son principe est de sélectionner k individus parmi l'ensemble réunissant l'ancienne population plus les enfants. Dans notre algorithme nous utilisons uu mode de remplacement qui remplace l'ancienne population par les enfants plus les parents.

algorithme :

4.2 Théorie musicale

Nous allons, dans cette partie, expliquer toute la théorie musicale que nous allons utiliser pour la réalisation du générateur de musique.

Algorithm 1 Moteur Evolutionnaire

construtution et évaluation d'une **population initiale**
repeat
  sélection d'une partie de la population
  reproduction des individus sélectionnés
  mutation de la descendance
  évaluation du fitness de chaque individu
  remplacement de la population initiale par une nouvelle population
until atteindre un **critère d'arrêt**

4.2.1 Division du temps

En théorie musical, le temps est divisé de la manière suivante. Nous avons une unité de temps nommée « temps », c'est sur celui ci que va être battue la mesure.

le tempo : le tempo est le nombre de battement par minutes. un tempo de 120 signifie donc que le temps sera battue deux fois par seconde

Une mesure est (dans notre cas) divisée en 4 temps.

un temps est, le plus souvent marquée par une noire (ce n'est pas tout le temps le cas). Á partir de cette noire, nous pouvons definir les notations suivantes dans un mode binaire (c'est a dire une division en deux)

1/4 ronde = 1/2 blanche = 1 noire = 2 croche = 4 double croche ...

pour un rythme ternaire. 1/9 ronde = 1/3 blanche = 1 noire = 3 croche = 9 double croche

Notons qu'en musique, dans une mesure, les temps 1 et 3 sont dit « fort », et les temps 2 et 4 « faible ».

4.2.2 Système tonal

Le système tonal est le résultat d'une lente évolution qui s'est opérée sur quelque six siècles. Dans cette partie, nous allons donc expliquer, et vulgariser, les grand axe du système tonal, Nous tenons a dire qu'il y a surement des imperfections dans les explication, veuillez nous en excusez.

structure musical Le système tonal est composée de la manière suivante : 8 notes dans l'ordre : Do Ré Mi Fa Sol La Si Do. qui représente un octave. Un octave est divise en 12 demi –tons. L'espacement des notes de la gamme Majeur (gamme principal) est : Do –1ton –Ré –1ton –Mi –1/2ton –Fa –1ton –Sol –1ton –La –1ton –Si

Définition du système tonal Le système tonal repose sur les sept degrés de l'échelle diatonique, avec comme degré de base la tonique (premier degré). Par exemple, les 7 dégrées de la gamme majeur en do donne la gamme majeur en do :

I	II	III	IV	V	VI	VII	
Do	Ré	Mib	Fa	Sol	Lab	Sib	Do
1	1/2	1	1	1	1	1/2	

Les gammes Nous pouvons, sur ces 7 degrés, appliqué la gamme mineur, ou majeur. Par exemple, la gamme mineur donne :

I	II	III	IV	V	VI	VII	
Do	Ré	Mib	Fa	Sol	Lab	Sib	Do

Les modes Un mode est une gamme, mais décalée. Prenons notre gamme majeur. Do Ré Mi Fa Sol La Si Do. le mode en Ré, de notre gamme majeur en Do donne : Ré Mi Fa Sol La Si Do Ré.

Modulation Moduler une mélodie consiste a jouer cette mélodie, mais sur des notes différentes, il existe deux type de modulation :

1. modulation modal : ça consiste à jouer une mélodie sur un autre mode de la gamme.
2. modulation tonnal : ça consiste à jouer une mélodie en changeant la tonique de la gamme.

L'harmonie est l'art de faire jouer plusieurs sons ensemble.

Chapitre 5

Conception

Ce chapitre essai de décrire et de justifier les choix de conception que nous avons fait pour le projet.

5.1 Technologie

5.1.1 Choix du langage de programmation

Le langage `Java` a été choisi pour la richesse de son api, pour ensuite avoir un code homogène. (les trois parties sont en java, le générateur, l'algorithme évolutionnaire, l'applet) de plus, c'est un langage metrisée par toute l'équipe.

5.1.2 Choix des applets

Pour permettre aux utilisateurs de générer et d'écouter des morceaux, un site Web dynamique à base du `html` et de `Java Script` à été mis en place. Ce site propose plusieurs services dont l'accès aux musiques du jour et aux musiques personalisées. Pour ces deux services on a utilisé une applet pour chaque. Le choix d'utiliser les applets a été motivé par plusieurs raisons :

- Le besoin d'utiliser du lecteur midi : le moteur de génération de la musique ne générant que du midi(comme l'impose le sujet du projet)
- Le besoin de travailler coté client : pour alléger le serveur on a jugé nécessaire de faire tout le travail de génération chez le client ce qui éviterait aussi tout problème lié au accès concurents et multiples du serveur.
- Le problème de compatibilité : la difficulté de trouver des lecteurs flash pour lire du midi, qui soient compatibles avec tous les systèmes d'exploitation mais aussi éviter toute conversion du midi en d'autre format comme mp3

5.2 La conception du générateur

Pour produire une musique évolutive, une architecture extrêmement modulable est préfèrable.

Informations relatives et independantes Le croisement des musiques impose de stocker les informations de façon relatives, pour que, par la suite, le générateur puisse combiner les informations indépendaments. C'est une décision très importante. car choisir de stocker par exemple une melodie en fonction des hauteur, et non avec une position relative dans la gamme, empecherait par la suite d'appliquer la mélodie sur une autre gamme que

sur celle de départ. Par exemple, une mélodie basée sur une gamme majeur sonnera « gai »tandis que la même mélodie sur une gamme mineur sonnera « triste ».

Générateur modulables Le générateur se veut modulable, il doit être facile de créer de nouveaux micro-Générateurs (classe instantiable). Un chargement par introspection des micro-Générateurs a été choisi, comme ça, chaqu'un peut créer son propre micro Générateur.

Micro–Générateurs simple à implementer Les micro-Générateurs instantiables devront redéfinir des classes abstraites, ou, toutes les méthodes nécessaires à la création de la musique ont été déclarées, ainsi, le programmeur n'aura qu'a piocher dans les informations des super classes. Il restera au final, que très peux de code à écrire pour generer un nouveau micro generateur.

5.3 Conception du moteur évolutionnaire

Algorithme générique Les algorithmes évolutionnaires sont utilisés pour toutes sortes de problèmes, même si l'implémentation derrière est différente pour chaque problème ils utilisent tous une même base commune.

Nous avons donc décidé, conjointement avec nos encadrants, de faire le squelette de l'algorithme de facon générique afin qu'il soit réutilisable. C'est d'autant plus intéressant que `Java` se préte particulierementà la réutilisation de code.

Le code de l'algorithme évolutionnaire se découpe donc en deux parties, un squelette générique et son implémentation spécifique au projet.

La partie générique modèlise la boucle évolutionnaire et ses cinq phases (4.1) :

1. initialisation
2. sélection
3. croisements
4. mutations
5. remplacement

Chacune des cinq phases est remplacée par une interface. L'implémentation de la phase de sélection et de celle de remplacement est aussi générique. La construction de l'objet représentant l'algorithme requiert le passage en paramètre d'une liste contenant les croisements ainsi que d'une liste contenant les mutations, avec leur probabilité. Celà permet de facilement rajouter ou enlever une variation, même dynamiquement.

Les solutions manipulées par l'algorithme génétique sont aussi représentés sous la forme d'une interface, qui implémente elle même l'interface Comparable, qui permet de définir une fonction de comparaison des individus. Elle possède aussi une méthode pour obtenir le fitness et une méthode pour faire une copie de l'élément, tout cela étant suffisant mais nécessaire à l'élaboration de la boucle évolutionnaire générique .

Au final l'élaboration d'un nouvel algorithme évolutionnaire ne demande plus qu'à implémenter l'interface modélisant les solutions ainsi que les phases d'initialisation, de croisement et de mutation.

Stockage des paramètres de musique Chaque musique contient un certain nombre de paramètres créés lors de la première génération puis ensuite manipulés par l'algorithme évolutionnaire. Il fallait décider d'un moyen de stocker ces données. Nous avons choisit de passer par un fichier `XML` regroupant tout les paramètres d'un morçeau musique. Car le `XML` est un format standardisé, dont les outils de manipulation sont complets et efficaces. Cela permet aussi d'avoir une persistance des données et un autre avantage important est que la lecture de l'`XML` par un être humain est relativement aisée.

Manipulation des paramètres de l'algorithme évolutionnaire Un algorithme évolutionnaire contient un grande nombre de variables paramétrables qu'il est nécessaire de souvent modifier pendant la phase de calibrage. C'est pourquoi on regroupé tout ces paramètres dans une même classe à accès statique. Ainsi on a une vision globale des variables du moteur et il est aisé d'y accéder et de les modifier.

5.4 Interfaces de présentation

Pour fournir les services aux utilisateurs, nous avons mis en place deux applets, l'une pour assurer un service « musique personnalisée », l'autre pour assurer celui de « musique du jour ».

Les fonctions , qui doivent être assurées pour les deux services, sont l'écoute et la sélection des morçeaux. L'écoute permet à l'utilisateur de se faire un avis sur les morçeaux proposés. La sélection permet de choisir deux morçeaux parmis les morçeaux proposés par l'applet.

5.4.1 Service "Musique personnalisée"

L'applet qui assure ce service fournit les fonctions d'écoute et de sélection ainsi qu'une fonction supplémentaire de génération de nouveaux morçeaux. Cette dernière fonction entraîne une écriture sur le disque de l'ordinateur du client ce qui oblige à signer cette applet. L'utilisateur doit donc accepter un certificat de confiance pour télécharger l'applet.

L'applet permetà l'utilisateur l'écoute de 8 morçeaux, chacun étant associé à une image le représentant. Cette image est élaborée en fonction du tempo, de la mélodie, de la basse et de la densité temporel de notes. Le tempo donne la couleur de l'image (du bleu au rouge, bleu pour les tempo lent, rouge pour les rapides). La densité de notes donne la teinte (plus elle est fonçée, plus la densité est élévée). La mélodie est représentée par la ligne supérieure, et la ligne inférieure représente la basse.

L'usage de l'applet doit être très simple. Après une écoute des morçeaux de son choix, l'utilisateur doit en sélectionner deux. Puis il doit demander la génération d'un nouvel ensemble de morçeaux. Cet ensemble a deux morçeaux de moins que l'ensemble précédent dont les deux premiers morçeaux sont ceux sélectionnés préalablement.

5.4.2 Service « musique du jour »

Ce service propose aux utilisateurs d'écouter et de voter autant de fois et pour autant de morçeaux qu'il le désire.

5.5 Fatigue de l'utilisateur

L'utilisation d'algorithmes évolutionnaires interactifs pose le problème de la fatigue de l'utilisateur. En effet pour évaluer correctement les solutions celui-ci n'a d'autres choix que d'écouter toutes les musiques.

Deux problèmes majeurs engendraient une fatigue rapide de l'utilisateur :

- D'une part l'écoute d'un trop grand nombre de musiques devenait vite ennuyeux.
- D'autre part une fois un certain nombre de morçeaux écoutés, il devenait difficile de se rappeler des premières chansons écoutées et donc de sélectionner les deux meilleures, ce qui obligeait soit à les réécouter (cercle vicieux), soità donner un avis pouvant être moins pertinent.

À partir de ces deux problèmes on a distingué trois points importants à améliorer :

- Limiter le nombre de musiques a écouter, ce qui impliquait une convergence rapide de l'algorithme évolutionnaire.
- Donner des repères visuels aux musiques afin de faciliter leur mémorisation.
- Réduire le temps d'écoute nécessaire à l'évaluation d'un morceau.

Pour diminuer le nombre de morceaux il a été décidé de commencer avec 8 morçeaux de musique puis de réduire ce nombre de 2 à chaque génération. Comme les morçeaux de musique sélectionnés sont aussi présents, le nombre de morçeaux de musique à écouter dans les générations suivantes est donc très réduit.

Le fait de mettre 8 morçeaux de musique au départ ne pose pas de problèmes car l'utilisateur est alors « frais », il permet d'avoir une meilleure chance que l'utilisateur trouve au moins deux morçeaux de musique qui lui plaisent. En outre, en réduisant la taille de chaque nouvelle génération de

deux, on sait que cela va finir au bout de quatre étapes, ce qui donne des repères à l'utilisateur.

Pour facilement reconnaître et mémoriser les différents morceaux on a intégré trois repères visuels. Un nom, généré à partir d'une banque de syllabes, est attribué à chaque morçeau de musique.

De plus une image, personalisée pour chaque morçeau de musique, est généré. Enfin l'utilisateur peut donner à un aide mémoire sur la qualité du morçeau écouté à travers trois boutons de jugement.

L'image a aussi l'avantage de montrer la structure du morçeau. Ainsi l'utilisateur peut se faire une bonne idée du caractère du morçeau juste en regardant son image. L'autre avantage est que l'image fait office de player et permet l'écoute d'extrait.

Efin une interface épurée et fonctionnelle permettant de limiter au maximum le temps écoulé entre deux écoutes de musique, a été choisi.

Chapitre 6

Description de l'implémentation

Ce chapitre détaille l'implémentation du projet, qui a découlé de des choix de conceptions.

6.1 Avancement du couple générateur/algorithme évolutionnaire, étape par étape

Voici les grandes lignes directrices qui ont guidées la création du générateur musical et de l'algorithme évolutionnaire.

1 : Création d'une version fonctionnelle On a voulu obtenir rapidement une première version qui effectue le cycle générateur musical / algorithme évolutionnaire. C'est pourquoi on s'est attaché à faire quelque chose de simple mais avec la volonté d'avoir une structure solide, qui soit réutilisable pour les versions ultérieures.

2 : Ajout de la mélodie Les morçeaux de musique générés lors de la première version étaient déjà d'une qualité satisfaisante grâce au respect des règles de la théorie musicale. Cependant il manquait un thème caractéristique. C'est pourquoi a ajouté la création d'une mélodie. Le morçeau s'en trouvait améliorée mais le problème était que d'une génération à l'autre la convergence musicale était trop faible. C'était dû au fait qu'une nouvelle mélodie était créé à chaque fois. On a donc décidé d'intégrer la mélodie aux paramètres de l'algorithme évolutionnaire, qui a donc été ajouté au fichier xml. Ce qui a permis d'effectuer des croisements sur les mélodies, qui mettent bien en valeur la relation avec les géniteurs.

3 : Ajout de la structure L'écoute de la musique était devenu plaisante, et les variations significatives mais la musique bouclait sur quelques mesures pendant tout la durée de la chanson. Pour y remédier il a été décidé d'introduire une structure à la musique, en la composant en différentes parties (introduction, couplet, pont, solo, fin), ce qui a ajouté de la dynamicité

à la musique. Afin de surprendre un peu l'utilisateur nous avons aussi ajouté des variations sur la structure.

4 : Calibration Pour l'algorithme évolutionnaire cela considérait surtout à ajuster tout les probabilités de croisement et de mutation. Globalement on a mis des probabilités faibles pour les variations apportant une grande diversité musicale afin de garder une convergence forte, et une probabilité plus élevée pour les variations apportant peu de diversité ou qui nous semblaient particulièrement intéressantes. Concernant le générateur musical la calibration s'est surtout faite au niveau de la création de la mélodie, en essayant de trouver un bon compromis entre une bonne probabilité d'avoir une mélodie qui sonne bien et un espace musical pas trop réduit.

6.2 La musique

6.2.1 La structure de la musique

Nous avons décidé, d'abstraire (partiellement) la structure midi, pour premièrement une utilisation plus facile, et deuxièmement une meilleur portabilité. Dans cette partie, nous expliquerons donc l'architecture de la structure musicale[1], mais aussi les fonctionnalités des différents éléments.

L'architecture d'une musique est la suivante : piste, cycle, mesure, note.

Une piste est la classe dans laquelle est stockée la musique générée. Cette classe possède :

- une liste de cycle.

la classe cycles : Un cycle peut être représenté un "couplet mélodie", un solo, un pont. Il est composée donc :

- d'une énumération : représentant le type du cycle (solo, etc...)
- d'une liste de mesures.

La classe mesure : Une mesure comporte plusieurs informations.

- la tonique : (la note de base de la gamme)
- le dégrée : le dégrée sur laquelle sera interprétée la mesure
- la gamme actuel : la gamme actuel sur laquelle sera calculé la note correspondante au dégrée

La classe note : La note est l'élément le plus bas de l'architecture. Une note est composée :

- d'une hauteur : représente la hauteur de la note, ici l'architecture midi a été conservée.
- d'une durée : représente la duré en temps de la note.
- d'un temps : représente le temps que la note sonne.
- d'un TIC : le tic est une variable statique, elle représente l'unité de temps.

[1] Plus de détail en annexes

i La classe accords : Elle étend la classe note : Sa particularité est de comportée plusieurs hauteurs.

La classe gamme : Cette classe représente une gammes, elle est crée a partir d'écart de notes. la aussi la position relative des notes est très importante pour la modularité du générateur. Cette classe est dotée de plusieurs fonctions clé très importantes

6.2.2 Le générateur

Le générateur est conçu d'un chef d'orchestre et de plusieur micro –générateurs, ce qui est une solution offrant à la fois une grande modularité, ainsi qu'une grande facilité d'implémentation et peu de limites dans la création. Le générateur chef d'orchestre ordonnancera la création des musiques. Les micro –générateurs possèdent plusieurs méthodes pour générer des mesures. Ces méthodes seront appelées par le chef d'orchestre en fonction du contexte.

Le générateur chef d'orchestre C'est une classe très simple qui est composée d'une méthode principal qui fait composer successivement les microGénérateur. Plus précisément, cette méthode est composée d'une itération qui boucle sur chaque mesure de tous les MicroGénérateurs, et qui, en fonction du contexte, leur demandera de composer cette mesure.

Un micro –générateur C'est une classe abstraite, c'est la classe Mère de tout les micro –générateurs, elle comporte toutes les informations principales utiles aux générateurs fils, mais aussi les méthodes, à redéfinir, appelées par le générateur chef d'orchestre pour la composition des mesures.

Les classes MicroGénérateurMelodie, MicroGénérateurBass, MicroGénérateurRyhtmique, MicroGénérateurBatterie. Ces quatres classes implementent la classe MicroGénérateur, elle sont aussi abstraites, mais elles implémentent `toutes` les méthodes de compositions. Ces classes possèdent de plus, des méthodes et attributs propres a leurs rôles.

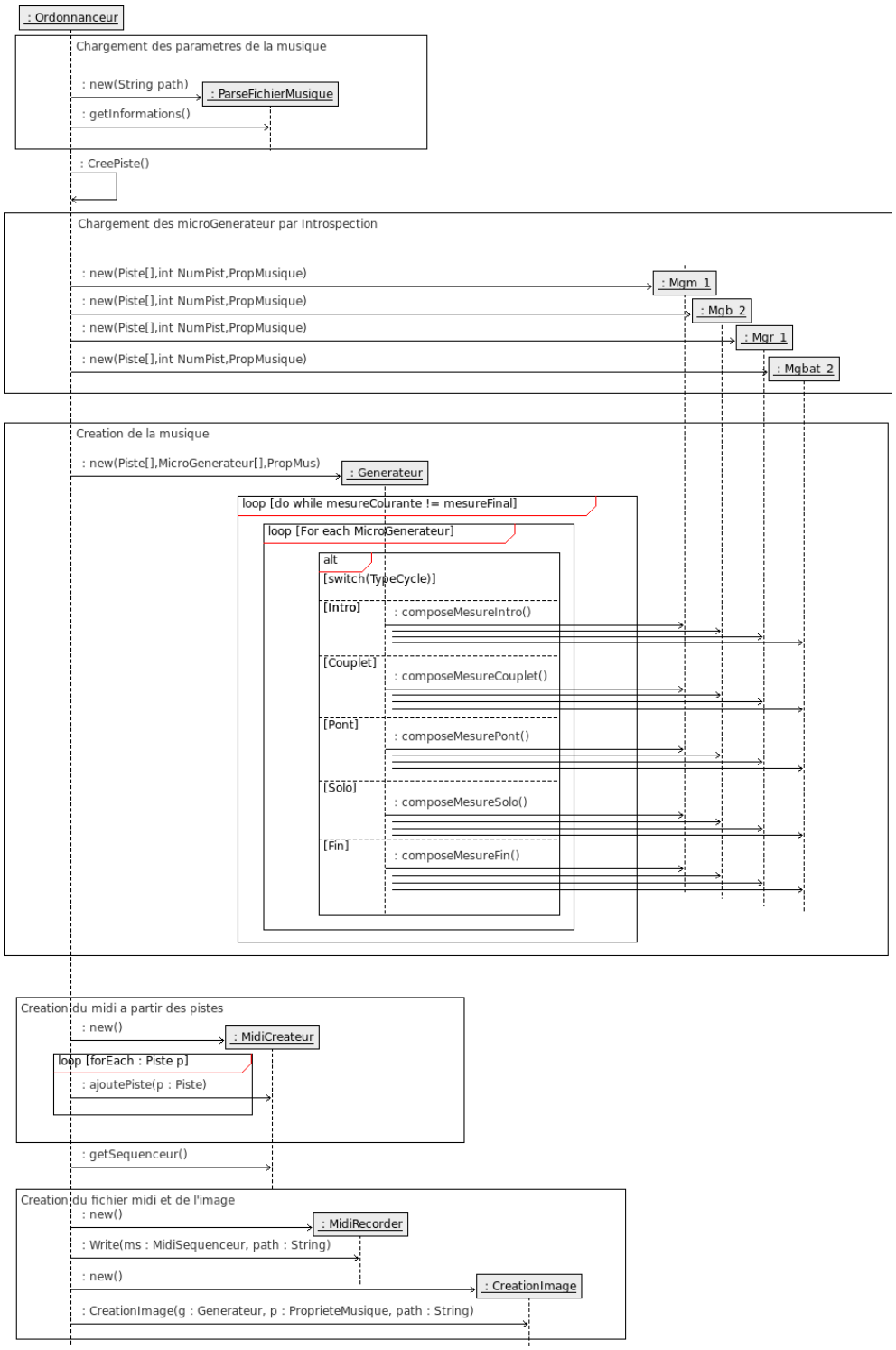
La classe MicroGénérateurMelodie : se charge de composée la melodie principale de la musique. Ce générateur a une méthode spécifique pour generer les premieres melodies : La classe MicroGénérateurBass : se charge de composer la bass de la chanson. La classe MicroGénérateurRyhtmique : se charge de composer la ryhtmique. La classe MicroGénérateurBatterie : se charge de composer la batterie.

Les micro –générateurs instantiables Ceux sont les générateurs qui seront chargés par introspection. Ils doivent étendre l'un des 4 micro –générateurs abstraits.

6.2.3 De la théorie à la pratique : le processus de création

La création de la musique

La premiere génération est une étape importante, car c'est là où sont tirés les paramètres des musiques. À chaque itération de l'algorithme évolutionnaire, des musiques vont donc être crées.



Premièrement, il faut charger les paramètres du fichier `XML`. Ensuite, les micro –générateurs choisis par instrument vont etre chargés par introspection. Le générateur chef d'orchestre va être créé avec un tableau de micro–générateurs en paramètres. La méthode startGénérateur va etre appelé pour générer la musique. Ensuite, la structure musicale créée au préalable va être transposée en message midi. Pour finir, un fichier midi sera créé, avec

l'image correspondante.

6.2.4 spectre des musiques générée par le Générateur

Il est important de calculer la taille de la population generée. Dans notre cas, cela sera utile pour, par la suite, optimiser l'algorithme. À noter que la musique créee par un generateur n'est pas deterministe, nous allons donc compter toutes les musiques cree par un micro–Generateur comme un element de la population.

Les parametres varian de notre generateur sont :

les gammes majeur et mineur.

Nombre de melodie possible Le générateur joue une melodie sur au plus 4 mesure.

une notes, dans une melodie, peux avoir : 5 durée possible (ronde, blanche, noir, croche, double croche), rappelons que 2 doubles croches = une croche. 2 croches = une noire, 2 noire = une blanche, 2 blanche = une rondes, si nous prenons en compte que notre générateur peux ne générer aucune note dans une mesure, nous pouvons intégrer la notion arbitraire de une double ronde = deux ronde une quadruple ronde = quatre ronde

7 hauteur possible (limitons nous a une gamme a 7 tons).

Raisonnons de la façons suivante, une noire peut avoir 8 états (silence, do, re, mi, fa, sol, la, si). cela fait donc 8 possibilités. Si nous autorisons la division de la noire en croche, cela fait soit nous aurons une noire, 8 états possibles, soit deux croche, avec chacune des croches 8 etats possibles, c'est a dire si 64 états possibles pour les deux croche, ce qui fait au total 8 état pour la noire plus 8 * 8 pour les deux croche. c'est a dir 72 états possibles.

Si nous conjecturons le raisonnement. soit k le nombre d'état possible d'une note, soit n le nombre de fois qu'une note peut être divisée en deux. alors la fonction $f(n) = k + f(n-1) * f(n-1).f(0) = k$ Nous donne le nombre de combinaisons possible.

notre générateur possède donc 6 degrés de temps possibles avec 8 état par note possible, ce qui nous donne : $2{,}78\hat{5}9$ melodie possibles

nombre de cadences possible rappelons que une cadence est une succession de dégrée dans une gamme sur lesquelles joue la melodie et les accords. nous générons des cadences sur 4 mesures. ps(notons que nous avons aussi intégrée des cadences spécifique au blues par exemple qui est sur 12 mesures, mais nous mentons cette exception de coté, ce qui ferait exploser la complexité)

nombres de cadences possibles $2\hat{4}$.

nombre de générateur possibles il y a au minimum un générateur par instrument,

notons : kMelo le nombre de générateur Melodie kBass le nombre de générateur bass kRyth le nombre de générateur ryhtmique kBat le nombre de générateur batterie.

le nombre de combinaisons possibles est donc kmelo * kbass * kryth * kbat.

tempo différents nous avons une variation de tempo de 80 a 180, cela fait donc 100 tempo differents possibles.

strucutres de la musiquess Nous avons 3 types ce cycles qui n'ont pas de regle de permutations, supposons que les MicroGenerateur crée, pour un cycle d'un certain type, le même cycle a chaque fois. une chanson peut aller d'un cycle a 10 cycle, nous avons donc $3\hat{1}0$ - $1\hat{1}$. claques combinaison de cycle peux posséder une introduction ou pas, chaque combinaison de cycle peux posséder une fin ou pas, nous arrivons donc a un total de $(3^10 - 1) * 4$ combinaison de cycle possible.

pour l'instant, le nombres de musique possibles est de 2 * $2{,}78\hat{5}9$ * $2\hat{4}$ * (kmelo*kbass*kryth*kbatt) * 100 * ($3\hat{1}0$ -1)*4

6.2.5 Optimisation

Le nombre de musique possible est énorme (supperieur au nombre d'atome dans l'univers, ce qui est courant pour un problème complexe). et nous somme d'accord que dans ce spectre, certaines musiques doivent être beaucoup mieux que d'autre. au vue du nombre très limitée d'itérations possible pour notre algorithme évolutionnaire, nous devons impérativement optimiser le générateur pour avoir de bonne musique rapidement, nous ne pouvons pas nous permettre de faire de l'aléatoire pur dans la population.

Conclusion partiel : notre algorithme, malgré d'énorme restriction
- se baser sur la théorie musical et ne se permettes que très peux d'extra.
- restriction due au croisement et à la généricité des musiques

possède encore une population trop vaste.

Comment réduire cette population ? en regardant les paramètres influents, nous pouvons voir que la population explose avec la melodie. Nous allons donc, lors de la création d'une melodie, et grâce a la théorie musicale, essayé de créer des melodie qui sonne bien. Comment allons nous faire ?
- accentuer les temps forts (une forte probabilité qu'une note qui sonne juste, soit jouée sur un temps fort)
- ajouter des blanc a la fin des mesures (sentiment de repos indispensable pour le testeur)
- utiliser certain paterne musicale connue et reconnaissable a l'oreille plutôt que de l'aléatoire pure (montée descente)

Il y a-t-il d'autre restriction possible ? bien-sur, Tout les aspects de l'harmonie et de la théorie musicale entre en jeu dans cette partie, il y a énormément de convenions, et dans l'absolue, nous aurions pu toute les implémenter Mais serait-il vraiment judicieux de faire cela ?

6.3 Les classes de l'algorithme évolutionnaire

Cette section donne une rapide description des classes les plus significatives de l'implémentation de l'algorithme évolutionnaire, notamment les classes effectuant les variations sur les paramètres des musiques.

`EvoParam` : Regroupe toute les variables utilisées par l'algorithme évolutionnaire

`ParametresMusiques` : Solution manipulée par l'algorithme évolutionnaire, réprésente l'ensemble des paramétres d'une musique

`Parametre` : Représente un paramètre de la musique

`ParseFichierMusique` : Lien entre le fichier xml représentant la musique et son abstraction en ParamètreMusique (transcription xml -> ParametresMusiques et ParametresMusiques -> xml)

`CroisementMusique` : Effectue un croisement de 2 musiques
- Instrument -> échange des valeurs
- Tempo -> échange des valeurs
- Fréquence des notes -> échange des valeurs
- Gamme -> échange des valeurs
- Mélodie -> prend des morceaux de chaque mélodie
- Cycles -> Enlève, ajoute où modifie un cycle

`MutationCycle1` : Échange la place de deux cycles

`MutationCycle2` : Mélange les toniques d'un cycle

`MutationRythme` : Fait varier la fréquence et le tempo grace à un tirage aléatoire autour de l'ancienne valeur

`MutationInstrument` : Change aléatoirement les micro-générateurs utilisés

`MutationMélodie1` : Demande au générateur de générer une nouvelle mélodie

`MutationMélodie2` : Réordonne aléatoirement les différentes parties de la mélodie

`MutationMélodie3` : Modifie les hauteurs de note de la mélodie

6.4 Le site web

Services

Le site web accessible à l'adresse suivante : *http ://evo-music.org* propose plusieurs services afin d'utiliser et découvrir l'application.Plus que les traditionnels onglets **Liens** et **Contact**,l'essentiel de l'application réside dans les onglets **Musique Personnalisée** et **Musique du Jour** qui est au moment de l'éciture de ce rapport en construction car implementé mais non accessible du fait du besoin d'un hébergeur Java souvent honéreux.Un **Forum** est également disponible à l'adresse suivante : *http ://evomusic.aceboard.fr/*.

Technologies

Le site web à été réalisé avec HTML et JavaScript par un soucis de simplicité et d'efficacité.Concernant l'application en elle même,elle fait appel à deux applets(dont le screenshot est visible plus bas) pour assurer les prestations des services et à une servlet.Cette dernière,théoriquement utilisée pour le service **Musique du Jour**,permet de récupérer et traiter les votes des utilisateurs côté serveur ainsi qu'assurer la persistance des morceaux.

Look and Feel

Pour que l'utilisateur puisse se sentir apaisé et concentré sur les tâches que le site lui propose,il a été décidé de mettre un point d'honneur à rendre le site reposant,attractif et simple.En effet,les couleurs ont été choisies pour être harmonieuses et ne pas aggresser l'oeil du visiteur,la disposition des différents élements est faite pour ne pas perdre l'utilisateur dans le superficiel ou l'inutile.

Chapitre 7

Etude auprès d'une population témoin

7.1 Questionnaire

7.1.1 Ce qu'on attend du questionnaire

Le questionnaire que nous avons rédigé a pour but, non seulement de savoir si le service développé plaît aux utilsateurs, mais aussi de savoir quelles évolutions ils ont remarqué. Notamment nous désirions savoir si ils avaient remarqué une convergence dans les musiques produites.

Dans un premier temps, c'est le but des cinq premières questions, nous nous intéressons à la personne répondant au questionnaire. Ces questions ont pour but à la fois de nous renseigner sur son "niveau" musical et de permettre à la personne de répondre rapidement, afin d'obtenir des réponses plus nettes[1].

Les trois questions suivantes nous servent à évaluer la perception par l'utilisateur de notre présentation. Nous cherchons à savoir si le moteur évolutionnaire a fourni un échantillon suffissament varié et si ce dernier apprécie le fait que le nombre de musique proposées diminue à chaque nouvelle génération.

Les cinq questions suivantes permettent de savoir si l'utilisateur a remarqué une convergence. Elles nous permettent de juger de l'éfficacité du moteur évolutionnaire et de son interaction avec le générateur musicale.

Les trois questions suivantes sont une sorte de deuxième essai pour savoir si l'utilisateur a remarqué une convergence. En effet les images sont directement générées à partir des fichiers de descriptions des musiques. Les perceptions visuelles étant relativement identiques d'un individu à l'autre[2], nou cherchons à savoir quel paramètre visuel prédomine. Et ce paramètre représentant directement un paramètre auditif, il serait peut être possible

[1] c'est aussi dans ce but que la majorité des questions acceptent oui ou non comme réponse

[2] trouver un article qui va dans ce sens

d'en déduire lequel prédomine. Pour celà nous pourrions changer les relations entre la représentation visuelle et la musique, par exemple la couleur ne représenterait plus le tempo mais la densité de note sur le morçeau, et représenter le questionnaire à une population d'étude similaire.

Les dernières questions sont utilisées pour permettre à la personne répondant au questionnaire d'exprimer plus librement son avis. Notamment sur ce qu'elle voudrait voir amélioré.

7.1.2 Le questionnaire lui même

Voir 9.1.2

7.1.3 La façon dont on l'a rédigé

Nous avons délibéremment choisi de limiter le questionnaire au service proposant la génération de musique à un seul utilisateur. En effet un questionnaire de même genre vis à vis du service "musique du jour" est biaisé par le fait que le « goût collectif »ne correspond pas forcément au gôût d'un individu particulier. Ce qui a une influence sur les musiques sélectionnées.

Avant la rédaction du questionnaire, nous avons fait une liste de ce qui pouvait être évalué pour ce service à savoir :

- l'aspect visuel du service
- l'ergonomie
- la qualité d'écoute (dépendant notamment du format)
- l'évolution des musiques
- l'évolution des images
- si l'utilisateur a du plaisir a utilisé ce service
- ....

.

Et afin de limiter la taille du questionnaire, nous avons décider de le limiter à l'évolution des musiques et des images. Celà pour obtenir un plus grand retour d'information sur le moteur évolutionnaire et son influence sur la production de musiques.Pour faciliter le traitement des réponses, nous les avons limiter à "oui" et "non".

Puis devant le résultat pour rendre le questionnaire moins austère, il a été décidé de rajouter quelques questions. La première idée venue, puisque les réponses dépendent des goûts et des percepitions de la personne répondant, a été d"'évaluer" cette personne. Puis après un premier passage de réponse au questionnaire, il a été décidé de laisser à la personne évaluée plus de liberté pour s'exprimer et c'est la raison des dernières questions

7.2 L'étude en elle même

En cours de réalisation.

7.2.1 Le panel de personnes interrogé

7.2.2 Les résultats

Chapitre 8

Conclusion

8.1 Résultats

Ce projet qui avait pour but de générer intelligemment de la musique est déjà abouti. En effet une solution à été trouvée pour générer aléatoirement de la musique, au goût de l'utilisateur et celà pour toutes les générations. De plus un sentiment de surprise et d'étonnement est ressenti lors d'une première expérience, ce qui est en soit une grande satisfaction. Bien sûr aucun des morçeaux produits ne peux rivaliser avec la production d'un compositeur[1] qui s'adresse à un public bien spécifique, mais le concept « Évo–Music »peut se targuer de s'adapter à chaque personnalité.

8.2 Améliorations

8.2.1 Musicale

Toutefois la prise en compte des spécificités de chaque instrument, au niveau des micro–générateurs, apporterait de meilleures résultats. Par exemple pour une guitare, il existe des effets tels que les « blend », « harmoniques »et les « slides »qui ajoutent de la couleur au morçeau.

8.2.2 Utilisation

Une plus grande exploitation des votes récoltés, par le service « musique du jour », permettrait une meilleure compréhension des goûts musicaux, et donc un raffinement de nos algorithmes évolutionnaires.

[1] Ce qui n'est pas le but recherché

Chapitre 9

Annexes & sources

9.1 Annexes

9.1.1 Description d'un algorithme évolutionnaire générique

construtionet évaluation d'une **population initiale** ;
jusqu'à atteindre un **critère d'arrêt** :
sélection d'une partie de la population,
reproduction des individus sélectionnés,
mutation de la descendance,
évaluation du degrés d'adaptation de chaque individu,
remplacement de la population initiale par une nouvelle population

9.1.2 Questionnaire

1. Écoutez vous souvent de la musique ?
2. Quelles sont vos goûts en matière musicale ?
3. Avez vous des connaissances théoriques dans le domaine musical(solfège, théorie musicale ...)
4. Si oui avez vous déjà composé de la musique ?
5. Jouez vous d'un instrument ?
6. Trouvez que le musiques proposées à la première étape sont suffissament variées ?
7. Comment qualifiriez vous le nombre de musique proposées :
   (a) trop important
   (b) suffissant
   (c) insuffissant
8. L'évolution du nombre de musiques proposées vous semble t elle satisfaisante ?
9. Avez vous trouvé une grande différence entre les musiques proposées à l'étape 1 ?
10. Avez vous trouvé une grande différence entre les musiques proposées à l'étape 2 ?
11. Avez vous trouvé une grande différence entre les musiques proposées à l'étape 3 ?
12. En sélectionnant les musiques imaginiez vous déjà les futures musiques produites ?

13. L'évolution des musiques, proposées aux étapes 2 et 3, correspondait elle à vos attentes ?
14. Les images vous aident elles à identifier les musiques ?
15. Trouvez vous les images représentatives des musiques ?
16. Avez vous noté une évolution dans ces images ?
    (a) au niveau des couleurs
    (b) au niveau du remplissage (les notes représentées)
17. Avez vous été surpris par les musiques qui vous ont été proposées ? Pourriez vous leur donnez un adjectif les représentants ?
18. Que voudriez vous voir amélioré ?
19. Pensez vous réutiliser l'application ?

Définition du système tonal Le système tonal repose sur les sept degrés de l'échelle diatonique, avec comme degré de base la tonique (premier degré). Par exemple, les 7 dégrées de la gamme majeur en do donne la gamme majeur en do :

I	II	III	IV	V	VI	VII	
Do	Ré	Mib	Fa	Sol	Lab	Sib	Do
1	1/2	1	1	1	1	1/2	

Les gammes Nous pouvons, sur ces 7 degrés, appliqué la gamme mineur, ou majeur. Par exemple, la gamme mineur donne : I II III IV V VI VII Do-Ré-Mib-Fa-Sol-Lab-Sib-Do

Les modes Un mode est une gamme, mais décalée. Prenons notre gamme majeur. Do-Ré-Mi-Fa-Sol-La-Si-Do. le mode en Ré, de notre gamme majeur en Do donne : Ré-Mi-Fa-Sol-La-Si-Do-Ré.

Modulation Moduler une mélodie consiste a jouer cette mélodie, mais sur des notes différentes, il existe deux type de modulation modulation modal : ça consiste à jouer une mélodie sur un autre mode de la gamme. modulation tonnal : ça consiste à jouer une mélodie en changeant la tonique de la gamme.

L'harmonie est l'art de faire jouer plusieurs sons ensemble.

9.1.3 Guide pour le programmeur : creation d'un MicroGénérateur instanciable

L'atout principal du générateur est sa modularité, il est très façile d'ajouter un nouveau micro Générateur. Voila un guide pour le programmeur qui souhaite implementer un nouveau micro générateur.

Conception et Questions reponses Voila la démarche à suivre pour créer un générateur instantiable.

Les conditions à respecter :

- Le micro Générateur doit implementer l'un des quatre microGénérateur abstrait.
- Son nom doit etre "microGénérateurXXX1" avec XXX le nom de l'instrument du micro Générateur, et le numéro du micro–générateur

Que doit faire le programmeur pour créer correctement un microGénérateur ? Simplement redéfinir les méthodes composeMesureXXX qu'il souhaite (avec XXX le type du cycle, solo, couplet ...) (il n'est pas obligé de toute les implémenter car les MicroGénérateur abstrait implémente chaque méthode de base).

Comment composée une mesure ? Le programmeur a à sa disposition toutes les informations nécessaire à la création de la musique dans la super classe. Voici les informations dont le programmeur aura besoin :

methodes et fonction utilise a la creation d'un micro générateur

classe MicroGénérateur Mesure mesureCourant : c'est une référence vers la mesureCourante, la référence est automatiquement changé au changement de mesure int indiceCycleCourant : c'est l'indice du cycle courant. int indiceMesureCourante : c'est l'indice de la mesure courante dans le cycle actuel, le programmeur peut s'aider de cette indice pour cree de la musique sur plusieur Mesure, List<Mesure> Mélodie : c'est la mélodie de la musique, le programmeur peut l'utilise ou non.

La classe TechniquePatern cette classe comporte des méthodes pour effectuer des technique musicale specifique. Les méthodes de cette classe sont :

public static ArrayList<Note> parcourGamme(ArrayList<Note> note,Gamme g,int tonnique,int start,int ecart,int tempsTotal) : cette méthode, très complète, permet au programmeur d'effectuer des monter, des descente, des répétition d'une ou plusieurs note sur une gamme.

ArrayList<Note> note : la liste de note. Gamme g : la gamme sur laquelle va s'effectuer les opérations. int tonique : la tonique sur laquelle la gamme va s'appuyer. int start : le dégrée de la gamme sur lequel les opérations vont débuter. int ecart : représente le décalage de degré après chaque opération. ecart = 1 ;représente une montée d'un degré. les notes seront premièrement jouées a un certain degré (start), a la deuxième génération, ces notes serons jouée sur start + 1, a la deuxiement start + 2. ecart = 0 ; représente une répétition. ecart = -1 ; représente une descente. int tempsTotal : le temps total final des notes en sortie, pour une meilleur utilisation et un meilleur rendu, il est conseillé que : tempsTotal

A noter que l'élément retournée est une ArrayList<Note>, il est donc possible d'effectuer des appel récursif de cette fonctions pour avoir, par exemple, des monter de monter, des répétition de monter etc...

la classe gamme int getNote(int tonique, int interval) : cette fonction revoie la note correspondant a l'interval dans la gamme, avec une tonique spécifique.

Accord getAccordNieme(int nombreTierce, int tonique,int degre, int duree, int temps) : cette fonction renvoi l'accord nieme (nieme sous entend le nombre de tierce que comporte l'accord)

- nombreTierce : le nombre de tierce que comporte l'accord (3 pour un accord parfait, 4 pour un accord 7ieme, 5 pour un accord 9ieme etc...)
- tonique : la tonique sur laquelle est basée l'accord
- degrée : sur quel dégrée est plaqué l'accord.

public int getInterval(int tonique,int note) : renvoie l'interval correspondant a la note.

grâces a ces méthodes, il est très facile pour le programmeur de piocher les informations utiles a la création d'une mélodies, d'une rythmique.

la classe mesure une mesure comporte plusieurs informations.

- La tonique : (la note de base de la gamme)
- le dégrée : le dégrée sur laquelle sera interprétée la mesure
- la gamme actuel : la gamme actuel sur laquelle sera calculé la note correspondante au dégrée

une Énumération modulation : indique (si c'est une mesure représentant une mélodie) comment cette mélodie a été joué, si elle a été modulé ou pas. Toutes ces information, une fois en-capsulée dans la gamme, permettes grâces aux méthodes d'accéder directement a la note tonique actuel, ou a la note correspondant au degré. une mesure comporte une liste de notes.

public int getDifferenceTempsMesure() : cette méthode est primordial pour la composition, elle permet au programmeur de savoir si la mesure est complète ou non.

Exemples Une 10ene de micro Générateur on été prealablement crée . Ils ont chaqu'un des methodes specifiques, voici ce qui les differencies :

les differents micro générateur melodie :

- recopier la melodie tel quel,
- moduler la melodie en fonction de la tonique actuel en suivant une modulation modal.
- moduler la melodie en fonction de la tonique actuel en suivant une modulation tonal.

les differents micro Générateur bass :

- se contente de jouer la tonique actuel en croche.

les differents micro Générateur rythmique :

- accentu les temps forts avec des accords suivant la gamme.
- accentu les temps faibles avec des accords suivant la gamme.
- accentu les notes de la melodie, seulement si ces derniere sont des notes a consonnance forte, et avec une durée minimum d'une noire.

les differents micro Générateur Batterie :

- riff de batteries connue en boucle
- accentu les notes de la melodie, seulement si ces derniere sont des notes a consonnance forte, et avec une durée minimum d'une noire.

9.2 Sources

Bibliographie

[1] J. Dréo, A. Pétrowski, P. Siarry, E. Taillard, Métaheuristiques pour l'optimisation difficile. Ed EYROLLES, 2003.

[2] Le guide de la théorie de la musique, Claude Abromont. Ed Fayard, 2001

`http://fr.wikipedia.org/wiki/Algorithme_évolutionniste`

`http://en.wikipedia.org/wiki/Interactive_evolutionary_computation`

Source le papier du japonais sur comment utiliser les algos évolutionnaire, à citer de son étude la partie qui prévoit l'interaction de l'utilisateur lorsqu'il n'existe pas d'algo efficace pour caractériser une solution, et celle qui doit limiter l'ennui de chez l'utilisateur.

Les tutoriaux de Sun (notamment ceux sur l'API MIDI)