UNIVERSITÉ
CAEN
NORMANDIE

2019

Rapport projet tutoré
Générateur de musique

CASTEL Flavien 21405613
GODEMENT Lancelot 21606808
LEHARIVEL Julien 21404694
PRADAL Romuald 21502222
VOISIN Dan 21502515

Tuteur : LEMIÈRE Valentin
L3 Informatique

Table des matières

La musique, c'est du bruit qui chante.
Victor Hugo

Introduction

Depuis le début de l'existence des sociétés humaines, la musique a tenu un rôle central dans le développement des différentes civilisations. Qu'elle soit utilisée comme indication du rythme de travail, comme divertissement de la haute société ou comme échappatoire de la basse société, la musique, qu'elle que soit son époque ou sa localisation, est une part intrinsèque de l'humain. La musique trouve également une place de choix dans la transmission du patrimoine, des connaissances, ou des légendes, traversant les temps bien plus facilement que la simple diction. Malgré son évolution au cours de l'histoire suite à la création de nouveaux instruments, de nouveau rythmes, de règles, de liberté ou encore de mode, tout type de musique se retrouve sur le fait qu'il se compose de notes, instrumentales et/ou vocales, assemblées et concaténées les unes avec les autres, le tout formant une mélodie plus ou moins construites.

On trouve dans les chansons les plus construites plusieurs plans. Dans la grande majorité, nous observons un plan sonore construisant la base et souvent exécuter par les percussions, un second plan sonore dit «rythmique», reprenant par des notes basses une vision plus ou moins répétitive et simpliste de la mélodie et enfin un ou plusieurs plans dit «lead», reprenant la mélodie en y intégrant de nouveaux éléments, qu'ils fassent partie de ce que l'on peut appeler «riff», «solo», etc. Ces différents plans se spécifient évidement en fonction du style et de l'époque de la musique. En effet, la construction d'un orchestre symphonique n'est évidement pas la même que celle d'un groupe de Hard Rock ou d'un ménestrel du moyen-âge.

Le problème posé était donc bien plus complexe que de simplement générer des notes les unes après les autres, nous avons du faire des choix de façon à pouvoir utiliser plusieurs instruments, comme dans la très grande majorité des chansons.

1 JMusic

JMusic est un projet de développement qui a vu le jour vers la fin des années 90 dans le département musical de la « Queensland University of Technology (QUT) » ayant pour objectif de donner au langage Java les outils lui permettant de composer et analyser de la musique. Il a été conçu de façon à offrir aux compositeurs et développeurs de logiciels un outil solide, extensible et ouvert sous beaucoup d'aspect, que ce soit d'un point de vue objet ou d'un point de vue purement fonctionnel. Il fournit des méthodes d'organisation, de manipulation et d'analyse des données permettant un traitement concret et efficace de piste musicale de divers formats.

Parlons maintenant de la structure de données de JMusic en la comparant à la structure d'une musique en commençant par le niveau le plus élevé, la partition représentée par la classe «Score» qui est utilisé pour contenir les données de la musique. Ces données sont communément générées et/ou lu à partir d'un fichier MIDI bien que dans notre cas nous nous contentons de sauvegarder ces données dans un fichier MIDI en écrivant les données dans ce dernier.
Vient ensuite la partie jouer par un instrument qui est représentée dans JMusic par la classe «Part» et qui est contenu dans un objet «Score». Cette classe contient une liste d'objet «Phrase» provenant de la classe «Phrase» qui représente la liste des notes qui vont être jouer par l'instrument à un moment précis de la composition nous permettant de par exemple donner une liste de note à jouer avec la main droite au piano à partir du premier «beat» et une autre liste à la main gauche à jouer à partir du cinquième «beat».
Enfin, les notes contenues dans l'objet «Phrase» sont représentées par la classe du même nom «Note». Cette classe contient plusieurs informations tel que la durée de la note, sa hauteur et son intensité. Les notes sont jouées dans l'ordre dans lequel elles ont été ajoutées à l'objet «Phrase».

PhraseMatrix prend une «Phrase» pour créer une matrice dans laquelle se trouve chaque note de la phrase en question, ce qui nous permettra par la suite d'appliquer implicitement un calcul pour générer la chaine de Markov. Le paramètre de profondeur permet de définir la taille d'analyse statistique de la phrase, c'est-à-dire combien de notes vont être utilisées comme pattern.

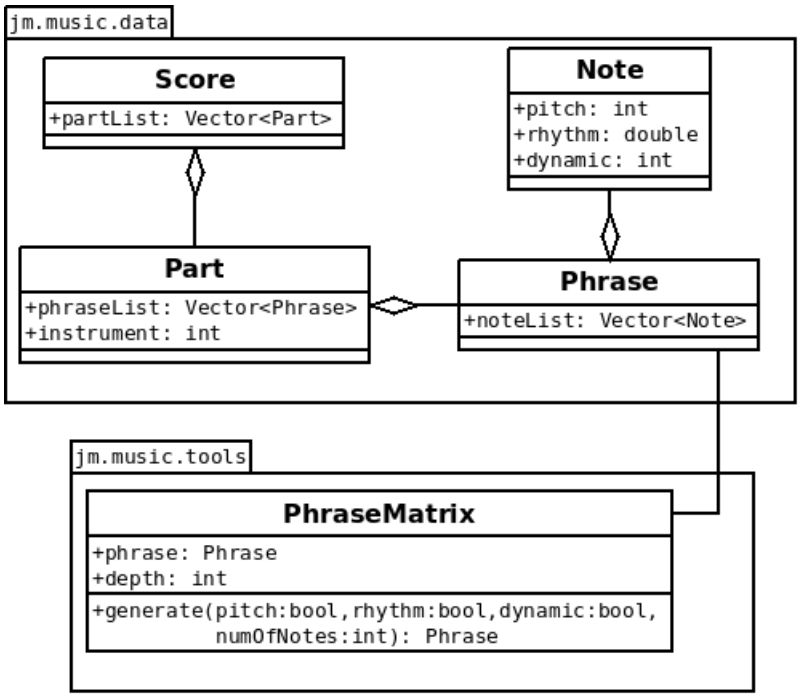

FIGURE 1 – Diagramme UML des classes principalement utilisées

Ainsi une musique sera représentée par un Score qui contient une liste de liste de liste de note.

2 Chaîne de Markov

2.1 Contexte mathématique

Une chaîne de Markov est un processus mathématique à temps discret ou continue et à espace d'états discret. C'est un processus stochastique influençant ses tirages par les résultats des tirages précédents. Il est lié au mouvement brownien et à l'hypothèse ergodique, des sujets importants au XXème siècle. Cet outil mathématique est apparu en 1906, lors de sa publication par Andreï Markov. Il sera repris par la suite par Kolmogorov 30 ans plus tard, qui le généralisera à un espace d'états infini dénombrable.

La propriété qui caractérise particulièrement cette chaîne est que «la prédiction du futur à partir du présent n'est pas rendu plus précise par des éléments d'information supplémentaires concernant le passé, car toute l'information utiles pour la prédiction du futur est contenue dans l'état présent du processus.» (Source : Wikipédia).

De façon analogue, on peut définir une chaîne de Markov dans un langage statistique ainsi : un processus aléatoire portant sur un nombre fini d'états, avec des probabilités de transition sans mémoire. Pour comprendre ce processus, nous citerons l'exemple donné par le site lien brisé.

Exemple de chaine de Markov à deux états :

On pense qu'un individu non endetté a 1/3 possibilité de devenir endetté. Un individu endetté a une possibilité de 1/6 de régler ses dettes.
On représente une chaîne de Markov avec une matrice de transition. Chaque rangée de la matrice correspond à un état et donne la probabilité de passer à un autre état. Dans le cas de notre individu, la matrice de transition est :

$$\begin{bmatrix} 2/3 & 1/3 \\ 1/6 & 5/6 \end{bmatrix}$$

Une matrice de transition se reconnaît parce que toutes les valeurs sont entre 0 et 1 inclusivement et que la somme de chaque ligne est égale à 1.
Naturellement, pour avoir une chaîne de Markov, il faut répéter le nombre de transitions possibles. Dans notre cas, supposons qu'à chaque jour qui passe, la matrice de transition s'applique. Si l'individu n'est pas endetté au départ, après un jour il y aura une probabilité de 1/3 qu'il soit endetté :

$$\begin{bmatrix} 1 & 0 \end{bmatrix} \begin{bmatrix} 2/3 & 1/3 \\ 1/6 & 5/6 \end{bmatrix} = \begin{bmatrix} 2/3 & 1/3 \end{bmatrix}$$

Après deux jours, il aura autant de possibilités d'être endetté que de ne pas l'être :

$$\begin{bmatrix} 1 & 0 \end{bmatrix} \begin{bmatrix} 2/3 & 1/3 \\ 1/6 & 5/6 \end{bmatrix}^2 = \begin{bmatrix} 1/2 & 1/2 \end{bmatrix}$$

Et ainsi de suite. On peut se demander ce qui se passera après un très long délai, disons un an ? Il suffit alors de savoir calculer les puissances de la matrice de transition. Dans tous les cas qui nous concernent, ces puissances convergent rapidement vers une matrice fixe : cela signifie qu'après avoir calculé 2, 3, 10 ou 20 puissances, la matrice ne change pratiquement plus :

$$\begin{bmatrix} 2/3 & 1/3 \\ 1/6 & 5/6 \end{bmatrix}^5 = \begin{bmatrix} 0.34375 & 0.65625 \\ 0.328125 & 0.671875 \end{bmatrix} \qquad \begin{bmatrix} 2/3 & 1/3 \\ 1/6 & 5/6 \end{bmatrix}^{100} = \begin{bmatrix} 0.33333 & 0.66666 \\ 0.33333 & 0.66666 \end{bmatrix}$$

On voit, par ces matrices, qu'après cinq ans ou dix ans, un individu, qu'il débute avec une dette ou sans dette, aura deux chance sur trois d'être endetté !

2.2 Appliqué à la musique

Dans notre cas, notre programme partant d'une «Phrase» de départ et utilisant la chaîne de Markov, il calcule la probabilité d'apparition de chaque note en fonction de celle qui la précède pour ensuite se servir de ces probabilités afin de générer les suivantes. Utilisons une «Phrase» exemple : $do, re, do, si, la, la, re, fa$ et un tableau pour imager le tout.
Nous avons un tableau à double entrées des notes de la Phrase et on regarde combien de fois une note des lignes est suivi par une note des colonnes. Ensuite on fait la somme des notes ligne par ligne.

	do	re	fa	la	si	nombre
do	0	1	0	0	1	2
re	1	0	1	0	0	2
fa	0	0	0	0	0	0
la	0	1	0	1	0	2
si	0	0	0	1	0	1

Grâce à cela nous pouvons ensuite calculer la probabilité qu'une note en suivent une autre.

	do	re	fa	la	si
do	0/2	1/2	0/2	0/2	1/2
re	1/2	0/2	1/2	0/2	0/2
fa	0	0	0	0	0
la	0/2	1/2	0/2	1/2	0/2
si	0/1	0/1	0/1	1/1	0/1

On peut ainsi voir dans cet exemple que le «do» a 50% chance d'être suivi par un «re» ou un «si», ou encore qu'après un «si» le «la» a 100% chance d'apparaitre.

3 Modélisation, conception et complexité

3.1 Diagramme de classe

Afin de mettre en place une solution pour générer de la musique grâce à une chaine de Markov, nous avons opté pour la modélisation suivante :

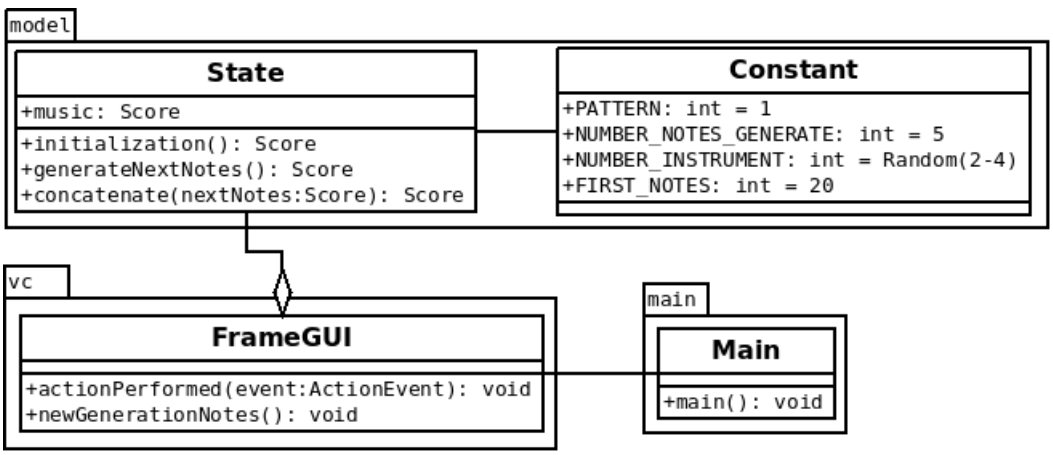

FIGURE 2 – Diagramme UML du projet

L'état va contenir la musique à l'instant t, et grâce à la méthode generateNextNote, l'état t va générer une possibilité calculée par la chaîne de Markov. On utilise une interface Constant qui va

contenir les variables globales du programme. Enfin la classe FrameGui sert pour définir l'interface graphique.

Au niveau de l'interface, en haut nous avons un lecteur pour l'état actuel, et en-dessous quatre lecteurs pour les propositions générées.

FIGURE 3 – Interface graphique

Ainsi à chaque fois qu'on choisit une proposition, on concatène la proposition à la musique de l'état actuel et les propositions sont recalculées.

3.2 Algorithmes et complexité

3.2.1 generate(bool p, bool r, bool d, int numOfNotes)

Lors de cette partie nous allons analyser quelques-uns des algorithmes utilisés.

Dans ce but, nous expliqueront ci-dessous le fonctionnement de la méthode generate(bool p, bool r, bool d, int numOfNotes) qui permet d'obtenir un objet Phrase et qui est appelé par l'instruction PhraseMatrix.generate(bool, bool, bool, Phrase). A l'initialisation, notre fonction prend en trois arguments booléens :

- Le pitch ou hauteur des notes correspond à l'autorisation de calculer des hauteurs de notes par rapport à celle déjà existantes.
- Le rythme ou la vitesse d'une note correspond à l'autorisation de calculer des rythmes de notes à partir à celle déjà existantes.
- La dynamique est la différence de décibels (dB) entre les différentes notes jouées. Si ce booléen possède la valeur vrai alors nous garderons la même concordance de la valeur, c'est-à-dire que les notes suivantes seront sur le même plan que celle déjà existentes. Si ce booléen est à faux, alors le rythme aura une valeur par défaut pour les nouvelles notes.

Comme expliqué ci-dessus, ces trois booléens correspondent au fait de rester dans la continuité de la phrase en cours, afin d'utiliser la chaine de Markov. Revenons à notre méthode generate(Boolean p, Boolean r, Boolean d, int numOfNotes). Le dernier argument est un entier qui correspond au nombre de notes que l'on souhaite générer.
Ensuite, notre fonction va initialiser les tableaux de variables pitch, rythme et dynamique, le tout de taille numOfNotes. Ces tableaux vont être remplis par le processus de Markov grâce aux information de la phrase, puis une nouvelle note sera générée grâce au parcourt des trois tableaux.

Complexité

Soit n le nombre de note.
La complexité au pire de cette méthode est : O(n*m)

3.2.2 concatenate(Score)

Cette fonction commence par instancier un objet Score en fonction de l'état actuel de son contenant. Elle effecte ensuite un parcours sur chaque note de chaque instrument.
Grâce à cela, nous obtenons une phrase à laquelle nous pouvons ajouter un tableau de notes, et ce pour chaque instrument utilisé.
Au final on a un nouvel état possédant la musique de son père avec la proposition à la fin de la musique.

Complexité

Soit n le nombre d'instrument.
Soit m le nombre de note dans une musique
Comme un instrument à m notes
La complexité au pire de cette méthode est : O(n*m)

3.2.3 initialization()

Initialization permet de générer un état de manière aléatoire

Complexité

Soit n le nombre d'instrument.
Soit m le nombre de note dans une musique
Comme un instrument à m notes
La complexité au pire de cette méthode est : O(n*m)

3.2.4 generateNextNote()

Cette méthode va générer une possibilité de suite de la musique en utilisant generate(Boolean p, Boolean r, Boolean d, int numOfNotes).

Complexité

Soit k le nombre d'instrument.
Soit m le nombre de note dans une musique
On sait que la complexité de generate(Boolean p, Boolean r, Boolean d, int numOfNotes) est O(n)
Comme un instrument à m notes
La complexité au pire de cette méthode est : O(k*n*m)

Ce que l'on a implémenté est plutôt «léger» et s'exécute rapidement de manière linéaire.
Au vue de la complexité, on peut dire que pour des pistes de petites tailles, cela est réalisable. Toutefois, l'exécution commence à être compliquée quand on veut l'appliquer à un projet de grande envergure tel un concerto.

4 Difficulté

Au cours de ce projet, nous nous sommes heurtés à un certain nombre de difficulté. Premièrement, nous avions pour consigne d'utiliser JMusic. C'est certes un outil adapté à la composition par ordinateur, mais comme expliqué dans la partie qui lui est consacré, il est né d'un projet de la fin des années 90. Le site officiel de cet outil date lui-même de 2001, tout comme sa documentation. De ce fait, le moindre petit problème résulte vers des heures de déchiffrage de document ancien à l'échelle de l'informatique et trop souvent peu précis. De façon analogue, nous avons constaté le manque d'exemple et d'évolution depuis sa mise en ligne, ce qui ne facilite pas son utilisation.
À contrario, nous nous trouvons exactement dans la situation inverse sur le sujet de la chaîne de Markov. En effet, c'est un standard des mathématiques prévisionnelles depuis plus de 110 ans, ce qui explique la grande abondance de document, chacun apportant sa propre vision du processus, ainsi que son interprétation. Il faut bien entendu aborder les imprécisions récurrentes entre différents processus, ainsi que les différentes applications associées au processus original. Il est d'autant plus compliqué de définir clairement cette chaîne, que celle-ci est souvent présenté sous des visions différentes, qu'elles soient purement mathématique, statistique, ou autre.

5 Pour aller plus loin

Visualisez la scène. Vous vous êtes décidé à vous rendre à un concert de musique, mais quel n'est pas votre surprise de constater que tous les musiciens et compositeurs sont purement robotiques. Pire, vous avez devant vous des génies aujourd'hui disparu, de retour pour composer à nouveau, à l'instar du film Janis et John. Si cette scène pourrait être tout droit sorti d'une œuvre de science-fiction, elle n'est pourtant pas si irréaliste. Ces dernières années, un certains nombres de chercheurs ont creusé le sujet de la composition par intelligence artificiel, même si très peu de leurs résultats sont arrivés aux oreilles avares des médias généralistes.

Pour commencer, nous allons parler d'une IA dont la spécialité est de composer sur commande. Jukedeck `https://www.jukedeck.com/`judedeck est un site web permettant en fonction des besoins de son utilisateur de générer de la musique composé sur commande. Si les compositions ne sont pas de pures chefs d'œuvres, elles permettent tout de fois d'obtenir un résultat tout à fait audible en un minimum de temps et de travail, moyennant une nouvelle génération si la piste n'est pas convaincante. C'est un outil ouvert à tous, qu'ils soient développeurs, entreprises ou simple curieux.

Un des pionniers du secteur, Melomics a fait son apparition en 2012. Créé par l'espagnol Francisco Vico, cet IA est capable de recréer l'ambiance des compositions au piano du début du XXème siècle. Ianus, le culster informatique de l'université de Malaga a été reconnu comme produisant quelque chose de très réaliste, mais ce n'est rien par rapport à son petit frère Melomics 109, qui a sorti son premier album en 2014 (disponible à l'écoute gratuitement). Que l'on aime ou pas, cette IA propose des compositions d'un genre beaucoup plus « pop » que son prédécesseur et propose des titres largement à la hauteur d'un « tube de l'été ». D'après leurs créateurs, ce système utilise les mêmes étapes de création qu'un réel compositeur, ce qui a attiré l'attention de la BBC.

Aiva `https://www.aiva.ai/`aiva est une IA spécialisé dans la composition symphonique, un sujet bien plus délicat qu'un simple « tube de l'été », d'autant plus que c'est un secteur musicalement beaucoup plus contraignant et complexe. Inspiré par le film de Science-Fiction Her, cette IA est construite de 2 éléments. D'un côté, plus de 15 000 partitions des plus grands génies de la musique savante occidentale, de l'autre une méthode d'apprentissage basée sur des réseaux de neurones artificiels. La machine analyse les tablatures et grâce à un algorithme, peut produire en quelques jours une symphonie en fonction des demandes effectuées. La première production d'Aiva date de 2016, mais la renommée de cet IA ne cesse de grandir, au point qu'elle a officiellement été reconnue comme

compositeur à part entière par la SACEM et a composé en 2017 un morceau pour la fête nationale du Luxembourg.

Vous connaissez la chanson Daddy's car des célèbres Beatles ? Ne cherchez pas, même si cette chanson possède toutes les caractéristiques des œuvres du mythique groupe des années 60, elle est en fait générée par une intelligence artificielle nourrit aux compositions du groupe : Flow Machines `http://www.flow-machines.com/`flow-machines. C'est une IA créée par François Pachet chercheur et directeur du laboratoire français Sony Computer Science Laboratory. Comment la machine parvient-elle à simuler le retour des « Fab Four » ? Grâce à un système d'apprentissage, elle consulte et analyse plus de 14.000 «lead sheet» du groupe et de musique de l'époque. Si elle n'est pas pour le moment suffisamment aboutie pour se passer de la main humaine, puisque les paroles et le mixage sont de François Pachet. Malgré ça, un album a été annoncé en 2017 et les quelques créations d'un niveau professionnel ont coiffé Google et son projet Magenta au poteau de l'innovation.«J'espère que l'on va remettre un peu de chaos, d'inventivité et d'audace dans la chanson populaire», conclue le créateur sur Trax.

Dernier point qui nous ramène tout droit vers les œuvres à l'influence SF des années 80 : les droits à la propriété intellectuel. Si une bonne partie des équipes de ces IA considèrent leurs créations comme des œuvres libres de droit, d'autre, comme Jukedeck, ne sont pas de cet avis et considère les droits sur les créations de leur « créature » comme les leurs. Or, Le tout premier article du code de la propriété intellectuelle dispose en effet que pour être protégée au titre des droits d'auteur, une œuvre doit être une « œuvre de l'esprit ». La jurisprudence nous dit que n'est « de l'esprit » que ce qui est original, ce qui porte « l'empreinte de la personnalité » de l'auteur. Seulement, difficile de dire qu'une intelligence artificielle possède une réelle personnalité, tout du moins en ce début d'année 2019. C'est sans aucun doute un point qui fera polémique dans le futur. De la même façon, nous pouvons sans grand risque prophétiser une guerre sainte entre les « pro production d'intelligence artificielle » et les « puristes de la composition ».

Conclusion

Notre application n'est pas dans le futur. Tout de fois elle montre l'implication des mathématiques dans la musique, mais avant tout le sens mathématique des gouts des êtres humains. En effet, tout type de musique, quelque-soit son époque ou sa localisation, possède un modèle mathématique et des propriétés qui, à grande échelle, permettent de représenter celle-ci selon un modèle mathématique. Nous avons parlé des avancées technologiques de ces dernières années, utilisant des intelligences artificielles douées d'apprentissages et pour certaines inspirées des connections neuroniques humaines. Notre modeste application n'en arrive certes pas là, mais son processus d'aide à la composition n'en reste pas moins proche de par son action, grâce aux prévisions de la chaîne de Markov, à l'échelle de notre projet tout du moins.

Nous avions à appliquer la chaîne de Markov à l'outil Java JMusic afin de développer un générateur de musique. Notre travail nous a amené à produire une application Java permettant la composition d'une piste appelée « LEAD » à partir de choix fais entre des pistes générées automatiquement grâce à une chaîne de Markov. Nous avons tous beaucoup appris de ce projet, que ce soit dans l'automatisation du recours à la documentation officiel, dans le développement pure, le travail d'équipe, ou enfin dans une vision évolutive de notre quotidien et des choses banales qui nous entourent.