UNIVERSITÉ FRANÇOIS RABELAIS DE TOURS

École Doctorale SST

Laboratoire d'Informatique (EA2101)
Équipe Handicap et Nouvelles Technologies

THÈSE présentée par :

Romain Clair

soutenue le : 6 décembre 2010

pour obtenir le grade de : Docteur de l'université François Rabelais de Tours

Discipline/ Spécialité : Informatique

Étude de méthodes de production d'art génératif et de leur application pour la conception d'outils de création artistique accessibles.

THÈSE DIRIGÉE PAR :

Slimane Mohamed	Professeur des universités, Université François Rabelais de Tours

RAPPORTEURS :

Lopez Krahe Jaime	Professeur des universités, Université Paris VIII
Lutton Evelyne	Chargée de recherche (HDR), Inria Saclay

JURY :

Assayag Gérard	Directeur de recherche, Ircam - CNRS
Lopez Krahe Jaime	Professeur des universités, Université Paris VIII
Lutton Evelyne	Chargée de recherche (HDR), Inria Saclay
Monmarché Nicolas	Maître de conférences, Université François Rabelais de Tours
Slimane Mohamed	Professeur des université, Université François Rabelais de Tours
Talbi El-Ghazali	Professeur des universités, Université de Lille I

Remerciements

Les travaux présentés ici n'auraient jamais pu aboutir sans le concours de nombreuses personnes. Parmi celles-ci, mes encadrants, Mohamed Slimane et Nicolas Monmarché, sans qui rien de tout cela n'aurait été possible. En me proposant ce sujet, ils m'ont permis de concilier les domaines pourtant réputés antagonistes que sont les arts et la science. Ils m'ont par ailleurs offert les conditions de travail dont j'avais besoin pour mener ces recherches à bien, et pour tout cela, je les remercie.

L'intégration au sein de l'équipe Handicap et Nouvelles Technologies et du Laboratoire d'Informatique de Tours m'a permis de côtoyer des gens passionnants et passionnés par leur métier. Leur contact et les échanges que nous avons pu avoir ont été particulièrement formateurs.

Merci aussi à tous ceux qui ont bien voulu se livrer à l'évaluation des outils proposés. Leurs remarques ont été précieuses. Ils sont beaucoup trop nombreux pour être tous cités, mais je tiens néanmoins à remercier Pascal Lenoir, Gaël Bouket et l'équipe du centre d'accueil spécialisé de Ballan Miré pour leur temps, leur enthousiasme et les conseils qu'ils m'ont donnés. Mes remerciements vont aussi à Alain Delime et aux organisateurs du challenge Handicap et Technologie ainsi qu'à l'équipe des Petits Débrouillards de Région Centre pour m'avoir accueilli lors de leurs manifestations, et m'avoir, ainsi, permis de présenter mes travaux à de nombreux testeurs.

Je remercie tout ceux, rencontrés lors des conférences et dont les manifestations d'intérêt ont contribué à entretenir la motivation nécessaire à mener ces recherches à terme.

Des remerciements tous particuliers vont vers Evelyne Lutton et Jaime Lopez Krahe qui ont eu la gentillesse d'accepter la lourde charge d'être les rapporteurs de cette thèse, ainsi que Gerard Assayag et El-Ghazali pour leur participation au jury.

Je tiens en dernier à remercier ceux, de ma famille et de mes amis, qui m'ont soutenu, supporté, et pour certains, énormément aidé pendant ces années de thèse, ils se reconnaitront et savent que ma gratitude leur est acquise.

Résumé

Partant d'une étude de méthodes algorithmiques de création artistique automatique, nos travaux s'intéressent à des adaptations de ces méthodes pour proposer des formes de création artistique accessibles centrées sur la musique et le dessin par ordinateur.

Le développement d'outils informatiques accessibles impose certaines contraintes. Elles servent de cadre au développement de deux outils qui illustrent la mise en œuvre de ces contraintes, associées à l'adaptation de systèmes de production artistique automatique, notamment basés sur des algorithmes de colonie de fourmis artificielles. Le premier programme est un instrument de musique virtuel permettant au plus grand nombre de jouer de la musique, et fournissant un accompagnement automatique. Le second est un atelier de dessin où des outils basés sur des méthodes génératives offrent un résultat complexe à partir d'actions très simples. Cette thèse détaille le développement de ces deux programmes ainsi que leur évaluation, sur le terrain, à la rencontre d'utilisateurs réels.

Mots clés : Accessibilité, Handicap, Art génératif, Algorithmes de colonie de fourmis artificielles, Algorithmes évolutionnaires, Musique, Dessin

Abstract

Starting from a survey of algorithimic methods for automatic artistic creation, our works deals with adaptations of these methods to provide accessible art creation forms, focusing on music and computer drawing.

the development of accessible computer software requires some constraints. Following these constraints and using adapted automatic art production systems, notably based on artificial ant colony algorithms, we developed two computer programs. The first one is a virtual music instrument, allowing most people to play music and providing and automatic accomapniement. The second one is a drawing workshop with generative methods-based tools provide complex results from simple actions. This PhD thesis details the development of this two programs and their evaluations, with real users meetings.

Keywords : Accessibility, Handicap, Generative art, Artificial ants colony algorithms, Evolutionnary computation, Music, Drawing

Table des matières

Introduction **17**

I État de l'art **21**

1 De la création artistique **25**
1.1 Petite introduction à la théorie musicale 25
1.1.1 La norme MIDI 29
1.2 Dessin et ordinateur 31
1.3 Art et handicap 31

2 Modèles et méthodes de résolution génériques **33**
2.1 Chaînes de Markov 33
2.2 Grammaires génératives 34
2.3 Réseaux de neurones artificiels 35
2.4 Automates cellulaires 36
2.5 Algorithmes génétiques 37
2.6 Systèmes multi-agents 38
2.7 Algorithmes de colonie de fourmis artificielles 38
2.7.1 Le déplacement des fourmis 39
2.7.2 Le tri du couvain 40
2.7.3 Partage de tâches chez les fourmis 41

3 De l'art génératif **43**
3.1 Quelques considérations générales sur l'art génératif 43
3.2 Étude de méthodes de génération de musique 44
3.2.1 Modèles de Markov 46
3.2.2 Grammaires génératives 47
3.2.3 Automates cellulaires 48
3.2.4 Programmation par contrainte 50

3.2.5 Réseaux de neurones artificiels . . . 52
3.2.6 Algorithmes évolutionnaires . . . 53
3.2.7 Systèmes multi-agents . . . 58
3.2.8 Autres méthodes bio-inspirées . . . 59
3.3 Étude de méthodes de génération de dessins . . . 60
3.3.1 Grammaires génératives . . . 60
3.3.2 Automates cellulaires . . . 60
3.3.3 Algorithmes évolutionnaires . . . 61
3.3.4 Système multi-agents . . . 63
3.3.5 Algorithmes de colonie de fourmis artificielles . . . 63

4 Accessibilité des logiciels 65
4.1 Interface Homme-Machine . . . 66
4.1.1 Affichage de l'information . . . 67
4.1.2 Contrôle de l'application . . . 68
4.2 Accompagnement et assistance . . . 70

II Les travaux 73

5 Un instrument de musique virtuel 77
5.1 Présentation et objectifs . . . 77
5.1.1 Accessibilité des interfaces . . . 78
5.1.2 Pas de pré-requis technique . . . 78
5.1.3 Pas de connaissances musicales requises . . . 78
5.1.4 Accompagnement automatique . . . 79
5.2 Les solutions apportées . . . 79
5.2.1 Utilisation de la norme midi . . . 79
5.2.2 Connaissances musicales embarquées . . . 82
5.2.3 Les interfaces . . . 83
5.2.4 Accompagnement automatique . . . 84
5.3 Évaluation et résultats . . . 90
5.3.1 Influence des paramètres . . . 90
5.3.2 Efficacité des interactions . . . 92
5.3.3 Évaluation de terrain . . . 93
5.3.4 Perspectives et évolutions . . . 96

6 Atelier de dessin 99
6.1 Présentation et objectifs . . . 99

6.2 Solutions adoptées 101
6.2.1 Algorithme de fourmis artificielles 104
6.2.2 Fractales 105
6.2.3 L-System 108
6.3 Résultats et perspectives 108

Conclusion **115**

Annexes **121**

A Références techniques **121**

Liste des tableaux

1.1 Tableau des échelles temporelles de la musique 26
1.2 Table des intervalles 27
1.3 Table des figures rythmiques 28
1.4 Types de messages MIDI 30

3.1 Jeu de construction de Valses de Mozart 45

5.1 Table des notes disponibles 81
5.2 Paramètres de l'algorithme 87
5.3 Taille des graphes 89
5.4 Influence du nombre de fourmis - paramètres. 91
5.5 Influence du nombre de fourmis - résultats. 91
5.6 Influence du coefficient d'évaporation ρ - paramètres. 92
5.7 Influence du coefficient d'évaporation ρ - résultats. 92
5.8 Influence de τ_u, interaction moteur-utilisateur - paramètres. 93
5.9 Influence de τ_u, interaction moteur-utilisateur - résultats. 93
5.10 Résultats des tests de terrain 95

6.1 Paramètres de l'outil algorithme de fourmis 104

A.1 Instruments *General MIDI*-1 122
A.2 Instruments *General MIDI*-2 123
A.3 Instruments *General MIDI*-3 124
A.4 Instruments *General MIDI*-4 125
A.5 Codes des percussions (canal 10) – 1 126
A.6 Codes des percussions (canal 10) – 2 127

Table des figures

1.1 Gamme chromatique de do 27
1.2 Gamme de do majeur 27
1.3 Gamme de do mineur harmonique 28
1.4 Gamme pentatonique de do mineur 28

2.1 Schéma de principe d'une chaîne de Markov 34
2.2 Schéma d'un neurone formel 35
2.3 Schéma d'un réseau de neurones artificiels 36

3.1 Valse produite avec le jeu musical de Mozart 44
3.2 Schéma de calcul des positions et durées de notes de *CAMUS* 49
3.3 Schéma de fonctionnement de *CAMUS* 49
3.4 Partition produite par *Adaptative Systems Music* 51
3.5 Interface de séléction de *Sbeat* 56
3.6 Exemple de motifs produits par K. Bentley 61
3.7 Danseuse d'Emmanuel Cayla produit avec *ArtiE-Fract* 62
3.8 Papier peint de Marie-Amélie Porcher produit avec *ArtiE-Fract* 63

5.1 Schéma des modules du logiciel 80
5.2 Schéma du système MIDI 82
5.3 Schéma de l'utilisation du clavier 84
5.4 Schéma du mécanisme d'avance accompagnement-lecture 90
5.5 Suite expérimentale et graphe résultant 91

6.1 Exemple de menu déroulant 100
6.2 Interface de l'atelier de dessin : menu principal 101
6.3 Interface de l'atelier de dessin : rosace de choix des couleurs 102
6.4 Interface de l'atelier de dessin : menu Pop-up 103
6.5 Interface de l'atelier de dessin 103
6.6 Outil fourmis : exemple avec deux fourmis déposant du bleu 105

6.7 Exemple de tableau de couleurs de fractales 106
6.8 Outil fractale de Mandelbrot : exemple 107
6.9 Outil fractale de Julia : exemple 107
6.10 Outil L-System 1 : exemple 109
6.11 Outil L-System 2 : exemple 110

Introduction

Cette thèse traite de méthodes de production d'art génératif et de leur application pour la conception d'outils de création artistique accessibles. Ce cadre général, assez large, sort des sentiers battus et propose d'explorer des pistes inhabituelles. Il s'intègre néanmoins aux thèmes de recherches de l'équipe Handicap et Nouvelles Technologies. Les recherches développées ici, se situent, ainsi, au croisement de plusieurs domaines.

Tout d'abord, précisons que ces travaux s'intègrent pleinement dans le cadre de l'informatique. La nature de leur objet amène à s'intéresser à d'autres domaines, celui de l'art, et celui, vaste et par essence pluri-disciplinaire, du handicap. Néanmoins, les solutions apportées sont des solutions informatiques. Et si parfois certaines questions, liées à ces champs d'application, peuvent être soulevées et ne pas relever du domaine informatique. Les réponses sont à chercher auprès d'experts de ces autres disciplines.

Ces recherches s'inscrivent dans la continuité de celles menées par Nicolas Monmarché sur l'exploration des capacités artistiques des algorithmes de colonie de fourmis artificielles [Monmarché et al., 2007]. Ces méthodes produisent des systèmes d'une organisation complexe résultant de l'interaction entre des individus au comportement simple. Cette complexité peut être assimilée à une forme d'art, auto-générée, comportant une part imprévisible.

Il est ainsi question de création artistique, plus précisément, de création artistique automatique ou semi-automatique. De nombreuses méthodes algorithmiques ont été étudiées, visant à produire ce qu'on appelle, souvent, de l'art génératif. Il s'agit d'algorithmes, comportant en général une part de hasard, et dont l'exécution génère une œuvre artistique. Cette exécution peut éventuellement faire intervenir un utilisateur, la méthode étant alors qualifiée de semi-automatique, ou être totalement automatique. La création générée peut être vue comme étant celle de l'auteur de l'algorithme, celle de l'utilisateur du programme, qui en a choisi les paramètres, ou même celle du programme lui-même. Aujourd'hui encore, l'art génératif est un terrain d'expérimentation pour les artistes comme pour les scientifiques.

Ces travaux comportent, aussi, la notion d'accessibilité. Au delà de la simple question éthique ou morale, l'accessibilité est un enjeu important et actuel. L'ONU estime qu'environ 10% de la population mondiale souffre d'un handicap, ce chiffre étant plus important dans les pays disposant d'une espérance de vie plus longue [ONU, 2010]. Cette part importante de la population, souvent négligée, commence enfin à être prise en considération, notamment par la mise en place de lois imposant sa prise en compte. En France, la loi 2005-102 du 11 février 2005 pour l'égalité des droits et des chances, la participation et

la citoyenneté des personnes handicapées, le *Americans with Disabilities Act* de 1990 aux États-Unis ou le *Disability Discrimination Act* de 1995 au Royaume-Uni sont autant de preuves de l'importance de se consacrer à ces problèmes.

Toutefois, l'accessibilité est une notion souvent méconnue et mal définie pour les non-spécialistes, aussi, la définition qui suit permet de préciser les concepts sous-jacents à cette notion, telle qu'elle est envisagée dans le cadre de cette thèse.

La définition de l'accessibilité dépend du contexte auquel elle s'applique. Dans le domaine du web, on peut citer la définition servant d'introduction à la *Web Accessibility Initiative* (WAI), le groupe de travail du *World Wide Web Consortium* (W3C) autour de l'accessibilité du web :

> *« L'accessibilité du web signifie que les personnes handicapées peuvent l'utiliser. Plus spécifiquement, elle signifie que ces gens peuvent percevoir, comprendre, naviguer, interagir avec le web, et y contribuer. L'accessibilité du web bénéficie également à d'autres, notamment les personnes âgées ayant des capacités diminuées dues au vieillissement. »*
>
> [Henry and the Education & Outreach Working Group (EOWG), 2005]

Certains auteurs poussent même la définition plus loin, notamment au-delà de la question du handicap, tel que Tim Berners-Lee, président du *World Wide Web Consortium* (W3C) :

> *« Mettre le Web et ses services à la disposition de tous les individus, quel que soit leur matériel ou logiciel, leur infrastructure réseau, leur langue maternelle, leur culture, leur localisation géographique, ou leurs aptitudes physiques ou mentales. »*
>
> [Berners-Lee, 1997]

La notion d'accessibilité est bien sûr associée à celle de handicap, sans y être strictement limitée. Les handicaps se classent généralement en deux grandes catégories : physiques et mentaux.

Les handicaps physiques touchent aux capacités motrices ou sensorielles. Ils englobent toutes les difficultés liées à la perception de notre environnement, qu'elles soient visuelles, auditives, olfactives, gustatives ou tactiles. La perception classique d'une application informatique passant principalement par les deux premiers de ces sens. Ce sont donc à ceux-ci que l'on s'intéresse en priorité ici. Les handicaps moteurs, quant à eux, recouvrent la très vaste étendue des problèmes liés aux mouvements du corps et à la fatigue associée. L'utilisation d'un ordinateur concerne principalement les membres supérieurs, de l'épaule aux doigts, aussi nous attacherons-nous plus spécifiquement aux handicaps de ces parties du corps. Il existe bien d'autres handicaps, tels que la mutité, les déficiences viscérales, parfois même certaines déformations morphologiques qui bien que lourds de conséquences dans la vie quotidienne ne sont pas des freins majeurs à l'usage de l'outil informatique.

Les handicaps mentaux sont liés aux déficiences intellectuelles. Cette catégorie regroupe une grande variété de difficultés qui peuvent affecter les capacités de raisonnement, de mémorisation, de concentration, de communication, etc. Nombre de ces handicaps, si difficiles à classer, rendent impossible l'utilisation d'outils informatiques. Il est pourtant des cas

dans lesquels des adaptations peuvent permettre un accès à l'informatique. Ces outils participent, alors, au confort voire au développement des capacités de ces personnes. Parfois, l'assistance d'une tierce personne reste nécessaire à l'utilisation de moyens informatiques.

Hors du strict cadre du handicap, il existe d'autres empêchements que l'accessibilité peut chercher à surpasser. Il peut s'agir de problèmes de langue, de contexte culturel ou de manque de formation. La barrière de la langue constitue un obstacle évident, heureusement de nos jours assez habituellement surpassé. Les différences culturelles sont plus délicates à appréhender, et peuvent, elles aussi, conduire à de profondes incompréhensions. Plus spécifiques, les freins liés à la familiarité de l'utilisateur avec le sujet abordé ou avec l'informatique en général sont aussi à prendre en considération. Certaines notions ou pratiques qui semblent naturelles à un expert ou un habitué ne le sont pas pour tout le monde.

L'accessibilité peut aussi recouvrir des considérations techniques ou économiques. Évidemment, l'utilisation d'un outil informatique nécessite de disposer d'un ordinateur... Mais peut-être n'est-il pas rigoureusement nécessaire d'attendre que les moyens matériels disponibles correspondent aux dernières avancées technologiques du moment, et que tous les dispositifs annexes soient à la portée de toutes les bourses. Les pré-requis matériels doivent être limités à un ordinateur simple muni des périphériques usuels.

Ainsi, la notion d'accessibilité considérée dans la suite de cette thèse, correspond à un idéal regroupant l'ensemble de ces notions, qui, bien qu'utopique et inatteignable en totalité dans la pratique, sert d'objectif et de guide dans la conception d'applications utilisables par le plus grand nombre.

Les fondements de ces recherches se situent donc à la jonction entre ces trois domaines, et visent à proposer des formes de créations artistiques accessibles utilisant l'outil informatique. Plus précisément, nous nous intéressons à ces deux domaines artistiques que sont la musique et dessin. Ils sont étroitement liés à l'informatique depuis que les progrès dans ce domaine ont permis aux ordinateurs d'afficher des images et de produire des sons. Ils le sont, notamment, bien plus que d'autres domaines artistiques, tels que la danse ou la littérature, dans lesquels l'utilisation de l'informatique est plus limitée. Ces deux aspects de la création artistique sont aussi complémentaires vis-à-vis de l'accessibilité, notamment puisque le premier met principalement à contribution l'ouïe, quand le second est plutôt basé sur la vue. C'est d'ailleurs sous ces deux aspects que des travaux ont été précédemment menés par notre équipe de recherche [Monmarché et al., 2007].

Ces recherches s'inscrivent dans le cadre de recherches exploratoires sur un sujet neuf, les liens entre ces trois différents domaines, de l'informatique, de l'accessibilité et de la création artistique, n'ayant pas été réellement explorés jusqu'alors. Dans ce genre de cas, plusieurs approches peuvent émerger. La première consiste à essayer le maximum d'idées, procédant ainsi à une sorte de balayage en largeur du sujet, en gardant pour plus tard, ou en laissant à d'autres, le soin d'approfondir chacune de ces pistes. La seconde approche, celle qui a été adoptée ici, vise plutôt à proposer, dans un premier temps, des concepts d'ordre généraux. Dans un second temps, en se basant sur eux, elle explore dans le détail quelques propositions de développement de ces concepts. L'étude de ces pistes procède donc, depuis les idées générales jusqu'à leur réalisation pratique, en incluant bien sûr tests, évaluations et retours sur leur conception. Cette approche permet de rencontrer la majeure partie des problèmes, notamment techniques, qui surviennent à chaque phase du projet et de leur

apporter une solution. Elle aura aussi permis de se faire une idée de l'adéquation entre des idées issues du domaine de la recherche et les besoins pratiques d'utilisateurs réels, notamment handicapés. Ce public spécifique demande d'ailleurs une attention particulière parce qu'il a des attentes et des besoins parfois très différents des personnes ordinaires. Évidemment, puisqu'il s'agit d'un sujet neuf, de nombreuses autres pistes ne sont pas explorées, et restent donc à parcourir.

Cette thèse comporte deux grandes parties. La première dresse un bilan des recherches bibliographiques effectuées. Elle se découpe en quatre chapitres. Le premier pose les bases artistiques et techniques nécessaires pour la suite. Le second s'intéresse plus particulièrement aux concepts généraux de certaines méthodes de résolution. Ces méthodes sont adaptées par les auteurs étudiés dans le troisième chapitre, pour créer des systèmes de production automatique de musique, puis de dessin. Enfin, le dernier chapitre traite des bonnes pratiques en matière d'accessibililté.

La seconde partie détaille, quant à elle, les travaux menés dans le cadre de cette thèse. D'abord ceux qui consistent à concevoir, développer, tester et évaluer un instrument de musique virtuel accessible. Ensuite ceux concernant un atelier de dessin accessible incorporant des outils génératifs. Le tout s'achève, bien sûr, par les conclusions à tirer de ces travaux et les perspectives qu'ils ouvrent.

Première partie

État de l'art

Cette première partie fait état des travaux de recherches bibliographiques effectués dans le cadre de cette thèse. Dans un premier chapitre, nous définissons la création artistique telle que nous la considérons dans ces travaux. Nous posons, tout d'abord, quelques notions de bases de musique, ainsi qu'une description de la norme MIDI qui est fréquemment utilisée par les auteurs cités ici, ainsi que dans nos travaux. Nous abordons ensuite quelques considérations techniques liées au graphisme. Puis nous terminons ce chapitre par quelques mots sur les relations entre la création artistique et le handicap.

Le second chapitre pose les bases de méthodes génériques classiquement utilisées pour résoudre des problèmes informatiques. Ces méthodes étant utilisées dans la suite, ce chapitre permet d'en définir les notions fondamentales communes.

Le troisième chapitre traite plus précisément des systèmes de production d'art génératifs. Ces systèmes emploient et adaptent les méthodes décrites précédemment pour de la génération automatique de musique, d'abord, de dessin, ensuite.

Le dernier chapitre s'intéresse à divers travaux ayant trait à l'accessibilité en informatique. Ils servent à définir un ensemble de bonnes pratiques à adopter dans le cas de la conception de logiciels accessibles.

Chapitre 1

De la création artistique

La création artistique est l'un des deux domaines d'application des propositions détaillées dans cette thèse, aussi, il convient en premier lieu de préciser les concepts que nous allons aborder. L'art est un sujet incroyablement vaste et particulièrement difficile à appréhender par une approche scientifique. En effet, le problème même de sa définition est hors du champ d'application de la science.

Les définitions qui suivent ne cherchent donc pas à englober la totalité de ce qu'est l'art, mais plutôt à préciser l'aspect considéré dans ces travaux. Néanmoins, face à un domaine si étendu et flou, il convient de s'imposer des limites. La création artistique abordée dans les projets qui suivent concerne uniquement deux aspects : la musique et le dessin.

1.1 Petite introduction à la théorie musicale

Considérons, d'abord, la définition minimaliste suivante :

> Ensemble de sons et de silences organisés dans le temps

Cette définition, très proche de celle que donne John Al Biles en introduction de ses travaux [Biles, 2007a], essaie de restreindre le moins possible ce qui peut être considéré comme de la musique. Plus pragmatiquement, Curtis Roads [Roads, 2004] définit, quant à lui, différentes approches de la musique suivant l'échelle temporelle considérée. Ces considérations sont regroupées dans le tableau Tab. 1.1. Des différences existent entre les différents niveaux où l'on peut se situer, notamment entre l'approche macro, sur des durées de l'ordre de la seconde ou plus, et l'approche micro à l'échelle de divisions de la seconde. Myriam Desainte-Catherine et Sylvain Marchand [Desainte-Catherine and Marchand, 1999] décrivent ces différences entre ce qu'ils qualifient d'approche musicale et d'approche électroacoustique. Ils proposent d'ailleurs un modèle d'unification de ces deux aspects.

Dans les travaux qui suivent, nous nous situerons dans une échelle temporelle allant de quelques secondes à quelques minutes. Il s'agit donc de compositions musicales, dans lesquelles peuvent être distinguées des phrases, elles-mêmes composées de notes.

Intéressons-nous maintenant à cette note. Elle constitue la brique de base, l'unité atomique à partir laquelle tout se construit. Elle est définie par deux caractéristiques princi-

TAB. 1.1 – Tableau des échelles temporelles de la musique

Échelle temporelle	Période	Exemple musical
Infini	infini	Sinusoïdes idéales de Fournier
Supra	Jours, mois, années	Concert, album, style musical
Macro	De la minute à l'heure	Une composition
Meso	De la seconde à la minute	Phrases, section musicale
Objet sonore	De la fraction de seconde à la seconde	Note, évènement sonore distinct
Micro	Milliseconde	Particule sonore, grain
Échantillon	Microseconde(taux d'échantillonnage)	Un échantillon numérique
Sous-échantillon	Nanoseconde	Evènements sous la fréquence de Nyquist
Infinitésimal	0	Fréquence d'impulsion idéale

pales :

- Sa hauteur
- Sa durée

Il ne s'agit pas, ici, de faire un exposé exhaustif sur le solfège, mais simplement de poser les quelques éléments musicaux nécessaires à la compréhension de ce qui suit. Les experts de la musique ne manqueront d'ailleurs pas de remarquer de nombreuses imprécisions. Elles sont le fruit d'une volonté de simplification du propos.

La hauteur d'une note peut être absolue ou relative. La hauteur absolue correspond à la fréquence du son que la note représente. Elle permet de situer une note dans le spectre harmonique, indépendamment de son contexte. Par exemple, la note « la » correspond habituellement à une fréquence de 440 Hz. La hauteur relative est, elle, la hauteur de la note par rapport à une autre, soit celle qui la précède directement, soit une note particulière qu'on appelle la tonique. Plus qu'aux notes elles-mêmes, c'est aux intervalles entre elles qu'on s'intéresse. L'unité de mesure d'intervalles, du moins dans la musique occidentale, est le demi-ton. Les valeurs et les noms des différents intervalles entre les notes d'une octave sont regroupés dans le tableau TAB 1.2.

La durée d'une note peut être simplement exprimée par sa mesure physique, en secondes ou en millisecondes. En musique, on utilise plus volontiers une mesure relative des durées, basée sur la notion de temps, qui définit l'unité de mesure de durée. On exprime alors les durées en nombre de temps. Elles peuvent caractériser la durée d'une note ou celle d'un silence, lorsqu'aucune note n'est jouée. Un ensemble des durées de notes et de silences couramment employées se trouve dans le tableau TAB 1.3.

Le rapport entre ces valeurs relatives et leur durée réelle dépend du tempo, exprimé en général en nombre de temps (ou de battements) par minute. Ainsi exprimé, un tempo de 90 bpm indique ainsi qu'il y aura l'équivalent de 90 noires par minute.

Il existe d'autres caractéristiques d'une note, telles que l'intensité et le timbre :

- L'intensité correspond à la puissance avec laquelle la note est jouée. Elle se traduit par un volume sonore plus ou moins important.
- Le timbre caractérise la nature du son. Il est en général dépendant de l'instrument utilisé pour produire la note.

Les notes sont organisées en gammes. Une gamme est un ensemble de notes construit à partir d'une note fondamentale, que l'on appelle la tonique, qui définit la tonalité de la gamme, et d'un qualificatif qui définit quels intervalles servent à construire la gamme depuis

TAB. 1.2 – Table des intervalles

Nom de l'intervalle	Notes en gamme de do	Distance en demi-tons
Unisson	do	0
Seconde mineure	réb	1
Seconde majeure	ré	2
Tierce mineure	mib	3
Tierce majeure	mi	4
Quarte	fa	5
Quinte diminuée	solb	6
Quinte	sol	7
Sixte mineure	lab	8
Sixte majeure	la	9
Septième mineure	sib	10
Septième majeure	si	11
Octave	do	12

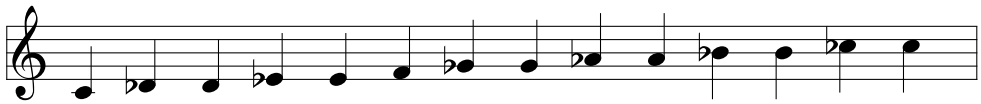


FIG. 1.1 – Gamme chromatique de do

cette tonique. La gamme qui comporte tous les intervalles, appelée la gamme chromatique, est peu utilisée en tant que telle. Mais elle sert de base à la création des autres gammes qui en sont des sous-ensembles. A titre d'exemple, la gamme chromatique de do se trouve sur la Fig 1.1) Parmi les autres gammes, on peut citer :

- la gamme majeure, une gamme de do majeur figure sur la Fig 1.2 ;
- la gamme mineure harmonique, une gamme de do mineur harmonique apparaît sur la Fig.1.3 ;
- la gamme pentatonique mineure, une gamme pentatonique de do mineur est représentée sur la Fig. 1.4.

Mais il existe bien d'autres gammes.

Certains instruments ne sont pas capables de produire la totalité des notes disponibles. L'ensemble des notes accessibles à un instrument est appelé sa tessiture. Par ailleurs,

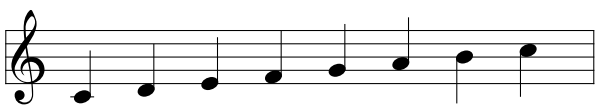


FIG. 1.2 – Gamme de do majeur

TAB. 1.3 – Table des figures rythmiques

Durée (en temps)	Nom note	Notation note	Nom silence	Notation silence
4	ronde		pause	
2	blanche		demi-pause	
1	noire		soupir	
1/2	croche		demi-soupir	
1/4	double-croche		quart de soupir	
1/8	triple croche		huitième de soupir	

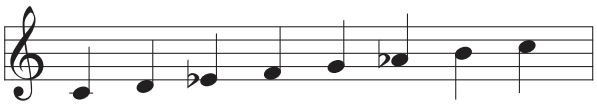


FIG. 1.3 – Gamme de do mineur harmonique

lorsque celle-ci est limitée à un sous ensemble des notes d'une octave, on parle d'instrument diatonique, en opposition aux instruments chromatiques, qui ont accès à la totalité de la gamme du même nom.

La création de phrases musicales, telles que nous la considérons dans la suite, se base sur ces gammes. L'organisation d'une série de notes suit certaines contraintes. Les considérations mélodiques sont liées aux intervalles entre des notes jouées les unes après les autres. Alors que les considérations harmoniques concernent les notes jouées simultanément, elles font intervenir la notion d'accord. Un instrument capable de jouer plusieurs notes simultanément, et donc de restituer seul des intervalles harmoniques, est qualifié de polyphonique.

On parle du contexte harmonique d'une note pour décrire l'ensemble des notes jouées en même temps qu'elle ou un peu avant, et qui permet de la considérer, non par sa hauteur absolue, mais pas les intervalles qu'elle forme avec les autres. Ainsi la note « sol » jouée avec « do » et « mi » est la quinte juste d'un « do majeur », mais produit un effet très différent si elle est jouée avec un « mi » et un « si », ou elle devient la tierce mineure d'un

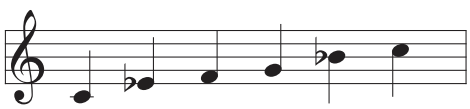


FIG. 1.4 – Gamme pentatonique de do mineur

« mi mineur ». Une suite de contexte harmonique est qualifiée de progression harmonique.

Dans la pratique, c'est du choix des règles utilisées pour sa création que va dépendre le morceau produit. Certaines approches prescrivent l'utilisation au sein d'un morceau, ou d'une partie de morceau, d'un ensemble de notes tirées d'une seule et unique gamme, à l'exclusion bien sûr de la gamme chromatique, ce qui reviendrait à ne pas restreindre l'ensemble des notes disponibles.

De sa création à sa diffusion, la musique fait intervenir différents rôles. Ces rôles sont d'ailleurs abordés par John Al Biles dans ses travaux [Biles, 2007a]. La première étape est celle de la composition. Elle consiste à concevoir une pièce musicale pouvant éventuellement être jouée plus tard. C'est le rôle de l'interprète, qui joue une composition musicale préalablement créée. Lorsque l'interprète compose de la musique en temps réel, on parle d'improvisation. Lors d'une diffusion musicale, interviennent le technicien qui est chargé de la restitution technique du son, ainsi, bien sûr, que l'audience qui va entendre la musique produite.

1.1.1 La norme MIDI

La norme MIDI, pour *Musical Instrument Digital Interface* est apparue dans les années 80. À l'origine, c'est un protocole de communication entre instruments de musiques électroniques. Dans la pratique, elle désigne et définit les interfaces de raccordement physique, le protocole de communication en lui-même et le format des fichiers permettant de stocker une suite de commandes. Elle a été complétée par la norme *General MIDI* qui définit, entre autres, un ensemble de 128 instruments nécessairement disponibles et les codes associés.

Le protocole est basé sur des messages qui transitent entre les équipements, qui peuvent, bien sûr, être des programmes informatiques. Ces équipements peuvent être regroupés en plusieurs catégories :

Les synthétiseurs sont les entités chargées de la production du son. Ils sont en général les récepteurs terminaux des messages qu'ils interprètent pour déclencher les actions associées. Un synthétiseur exécute un message MIDI au plus tôt, en général dès sa réception.

Les contrôleurs sont les entités permettant d'envoyer des messages. Reliés à un synthétiseur, ils permettent ainsi de jouer de la musique directement. Les équipements physiques permettant d'envoyer des messages MIDI sont nombreux, le plus courant étant le clavier maître. Un clavier maître est un clavier de piano numérique pouvant envoyer des messages MIDI via ses interfaces. Il traduit la pression de ses touches en messages MIDI. Un clavier maître peut incorporer directement un synthétiseur, sans que ce soit toujours le cas. Quand il n'en dispose pas, il ne produit donc aucun son.

Les séquenceurs comme leur nom l'indique, permettent d'organiser dans le temps des séquences de messages MIDI. Ils sont, en général, reliés à un synthétiseur auquel ils envoient, en fonction de contraintes temporelles, les messages MIDI contenus dans leur séquence.

Un message MIDI comporte plusieurs informations :

Le type de message indique le type d'action requis. Il existe 7 types de messages principaux, plus un type particulier dit *system exclusive* servant à faire tout ce que la

norme n'a pas prévu. Les commandes associées à ce type sont spécifiques à un équipement. Dans la pratique, chaque constructeur édite sa propre norme. Les équipements ne gèrent pas toujours tous les types de messages, laissant parfois certains messages sans effet. Les types de messages sont regroupés dans le tableau Tab 1.4.

Le canal auquel ce message est destiné. La norme MIDI spécifie 16 canaux, qui permettent donc à l'aide d'un même contrôleur de diriger 16 instruments différents. La norme *General MIDI* impose l'utilisation du dixième canal pour les percussions.

Un champ de données 128 valeurs possibles, dont la signification dépend du type de message. Dans le cas des messages *Note on* et *Note off*, il indique la hauteur de la note en nombre de demi-tons. La valeur 69 correspond au « la » (440hz).

Un second champ de données 128 valeurs, dépendant lui aussi du type de message. Dans le cas des messages *Note on* et *Note off*, ce champ définit la vélocité avec laquelle la note doit être jouée. Plus la valeur est élevée, plus la note sera forte.

Tab. 1.4 – Types de messages MIDI

Numéro	Message	Action
1000	Note Off	Termine une note
1001	Note On	Débute une note
1010	Polyphonic Key Pressure	Modifie l'intensité d'une note
1011	Control Change	Envoie de valeur de contrôle
1100	Program Change	Choix du timbre
1101	Channel Pressure	Modifie l'intensité générale
1110	Pitch Wheel Change	Transposition de la hauteur générale
1111	System Exclusive	Permet des messages spécifiques

Un évènement MIDI est un message MIDI auquel est associée la date à laquelle on souhaite que cet évènement se produise. Ce sont donc les entités que manipulent les séquenceurs.

La norme MIDI permet ainsi une approche symbolique de la musique, détachée du traitement du signal. Les hauteurs y sont exprimées non pas en fréquence, mais en demi-tons. Les durées y sont, elles, exprimées en durée relative au tempo. Elle permet, du point de vue du développeur, de s'intéresser d'avantage à des considérations musicales qu'à des considérations techniques de production du son.

Pour de plus amples informations, une étude de la norme MIDI, ses capacités, ses limites et des améliorations possibles peut être trouvée dans l'article *« Musicians make a standard : The MIDI phenomenon »* de Gareth Loy [Loy, 1985]. Le site web de la *MIDI Manufacturers Association* [The MIDI Manufacturers Association (MMA), 2010] regroupe l'intégralité des spécifications de la norme.

1.2 Dessin et ordinateur

L'approche du dessin adoptée dans la suite est une vision profane et naïve. Elle ne cherche pas à se confronter à un style ou à un courant artistique particulier. Les formes produites pouvant être figuratives, abstraites ou simplement proches du gribouillage.

Les toiles que nous considérons sont des images numériques, c'est-à-dire des tableaux rectangulaires de pixels, où chaque pixel est un point caractérisé par une couleur. La taille d'une image est ainsi définie par une largeur et une hauteur, exprimées toutes deux en nombre de pixels. Un pixel n'a pas de taille définie, elle dépend du système d'affichage sur lequel il est projeté. Ainsi un vaste écran affichant une résolution de 1024 pixels de large par 768 de haut affichera de gros pixels, quand un petit écran configuré en 1600*1200 pixels produira des pixels beaucoup plus petits. Une notion de résolution peut être affectée à une image, pour associer un nombre de pixels à une unité de longueur physique. Les résolution sont exprimées en pixels par pouce (ppp). Cette notion est particulièrement importante lorsqu'une image est vouée à être imprimée. Les imprimeurs considèrent que l'oeil ne permet plus de distinguer les différences de qualité au-delà d'une résolution de 300 ppp, aussi cette valeur fait-elle office de standard pour une impression de qualité.

Les couleurs peuvent être exprimées de plusieurs façons en informatique. La plus courante consiste à lui attribuer trois valeurs entre 0 et 255 pour chacune des couleurs rouge, vert et bleu. Ce codage est appelé RVB. Plus la valeur d'une composante est élevée plus l'intensité de la couleur correspondante est grande. Ainsi 0.0.0 est le code RVB du noir, et 255.255.255 est celle du blanc. Mais il existe d'autres formes de codage des couleurs, tel que CMJN, pour Cyan, Magenta, Jaune et Noir, plus proche de la technique de quadrichromie utilisée en imprimerie et où chacune des quatre composantes est caractérisée par une valeur entre 0 et 100. Le codage TSV pour teinte, saturation et valeur est composée de trois valeurs :

- la première, comprise entre 0 et 360, donne la teinte suivant son angle dans le cercle des couleurs ;
- la saturation est le pourcentage d'intensité de cette teinte ;
- la valeur, entre 0 et 100, caractérise la brillance de la couleur.

La thèse de Nicolas Vandenbroucke [Vandenbroucke, 2000] est une excellente source d'informations sur les différents codages de couleurs.

1.3 Art et handicap

Le rapport à l'art des personnes en situation de handicap peut sembler un sujet délicat. Il semble pourtant que les besoins artistiques de ces personnes soient au moins aussi importants que ceux du reste de la population. D'ailleurs, de nombreuses initiatives visent à faciliter l'accès à l'art, ou à promouvoir les œuvres d'artistes handicapés. Elles regroupent des acteurs institutionnels, des associations...

Du coté institutionnel, la Commission Nationale Culture-Handicap (CNCH) sert, depuis sa création en 2001, d'instance de dialogue et de consultation entre le ministère de la culture et de la communication et les associations représentatives des personnes en situation de handicap [Ministère de la culture et de la communication, 2010]. Le 11 février 2009, le

Réseau National Musique et Handicap [Réseau National Musique et Handicap, 2010], sous le patronage, lui aussi, du ministère de la culture et de la communication lance la charte musique et handicap. Les signataires s'engagent à favoriser la pratique de la musique aux personnes handicapées. En 2010, elle compte 248 signataires. Ces initiatives sont la marque d'une volonté nationale de dynamiser ce secteur.

Les acteurs du monde associatif sont, eux aussi, nombreux et dynamiques. On peut, par exemple, citer l'association Art Prime, qui vise à sensibiliser le public sur le thème de l'art et du handicap, notamment en favorisant la diffusion des œuvres d'artistes handicapés [Art Prime, 2010]. Le festival Art et Handicap [Fédération APAJH de Haute-Marne, 2009] assure une diffusion de productions artistiques réalisées par des personnes artistes handicapées, et ce depuis la première édition en 2005. Ainsi, le nombre d'évènements et d'initiatives dans le domaine de l'art et du handicap, ainsi que la vitalité des acteurs attestent de l'importance de cet enjeu.

Chapitre 2

Modèles et méthodes de résolution génériques

Dans ce chapitre, nous proposons un panorama de modèles et méthodes de résolution génériques qui sert de définition aux termes et concepts utilisés et développés dans le cadre des applications abordées par la suite. Ainsi, seules les méthodes adaptées ensuite pour de la production d'art génératif sont abordées. Elles sont organisées par ordre chronologique d'apparition.

2.1 Chaînes de Markov

Une chaîne de Markov est un processus stochastique qui modélise un système à états discrets évoluant en temps discret. Les chaînes de Markov tiennent leur nom du mathématicien russe Andreï Andreïevitch Markov qui les a décrites le premier, dans un article de 1913, [Markov, 1913]. Elles sont notamment utilisées par Claude E. Shannon [Shannon, 1948] pour fonder sa théorie de l'information.

Une chaîne de Markov est définie comme l'ensemble des états que peut prendre un système. À chaque état sont associées les probabilités que le système évolue vers un autre état. Ces probabilités sont conformes à la propriété de Markov : elles dépendent uniquement de l'état actuel.

Le schéma Fig 2.1 illustre une chaîne de Markov à 3 états : A, B et C.

On peut noter que $\sum_{i\in\{A,B,C\}} P_{x-i} = 1 \quad \forall x \in \{A, B, C\}$. Le schéma représente toutes les transitions possibles. Dans la pratique, les transitions dont la probabilité est nulle ne sont pas représentées.

Les chaînes de Markov permettent d'établir des modèles de phénomènes réels où il n'y a pas de dépendance temporelle entre l'état actuel du système et ses états antérieurs. Elles sont donc utiles pour simuler ou faire de la prédiction. Le lecteur intéressé peut se reporter, par exemple, au livre de John G. Kemeny et J. Laurie Snell [Kemeny and Snell, 1960] ou aux travaux de Sebastien Aupetit [Aupetit, 2005] pour aller plus loin sur les chaînes de Markov.

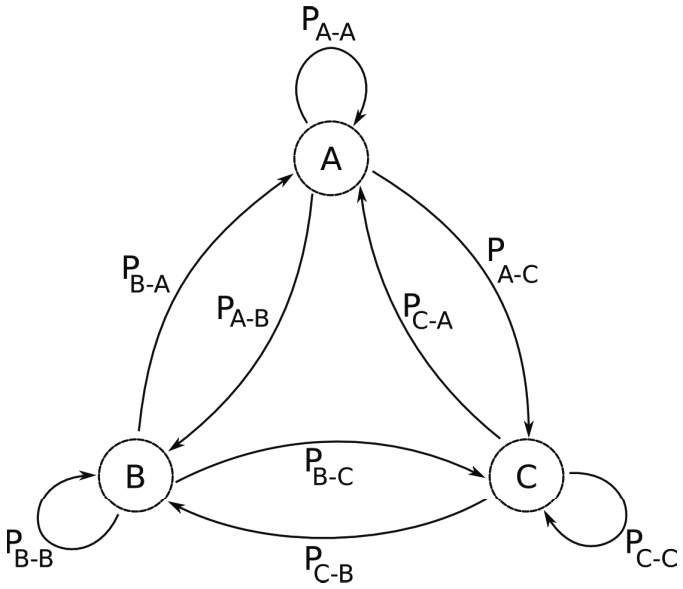


FIG. 2.1 – Schéma de principe d'une chaîne de Markov

2.2 Grammaires génératives

Les grammaires génératives sont issues des travaux de Noam Chomsky [Chomsky, 1957]. Commençons par une rapide définition des concepts de base de ces grammaires formelles :

Le symbole est l'élément unitaire de base d'une grammaire ;

Un mot est une suite de symboles ;

Un alphabet est l'ensemble des symboles utilisés pour former les mots du langage ;

Un langage est l'ensemble des mots valides formés à partir de l'alphabet.

Une grammaire formelle se définit par quatre ensembles :

- un ensemble fini et non vide de symboles non-terminaux (notés en lettres majuscules), non-terminaux signifiant qu'ils peuvent se voir appliquer une règle de production ;
- un ensemble fini et non vide de symboles terminaux (notés en lettres minuscules) ;
- un ensemble fini et non vide de règles de production qui associent à une suite de symboles contenant au moins un symbole non-terminal, une suite de symboles (terminaux ou non) qui peuvent s'y substituer.
  Par exemple : $A \rightarrow Ba$.
- un élément de l'ensemble des symboles non-terminaux, appelé axiome (noté S), qui constitue le point de départ d'application des règles de production.

L'application d'une règle de production consiste à remplacer, dans une séquence de symboles, une occurrence du symbole de gauche de la règle par les symboles de droite. Le langage défini par une grammaire est l'ensemble des mots formés uniquement de symboles terminaux, que l'on peut obtenir par applications successives des règles de production à partir de l'axiome.

Les grammaires sont notamment utilisées pour l'analyse et la conception de langages de programmation ainsi que dans le traitement automatique des langues.

Les L-system [Prusinkiewicz and Lindenmayer, 1990] ou systèmes de Lindenmayer, du nom du biologiste Aristid Lindenmayer qui les a conçus, sont des grammaires formelles. Ces systèmes permettent, entre autres, de modéliser la croissance de plantes. Il s'agit de grammaires particulières dans lesquelles il n'y a pas de symbole terminal. Ils sont déterministes quand chaque symbole n'est le membre de gauche que d'une unique règle de production. Un L-System progresse donc par étapes, depuis son axiome, en appliquant à chaque étape la règle, ou l'une des règles, associée à chaque symbole. Ce processus récursif est potentiellement infini, créant un mot de plus en plus long à chaque étape.

2.3 Réseaux de neurones artificiels

Les réseaux de neurones artificiels s'inspirent du fonctionnement des neurones biologiques. Ils se basent sur un modèle simplifié de neurone : le neurone formel de McCulloch et Pitts [Lettvin et al., 1959].

Dans ce modèle, le neurone possède plusieurs entrées (X_i avec $i = [1..n]$) et une sortie (O), toutes binaires, c'est-à-dire véhiculant un signal valant 0 ou 1. La sortie est la somme pondérée des entrées (W_i est le poids associé à l'entrée i) à laquelle on applique une fonction seuil (ϕ). Plusieurs fonctions seuils peuvent être utilisées, telle que la fonction sigmoïde ou la tangente hyperbolique.

$$O = \phi\left(\sum_{i=0}^{n} X_i W_i\right) \quad \text{tel que} \quad O \in \{0, 1\} \tag{2.1}$$

Une représentation schématique est donnée Fig. 2.2

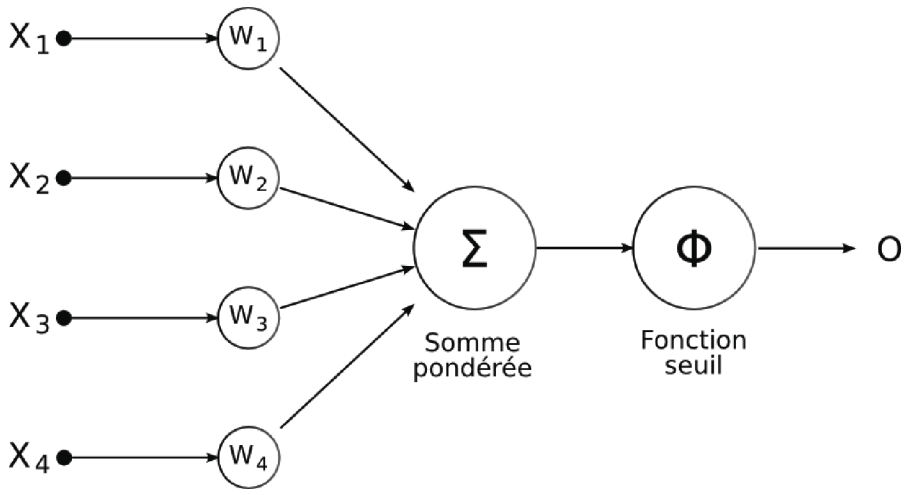


Fig. 2.2 – Schéma d'un neurone formel

Ces neurones sont reliés les uns aux autres pour former un réseau. Ce réseau peut être constitué de couches successives, les neurones de chaque couche n'étant reliés qu'aux

neurones de la couche supérieure (Cf. Fig. 2.3). Ce type de réseau est appelé perceptron [Rosenblatt, 1958, Minsky and Papert, 1987]. Les premiers perceptrons, mais aussi les plus simples ne sont constitués que d'une couche de neurones, on parle de perceptron multi-couche dans les autres cas.

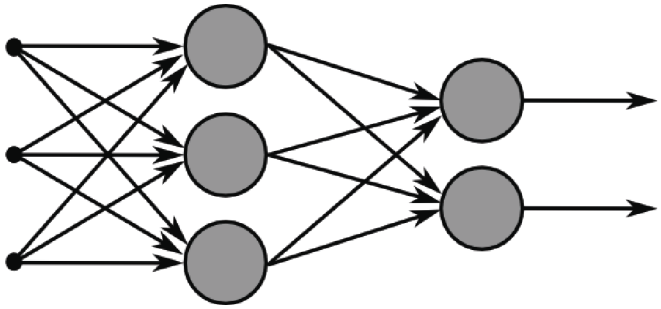


Fig. 2.3 – Schéma d'un réseau de neurones artificiels

L'introduction de boucles dans la topologie du réseau change profondément la complexité, mais aussi les capacités de modélisation des réseaux de neurones, on parle alors de réseaux de neurones récurrents.

Un réseau de neurones artificiels constitue donc un système qui, pour un vecteur d'entrée correspondant aux entrées des neurones de la première couche, produit un vecteur de sortie composé des sorties des neurones de la dernière couche. Pour que cette sortie soit significative, le réseau nouvellement créé doit passer par une phase d'apprentissage, dont l'objectif est de déterminer les poids associés aux entrées de chacun des neurones. Cette phase nécessite un ensemble de solutions déjà connues, qu'on appelle base d'apprentissage, pour lesquelles on va appliquer une correction des poids du réseau lorsque la sortie produite ne correspond pas à celle attendue. Elle permet au réseau de mémoriser les exemples qui lui ont été présentés et de généraliser à d'autres entrées encore inconnues.

Les réseaux de neurones artificiels sont particulièrement adaptés pour résoudre les problèmes de classification, comme le souligne Antonio Ciampi et Yves Lechevallier [Ciampi and Lechevallier, 1995]. Néanmoins, il convient de faire particulièrement attention à la phase d'apprentissage, potentiellement longue. Le sur-apprentissage en est un biais que l'on cherche à éviter, lorsque le réseau n'est plus capable de généraliser et qu'il ne peut que reconnaître les exemples de la base d'apprentissage.

2.4 Automates cellulaires

Les automates cellulaires permettent de modéliser simplement des systèmes complexes [Von Neumann, 1966]. Un automate cellulaire est un ensemble de cellules regroupées en une grille. Chaque cellule peut prendre un nombre fini d'états. Un automate cellulaire évolue pas à pas. À l'étape $n+1$ l'état de chaque cellule dépend de l'état, au temps n, de ses cellules voisines. L'état de toutes les cellules de l'automate est recalculé à chaque pas de temps, la même règle de transformation étant appliquée à chaque cellule.

2.5 Algorithmes génétiques

Globalement inspirés du principe de sélection naturelle de Charles Darwin, les algorithmes génétiques font partie de la famille des algorithmes évolutionnaires. Les premiers travaux qui en posent les bases sont ceux de John Holland [Holland, 1975]. Souvent utilisés pour résoudre des problèmes d'optimisation, ils font évoluer un ensemble de solutions au problème donné vers un optimum, parfois seulement local. La proximité de la solution obtenue par rapport à l'optimum théorique est souvent inconnue. Les algorithmes génétiques tirent la terminologie des notions qu'ils utilisent du domaine de la biologie, ainsi :

Un individu identifie une solution particulière de l'ensemble des solutions au problème donné.

Une population regroupe l'ensemble des individus d'une génération.

Une génération est une itération de l'algorithme, chaque génération produisant une nouvelle population d'individus.

Le codage d'un individu, autrement dit, la façon dont une solution au problème donné est représentée, est une phase délicate de la conception d'un algorithme génétique, à ne pas négliger. Elle associe une signification à une chaîne de bits, appelée chromosome. Elle consiste, donc, à choisir quels éléments représentent un individu, éventuellement à les regrouper au sein de chromosomes distincts. Dans de nombreux cas, on procède même à une distinction entre le génotype (l'exact contenu des chromosomes d'un individu), et le phénotype (la façon dont ils s'expriment effectivement, et peuvent être observés sur l'individu).

Les algorithmes génétiques reposent sur un ensemble d'opérations qui vont s'appliquer aux individus de la population :

L'évaluation consiste à noter, ou au moins, à classer les individus d'une population.

La sélection vise à tirer de la population les meilleurs individus. Basée sur l'évaluation, elle définit les individus qui seront les parents de la génération suivante.

Le croisement permet d'obtenir un nouvel individu fils à partir de deux individus parents, par mélange de leurs chromosomes.

La mutation est une modification aléatoire et rare (en général, inférieur à 0,1%) d'un élément des chromosomes d'un individu.

Un algorithme génétique procède, à partir d'une population de départ, de préférence aléatoire, à une série de générations. Chaque itération est constituée ainsi :

1. Une phase d'évaluation de la population de génération n.

   Certains problèmes peuvent rendre la phase d'évaluation délicate, notamment lorsqu'elle ne peut pas être objective, et donc difficilement automatisable. Une adaptation pour pallier ce problème d'évaluation, proposée notamment par Hideyuki Takagi [Takagi, 1998], est de s'appuyer sur une évaluation humaine. On va ainsi demander à un expert de procéder lui-même à l'évaluation. On parle alors d'évolution artificielle interactive.

2. Une phase de sélection des parents.

   Il existe plusieurs politiques de sélection, plus ou moins aléatoires. Elles peuvent consister à prendre les n meilleurs individus, ou à offrir, à chaque individu, une

chance proportionnelle à son évaluation, d'être choisi. La sélection peut aussi s'effectuer directement sur des couples d'individus, en fonction de la combinaison de leurs évaluations. Dans certains cas, elle est même totalement aléatoire.

3. Une phase de croisement des parents.

   L'opérateur de croisement a pour objectif d'obtenir un nouvel individu (parfois plusieurs) à partir de deux individus parents en mêlant une partie des chromosomes de chacun des parents. Il est donc très dépendant du codage utilisé pour les individus.

4. Une phase de mutation.

   Les mutations consistent à modifier une partie, souvent un bit dans le génotype d'un individu. Elles permettent, conjointement avec l'opérateur de croisement, d'éviter une convergence prématurée vers un optimum local.

5. Une phase de constitution de la population de génération $n + 1$.

Il existe plusieurs politiques d'insertion des nouveaux individus et de suppression des anciens dans la population de la génération suivante. En général, on insère les nouveaux individus, on évalue toute la population et on enlève les moins bons pour obtenir une population de la taille choisie, c'est l'approche élitiste. Le remplacement générationnel consiste lui, à chaque génération, à remplacer la totalité de la population. On génère donc autant de fils que la taille de la population voulue. Le remplacement continu insère les nouveaux individus en remplacement des individus de plus faible évaluation.

Pour plus de détails sur les algorithmes génétiques, vous pouvez consulter, par exemple : «The design of Innovation : Lessons from and for Competent Genetic Algorithms» de David E. Goldberg [Goldberg, 2002], l'article « Contrôle d'un algorithme génétique » de Michèle Sebag et Marc Schoenhauer [Sebag and Schoenauer, 1996] ou celui d'Evelyne Lutton, « Darwinisme artificiel : une vue d'ensemble » [Lutton, 2005].

2.6 Systèmes multi-agents

Les systèmes multi-agents sont basés sur l'interaction entre plusieurs entités distinctes. Ils peuvent prendre des formes variées suivant l'autonomie et la complexité des agents et les règles qui s'appliquent à eux dans l'environnement où ils évoluent. Ces méthodes partagent des concepts avec les méthodes évolutionnaires. Les travaux de Michael Wooldridge [Wooldridge, 1992] ou le livre édité par Gerhard Weiss [Weiss, 1999] permettent d'approfondir la question des systèmes multi-agents. Selon les auteurs, les algorithmes de colonie de fourmis artificielles peuvent faire partie de cette catégorie des systèmes multi-agents. Nous choisissons de les en distinguer du fait de l'intérêt particulier que nous avons pour ces algorithmes de fourmis. Ils sont d'ailleurs détaillés dans la partie suivante.

2.7 Algorithmes de colonie de fourmis artificielles

Les algorithmes de colonie de fourmis artificielles naissent du constat que, bien que constituée d'individus au comportement assez simple, une colonie de fourmis est capable de résoudre des problèmes complexes. On parle d'ailleurs d'intelligence collective ou de

comportement émergent. En effet, bien que ne disposant pas d'un contrôle centralisé, la colonie dans son ensemble est capable de plus que la simple somme de ce que les individus qui la composent sont capables de faire. Ces constatations sont d'ailleurs dans l'ouvrage *« Swarm Intelligence : From Natural to Artificial Systems »* de Bonabeau et al. [Bonabeau et al., 1999].

Cette intelligence collective est basée sur les multiples interactions entre les membres du groupe. Ces interactions pouvant être directes, telles que des communications d'individus à individus, ou indirectes, telles que la communication via des dépôts de phéromones sur l'environnement. Cette communication indirecte est appelée stigmergie.

Des modèles mathématiques probabilistes permettent de simuler le comportement des fourmis. Ce sont ces modèles qui servent de base aux méthodes de résolution décrites ensuite. Ils utilisent des notions de rétroactions positives, notamment via le renforcement des pistes de phéromones, ou négatives comme l'évaporation de ces mêmes phéromones. Plusieurs modèles de comportements sont utilisés.

2.7.1 Le déplacement des fourmis

Le déplacement collectif des fourmis, notamment pour la recherche de nourriture est l'un des plus utilisé. Dans les algorithmes imitant ce comportement, on utilise un ensemble de fourmis qui se déplacent sur un environnement. Pour la suite, nous supposerons que cet environnement est modélisé sous la forme d'un graphe. À chaque instant, chaque fourmi se trouve donc sur un sommet de ce graphe. La communication au sein de la colonie s'effectue par des dépôts de phéromones sur ce graphe. En effet, lors de leurs déplacements, les fourmis déposent une certaine quantité de phéromones sur les sommets sur lesquels elles passent.

Pour se déplacer, une fourmi est amenée à faire des choix entre les différentes directions qui s'offrent à elle. Elle s'intéresse alors aux sommets accessibles en empruntant un seul arc, à partir du sommet où elle se trouve. En effet, on considère qu'une fourmi ne s'intéresse qu'aux informations locales pour faire ses choix, et qu'elle ne prend pas de décision à long terme. Elle procède ensuite à un tirage au sort entre ces différentes directions. Ce tirage n'est pas équiprobable et les probabilités de choisir une direction dépendent des quantités de phéromones trouvées sur chacun des sommets voisins. Plus la quantité est importante, plus la probabilité de choisir ce chemin est forte. Ainsi, une fourmi à plutôt tendance à suivre les directions ayant une forte concentration en phéromones que les autres. Ce choix peut être défini, dans le cas de deux chemins A et B, par la probabilité suivante [Bonabeau et al., 1999]

$$P_A = \frac{(k + A_i)^n}{(k + A_i)^n + (k + B_i)^n} = 1 - P_B \tag{2.2}$$

avec :

P_A : probabilité de choisir le chemin A au moment $(i + 1)$;

k : degré d'attraction d'un chemin non marqué ;

A_i **et** B_i : quantité de phéromones sur le chemin A et B au moment i ;

n : paramètre de l'algorithme, permettant d'ajuster l'influence des phéromones.

La fourmi dépose des phéromones sur son passage, ainsi : $A_{i+1} = A_i + 1$ si la fourmi i a choisi le chemin A, $A_{i+1} = A_i$ sinon.

Utilisé dans un environnement avec une source et une cible, telles que le nid de la colonie et une source de nourriture, par exemple, le comportement général de cet algorithme est d'avoir, dans un premier temps, des fourmis qui partent dans tous les sens. Puis, progressivement, les fourmis utilisent toutes, ou presque, le même chemin. Ce chemin a une forte chance d'être le plus court chemin entre la source et la cible.

Ce comportement permet, donc, de résoudre des problèmes d'optimisation, notamment ceux consistant à rechercher un plus court chemin dont le problème du voyageur de commerce est l'exemple le plus connu. Le travail de Marco Dorigo sur *Ant-system* (AS) [Dorigo et al., 1996] l'illustre d'ailleurs. Il constitue le premier algorithme de la famille des algorithmes d'optimisation basé sur des colonies de fourmis, appellés ACO pour *Ant Colony Optimization* [Dorigo and Di Caro, 1999, Dorigo and Stützle, 2004].

Cette famille n'a depuis cessé de s'agrandir avec, par exemple, *ANTS* [Maniezzo, 1999] ou *MAX-MIN Ant System* [Stützle and Hoos, 2000], qui proposent des améliorations d'AS, ou encore API [Monmarché et al., 1998] qui permet de résoudre une vaste catégorie de problèmes d'optimisation, comme par exemple, l'apprentissage de modèles de Markov cachés [Aupetit et al., 2009].

2.7.2 Le tri du couvain

Le tri du couvain est un autre comportement qui sert de modèle à d'intéressantes méthodes. Les fourmis se déplacent dans un environnement où se trouvent des objets. Quand elle passe à coté d'un objet, une fourmi a une probabilité P_r de le ramasser. Une fois chargée, cette fourmi a une probabilité P_d de déposer sa charge. Eric Bonabeau, Marco Dorigo et Guy Theraulaz [Bonabeau et al., 1999] proposent les fonctions suivantes pour calculer ces probabilités

$$P_r = \left(\frac{k_1}{k_1 + f}\right)^2 \tag{2.3}$$

Avec :

f : le taux d'objet perçu dans le voisinage ;

k_1 : un seuil constant.

Ainsi, quand $f \ll k_1$, la probabilité est élevée et quand $f \gg k_1$, la probabilité est faible. Les fourmis ont plutôt tendance à ramasser les objets isolés.

La probabilité de dépôt est calculée ainsi :

$$P_d = \left(\frac{f}{k_2 + f}\right)^2 \tag{2.4}$$

Avec :

k_2 : un autre seuil constant.

Le taux d'objet perçu, f peut être simplement le nombre d'objets que l'agent a croisé récemment rapporté au nombre maximum d'objets qu'il aurait pu croiser. Par exemple, dans le cas d'une grille où se déplacent des fourmis à la vitesse d'une case par pas de temps, et où chaque case de grille ne peut contenir plus d'un objet. Pendant une unité de temps, une fourmi ne peut pas croiser plus d'un objet, donc $f = \frac{N}{T}$ quand l'agent a croisé N objets durant T pas de temps. Ce comportement permet à la colonie de regrouper les objets dispersés dans leur environnement. Il sert de base à des méthodes d'analyse de données, de partitionnement de graphe ou encore de classification.

On peut citer l'algorithme d'Eric D. Lumer et Baldo Faieta [Lumer and Faieta, 1994] où les agents se déplacent au hasard, ramassent ou déposent des objets pour regrouper ces objets en classes, AntClass [Monmarché, 2000], AntTree [Azzag et al., 2003] ou encore l'adaptation de Steven Schockaert [Schockaert et al., 2004].

2.7.3 Partage de tâches chez les fourmis

Les mécanismes de division du travail au sein d'une colonie s'avèrent, eux aussi, être d'intéressantes sources d'inspiration pour la conception d'algorithmes. Le modèle de Eric Bonabeau, Marco Dorigo et Guy Theraulaz [Bonabeau et al., 1999] décrit les mécanismes de répartition de la charge de travail pour une colonie de fourmis.

On considère qu'il existe un ensemble de tâches à réaliser, chaque tâche étant caractérisée par une valeur : l'intensité du stimulus S_i, pour la tâche i. Cette valeur mesure l'importance pour la colonie de voir cette tâche effectuée. Plus la valeur est grande, plus la tâche est urgente ou importante. Chaque fourmi dispose d'une série de seuils de réponse, un par tâche, qui définissent la préférence qu'elle a pour chaque tâche. On note θ_i, le seuil de réponse associé à la tâche i. La probabilité qu'une fourmi exécute une tâche dépend de ces deux paramètres. On note T_i la probabilité qu'une fourmi effectue la tâche i. Plusieurs fonctions peuvent être utilisées pour la calculer mais l'idée générale est que si $S_i \ll \theta_i$ alors cette probabilité est faible, elle est forte quand $S_i \gg \theta_i$.

Une famille de fonctions répondant à ce critère est celle-ci :

$$T_i = \frac{S_i^n}{S_i^n + \theta_i^n} \quad \text{avec} \quad n > 1 \tag{2.5}$$

Le paramètre n permet ainsi de définir différentes fonctions, au comportement légèrement différent.

Cette autre fonction est aussi proposée par les auteurs :

$$T_i = 1 - e^{S_i/\theta_i} \tag{2.6}$$

Ce modèle s'est ensuite enrichi, en ajoutant la notion de spécialisation aux fourmis. Ainsi, une fourmi qui effectue une tâche a plutôt tendance à choisir cette même tâche qu'une autre. Deux nouveaux paramètres apparaissent :

ϵ : un coefficient d'oubli

ϕ : un coefficient d'apprentissage

Chaque fourmi, à l'issue de l'exécution de la tâche i pendant le temps Δt, voit son seuil de réponse correspondant θ_i augmenter :

$$\theta_i \leftarrow \theta_i + \phi\Delta t \tag{2.7}$$

Tandis que les seuils correspondant aux autres tâches vont diminuer :

$$\theta_i \leftarrow \theta_i - \epsilon\Delta t \tag{2.8}$$

Cette dynamique permet de concevoir des systèmes de répartition de charge en ligne réactifs et robustes. Ils ont d'ailleurs été utilisés avec succès pour adapter automatiquement le niveau d'un jeu au joueur qui l'utilise [Sepchat et al., 2008a, Sepchat et al., 2008c]. Ces articles décrivent un système basé sur ce partage de tâches chez les fourmis, servant de moteur d'intelligence artificielle à un jeu vidéo accessible. La colonie décide des actions de l'adversaire virtuel en fonction de stimuli induits par les actions du joueur humain. Il tend à réduire l'impact du joueur sur le jeu, ce qui est particulièrement intéressant dans ce cas là, comme nous le verrons dans la partie 4.2.

Il existe d'autres comportements qui sont ou pourraient être utilisés, comme la construction collective sur patron basée sur l'observation des colonies de termites et d'abeilles, ou les mécanismes de transport collectif, utilisés notamment pour des robots. Pour de plus amples détails sur les algorithmes de fourmis, voir [Monmarché et al., 2009]

Chapitre 3

De l'art génératif

Ce chapitre dresse un état de l'art des méthodes génératrices de créations artistiques, vues sous l'angle de la musique et de l'art visuel numérique. Il s'appuie sur les modèles ou méthodes de résolutions génériques détaillés au chapitre précédent.

3.1 Quelques considérations générales sur l'art génératif

La pratique d'une forme d'expression artistique amène à s'interroger sur les techniques mises en œuvre pour cette pratique. Bien qu'elles sont parfois intuitives ou indéfinies, ces techniques peuvent quelquefois être formalisées, voir automatisées. On s'approche alors de procédés algorithmiques visant à la production d'œuvres artistiques. La musique sérielle est un exemple de ces techniques formelles, qu'on peut apprécier dans les œuvres de Pierre Boulez [Leleu and Decroupet, 2006] ou celles de Karlheinz Stockhausen [Cott, 1979]. Dans le domaine graphique, on peut aussi citer l'art numérique d'Yvaral [Lambert, 1977].

Les articles regroupés dans *« Dimensions of Creativity »* par Margaret A. Boden [Boden, 1996] visent à la définition du processus de création artistique lui-même. Les auteurs se questionnent sur la source de la créativité, sa nature, et s'il est possible de la définir, voir de la mesurer. La notion de création combinatoire, où le processus créatif procède par combinaisons nouvelles d'éléments pré-existants, est complétée par la constatation qu'il existe des formes de création totalement nouvelles. Par ailleurs, la notion de nouveauté dépend de l'échelle considérée, et ce qu'un être humain fait pour la première fois, à l'échelle personnelle, peut avoir déjà été fait, à l'échelle de l'humanité. En lien avec les méthodes dites d'intelligence artificielle, dans son article *« Creativity and Artificial Evolution »*, Margaret A. Boden [Boden, 2007] pousse plus loin son exploration et sa formalisation du processus de création artistique. Considérant la création artistique comme une forme d'exploration d'un vaste espace de possibles, assimilable à l'espace des solutions des techniques d'optimisation, elle propose trois formes de création artistique :

Combinatoire : la création artistique est le fruit de la combinaison d'œuvres existantes.

Exploratoire : elle explore de nouvelles voies créatives.

Transformationnelle : l'obtention d'une nouvelle création se fait par la modification d'une préexistante.

La suite de ce chapitre détaille des méthodes mettant en œuvre ces principes, avec pour objectif la génération de créations musicales ou graphiques.

3.2 Étude de méthodes de génération de musique

La musique n'est pas un champ d'application récent, en ce qui concerne la formalisation de techniques créatives. Dès le XVIII^{e} siècle, Mozart ou son ami clarinettiste Stadler ont proposé, sous la forme de jeux, des outils de création de valses ou de menuets [Lorrain, 2003]. Il s'agit en fait de tableaux de mesures, déjà composées, pour chaque partie du morceau, plusieurs jets de dés permettant d'en créer la succession.

Le jeu de Mozart, par exemple, est décrit dans son livret « Instruction pour composer autant de Walzes ou Schleiser que l'on veut par le moyen de deux Dez sans savoir la musique ou la composition » [Mozart, 1793]. Ce livret contient deux tableaux qui sont reproduits sur la figure Tab. 3.1, 176 mesures déjà composées, qui ne sont pas reproduites ici, mais qui peuvent être trouvées dans le document original, ainsi que les instructions pour utiliser ces éléments.

Le jeu consiste à jeter une paire de dés à six faces et à se reporter, dans un des tableaux, à la ligne qui correspond à la somme des dés. On procède ainsi à 8 jets successifs, un pour chaque colonne (A,B,C,D,E,F,G,H) du premier tableau, et on note, pour chaque colonne, le nombre se trouvant dans la case à l'intersection du résultat des dés et de la colonne en question. On obtient donc une série de 8 nombres, compris entre 1 et 176. Par exemple, un résultat de 6 au premier jet, suivi d'un 10 au second, donnera une suite A = 148, B = 142... On récupère alors, dans les mesures préconstruites, celles correspondant à chacun des nombres obtenus et on les accole dans l'ordre alphabétique des lettres. On dispose, maintenant, d'une série de huit mesures qui constitue la première partie de la valse. Il ne reste plus qu'à procéder de même avec le second tableau, pour obtenir la seconde partie de la valse. Un exemple de résultat produit se trouve à la figure Fig. 3.1.

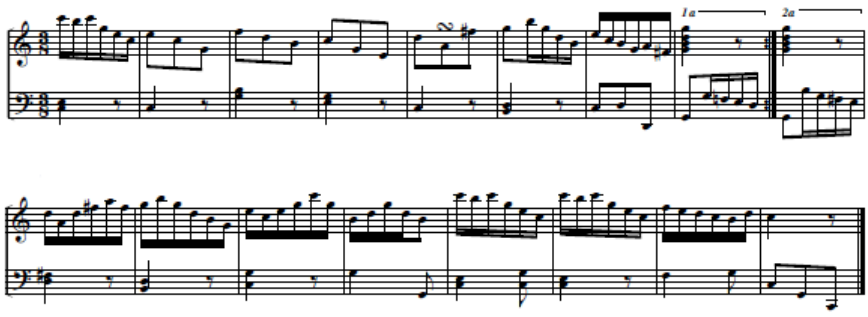


Fig. 3.1 – Valse produite avec le jeu musical de Mozart

Ce genre de jeux peut être considéré comme les débuts de la création musicale automatique. Plus récemment, depuis l'apparition d'équipements permettant aux ordinateurs de produire des sons, de nombreuses méthodes algorithmiques ont été proposées

TAB. 3.1 – Jeu de construction de Valses de Mozart

Première partie

	A	B	C	D	E	F	G	H
2	96	22	141	41	105	122	11	30
3	32	6	128	63	146	46	134	81
4	69	95	158	13	153	55	110	24
5	40	17	113	85	161	2	159	100
6	148	74	163	45	80	97	36	107
7	104	157	27	167	154	68	118	91
8	152	60	171	53	99	133	21	127
9	119	84	114	50	140	86	169	94
10	98	142	42	156	75	129	62	123
11	3	87	165	61	135	47	147	33
12	54	130	10	103	28	37	106	5

Seconde partie

	A	B	C	D	E	F	G	H
2	70	121	26	9	112	49	109	14
3	117	39	126	56	174	18	116	83
4	66	139	15	132	73	58	145	79
5	90	176	7	34	67	160	52	170
6	25	143	64	125	76	136	1	93
7	138	71	150	29	101	162	23	151
8	16	155	57	175	43	168	89	172
9	120	88	48	166	51	115	72	111
10	65	77	19	82	137	38	149	8
11	102	4	31	164	144	59	173	78
12	35	20	108	92	12	124	44	131

pour produire de la musique automatiquement. Dans la suite, nous décrivons plusieurs de ces méthodes en dégageant les objectifs et les contraintes associés à chacune d'elles, en nous appuyant sur les expériences que nous avons menées, les conclusions des auteurs eux-mêmes, ainsi que celles d'études sur le sujet. Parmi ces études Georges Papadopoulos [Papadopoulos and Wiggins, 1999] dégage les faiblesses de chacune des méthodes qu'il décrit, pour conclure sur la proposition d'hybrider ces méthodes pour tirer parti des avantages de chacune, et de permettre à ces systèmes de composition musicale d'évoluer pour pouvoir augmenter leur base de connaissances. Gerhard Nierhaus [Nierhaus, 2009], quant à lui, retrace, dans son livre, l'histoire de la composition algorithmique en remontant aux fondements de l'algorithmique, puis l'utilisation successive de différentes méthodes. Il note notamment les difficultés pour ces méthodes de s'affranchir de la simple imitation d'un style donné, celles liées à l'encodage et à la modélisation de la musique ainsi que celles de l'évaluation des productions de ces algorithmes.

3.2.1 Modèles de Markov

Dès les années 60, les modèles de Markov ont commencé à être utilisés comme systèmes de génération musicale. L'Illiac Suite d'Hiller et Isaacson [Hiller and Isaacson, 1959], par exemple, résulte de l'adaptation de règles harmoniques en modèles mathématiques. Ces modèles, d'abord totalement aléatoires, ont évolué en chaînes de Markov, plus à même de produire un résultat organisé.

Le compositeur Iannis Xenakis, dans son ouvrage Musiques formelles [Xenakis, 1981], propose une analyse formelle du processus de composition musicale et de méthodes d'automatisation de ce processus. Il y définit l'être sonore comme un vecteur à 4 composantes :

- le timbre
- la hauteur
- l'intensité
- la durée

Il envisage dans un premier temps de faire suivre à ces êtres sonores une loi de probabilité (ex. loi de Poisson), pour produire ce qu'il appelle de la « musique stochastique libre ». Il s'agit, selon lui, de la forme la plus générale de composition, qu'il appelle libre, car indépendante de toute contrainte. Il raffine ensuite son approche en partant du postulat que faire de la musique, c'est suivre des règles, celles que l'on a apprises ou parfois de nouvelles que l'on se fixe soi-même. Cet ensemble de règles est en quelque sorte un langage commun qui permet de se faire comprendre de ceux à qui la création est destinée. Plus généralement même, la « pensée créatrice » utilise un ensemble de contraintes qu'on peut parfois formaliser.

Partant de ces arguments, il propose l'utilisation de chaînes de Markov pour sa « musique stochastique markovienne ». La musique produite résulte d'une série de processus markoviens hiérarchiques déterminant :

- les instruments disponibles ;
- les dates d'occurrences des sons.
  Pour chacun de ces sons :
  - l'instrument à utiliser ;
  - sa hauteur en fonction de l'instrument ;
  - sa durée du son ;
  - sa dynamique d'émission.

  Ainsi, le passage du processus d'état en état permet la création d'une œuvre musicale.

Les chaînes de Markov permettent donc de générer de la musique, éventuellement en temps réel et sans besoin d'une importante puissance de calcul, comme le souligne Georges Papadopoulos [Papadopoulos and Wiggins, 1999]. Les chaînes de Markov, par contre, rendent difficile la gestion du contexte harmonique, en tout cas sans avoir à déployer un nombre d'états trop important, qui viendrait ralentir le processus de génération, voire le rendre tout simplement impossible car bien trop long. Un système à base de chaînes de Markov peut donc produire une mélodie, mais n'est pas indiqué pour générer une pièce polyphonique. Le passage à des modèles de Markov cachés permet de surpasser cette limitation. Dans ses travaux, Darrell Conklin [Conklin, 2003] insiste sur le fait que les systèmes génératifs à base de chaînes de Markov sont surtout utilisés pour imiter un style donné, ce qui dénote de leur manque de flexibilité. Par ailleurs, du fait de la propriété de Markov, il

n'existe que des liens locaux dans la musique produite. Chaque note dépend uniquement de la précédente, aucune structure de plus haut niveau n'apparaît dans les phrases musicales produites par ce genre de système. La nature des pièces générées étant fortement dépendante de la table des probabilités, on peut assez facilement prévoir le type de musique qui va être généré.

3.2.2 Grammaires génératives

Les grammaires formelles servent, elles aussi, de générateur de musique, comme l'illustrent les travaux de Jon McCormack [McCormack, 1996]. Il propose d'utiliser les grammaires génératives, plus précisément, les L-System, comme une alternative palliant les problèmes rencontrés avec les chaînes de Markov. Dans ses travaux, les symboles de la grammaire sont la hauteur, la durée et le timbre des notes auxquels seront appliquées des règles de ré-écriture. L'utilisation de la norme MIDI permet alors de jouer directement les morceaux produits. Jon McCormack poursuit par une modification des symboles de la grammaire, pour utiliser des intervalles à la place des hauteurs. Il obtient ainsi, à partir d'une grammaire simple, des compositions plus complexes. Un passage de grammaires simples à un système de grammaires hiérarchiques permet, selon lui, l'apparition de structures de plus haut niveau.

Alfonso Ortega de la Puente et al. [Ortega de la Puente et al., 2002] proposent un système similaire, couplé à un algorithme génétique chargé de faire évoluer les règles de la grammaire. L'objectif est d'obtenir un ensemble de règles qui produisent une musique ressemblant à une composition humaine.

Les travaux de Shlomo Dubnov [Dubnov et al., 2003] et ses collègues sur un système visant à générer des morceaux dans un style musical préalablement appris mettent en comparaison une méthode à base de grammaire générative avec une autre fondée sur l'utilisation d'un arbre de prédiction de suffixes. La phase d'apprentissage consiste à faire analyser au système des morceaux sous la forme de fichiers MIDI, pour permettre le calcul des probabilités d'application des règles. Le système peut ensuite passer à une phase de génération. Les auteurs concluent que le mécanisme à base de grammaire a tendance à copier des blocs complets de notes des morceaux appris, en les associant à d'autres sans grande cohérence, alors que l'autre mécanisme produit des résultats plus inventifs, mais permet l'apparition de plus de « fausses notes ».

Plus généralement, les limitations des grammaires génératives sont proches de celles qu'on rencontre avec les modèles Markoviens. Dans un cadre polyphonique, il est très difficile de prendre en compte les relations entre les différentes voix jouées simultanément. Elles sont donc plutôt adaptées au traitement séquentiel de l'information, comme pour la production de mélodies dans un contexte donné, l'analyse ou la production de suite d'accords. La production de musique en ligne n'est pas possible puisqu'il faut attendre la fin de l'application des règles pour obtenir le morceau complet, sauf dans le cas des L-System qui sont, eux, à même de produire en temps réel. Une fois les règles de la grammaire établies, il n'est plus possible d'influencer la production du système. De la même manière qu'avec les modèles Markoviens, les résultats sont prévisibles et suivent tous un même style dépendant de la définition des règles.

L'intérêt des grammaires formelles pour la production musicale automatique se situe dans une assez large palette d'expressions possibles à l'aide d'une série compacte de règles de production.

3.2.3 Automates cellulaires

L'utilisation d'automates cellulaires dans un cadre de production musicale a, dans un premier temps, été réservée à la synthèse sonore, c'est-à-dire à la production du son lui-même. *LASy* et *Chaosynth* sont deux outils qui illustrent ces mécanismes.

Le premier, *LASy* (pour *Linear Automata Synthesis*) [Chareyron, 1990] utilise un automate cellulaire en combinaison avec un algorithme de synthèse sonore. Ces algorithmes, utilisés dans les synthétiseurs, sont souvent d'un paramétrage long et complexe. Dans cette approche, l'automate sert à fixer les paramètres de l'algorithme de synthèse sonore. Dans le cas de *LASy*, la synthèse est faite par combinaison d'échantillons. L'automate cellulaire, à une dimension, détermine les échantillons utilisés.

Chaosynth [Miranda, 2007] produit, quant à lui, directement les éléments de base de synthèse granulaire : des échantillons de sons très courts, d'une durée de l'ordre de la dizaine de millisecondes qu'on appelle grains. Ces grains concaténés, dans une enveloppe sonore donnée, produisent le timbre, la texture du son. L'enveloppe détermine la hauteur, la durée et le volume de la note. *Chaosynth* est basé sur un automate à deux dimensions, séparé en plusieurs portions, chaque portion servant à la configuration d'une composante du grain sonore. Chacune de ces composantes est une onde sonore, dont la fréquence et l'amplitude dépendent des valeurs des cellules correspondantes. Elles sont assemblées pour former le grain qui détermine la nature du son produit.

Eduardo R. Miranda propose d'étendre l'utilisation d'automates cellulaires à la production de pièces musicales. C'est l'objectif de son logiciel appelé *Cellular Automata Music (CAMUS)* [Miranda, 2007]. Il s'agit d'un système constitué de deux automates. Le premier est une grille à deux dimensions, dans laquelle chaque cellule définit une série de trois notes. Les règles qui s'appliquent sont celles du jeu de la vie de John H. Conway [Gardner, 1970]. Partant d'une note de base définie par l'utilisateur pour chaque cellule, on obtient la seconde en décalant la hauteur de la première note d'autant de demi-tons que l'ordonnée de la cellule. La troisième est calculée de la même façon en utilisant l'abscisse de la cellule. Les cellules de cet automate peuvent prendre deux états : vivantes ou mortes. Le second automate dispose de plusieurs états, qui correspondent aux timbres MIDI utilisés. C'est aussi un automate à deux dimensions, comportant autant de cellules que le premier automate. Ses règles sont celles de la croissance cristalline en spirale de David Griffeath [Dewdney, 1981]. Le premier automate est parcouru, ligne par ligne, et pour chaque cellule vivante, on joue la série de notes correspondante, avec le timbre de l'état de la cellule de mêmes coordonnées sur le second automate. Les durées des notes sont calculées en combinant les états des cellules voisines. Le schéma Fig. 3.2 donne l'identifiant des cellules voisines. Les opérations binaires $Tgg = abcd$ OU $dcba$ et $Dur = mnop$ OU $ponm$ donnent respectivement les indices dans une table de la répartition du début (Tgg) des trois notes du bloc, et de la fin (Dur) de ces trois notes.

Pour illustrer ce fonctionnement, le schéma de la Fig. 3.3 représente deux automates

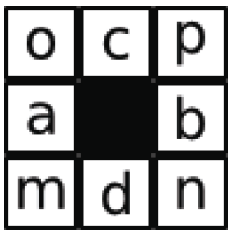


FIG. 3.2 – Schéma de calcul des positions et durées de notes de *CAMUS*

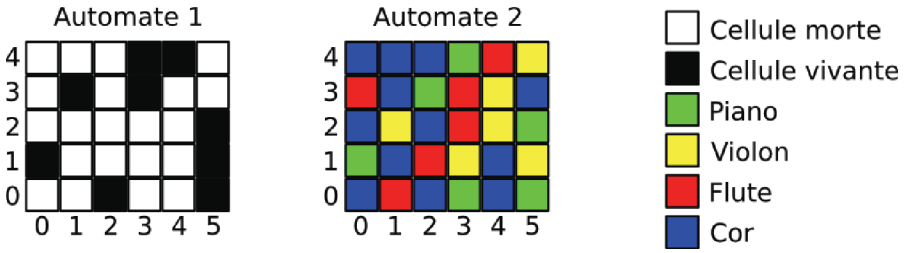


FIG. 3.3 – Schéma de fonctionnement de *CAMUS*

tels que décrits par Eduardo R. Miranda. Le processus parcourt l'automate 1, calcule les notes et cherche sur l'automate 2 les timbres correspondants.

- cellule (3,4) :
  - note de base : do ;
  - Série de 3 notes : do - ré# - mi ;
  - timbre correspondant : piano ;
- cellule (4,4) :
  - note de base : ré ;
  - série de 3 notes : ré - fa# - la# ;
  - timbre correspondant : flute ;
- cellule (1,3) :
  - note de base : sol ;
  - série de 3 notes : sol - sol# - si ;
  - timbre correspondant : cor ;
- cellule (3,3) :
  - note de base : mi ;
  - série de 3 notes : mi - sol - la# ;
  - timbre correspondant : flute ;
- cellule (5,2) :
  - note de base : la ;
  - série de 3 notes : la - ré - mi ;
  - timbre correspondant : piano ;
- cellule (0,1) :
  - note de base : do ;
  - série de 3 notes : do - do - do# ;

  - timbre correspondant : flute ;
- cellule (5,1) :
  - note de base : si ;
  - série de 3 notes : si - mi - fa ;
  - timbre correspondant : violon ;
- cellule (2,0) :
  - note de base : fa ;
  - série de 3 notes : fa - sol - sol ;
  - timbre correspondant : cor ;
- cellule (5,0) :
  - note de base : mi
  - série de 3 notes : mi - la - la ;
  - timbre correspondant : piano.

Les différentes notes sont jouées, après calcul de leur répartition au sein du bloc, à partir de l'état des cellules voisines.

CAMUS produit, selon son auteur, des passages musicaux peu convaincants car non conformes à ce que produit normalement l'instrument utilisé. Il en conclut que, bien que pouvant être utilisés pour générer des morceaux de musique, les automates cellulaires sont plus pertinents pour la synthèse sonore. En effet, il est particulièrement difficile de guider le résultat que l'on va obtenir, du fait de la complexité du comportement d'un automate cellulaire. De plus, une fois les règles écrites, la musique produite suit toujours la même forme, sans possibilité d'adaptation.

Un autre exemple, basé, entre autres, sur un automate cellulaire, est le fruit des recherches d'Alice Eldrige. Son projet *Adaptative Systems Music* [Eldridge, 2002] utilise des algorithmes bio-inspirés ayant des capacités d'auto-régulation pour une composition musicale automatique. La structure rythmique est produite par un automate cellulaire à une dimension où chaque cellule correspond à une fraction de temps. À l'état 1, une note est jouée, un silence, à l'état 0. Un réseau homéostatique est en charge de définir la partie harmonique de la composition. Ce type de réseau est constitué d'unités inter-connectées. L'état de chaque unité à l'instant t est fonction des valeurs pondérées des états de toutes les unités à l'instant $t-1$. Le réseau utilisé ici est une combinaison de plusieurs réseaux. Le système opère en temps réel par un processus itératif. Il produit trois parties différentes :

- un accord à 4 sons maintenus ;
- une partie rythmique où le placement des notes résulte de l'automate cellulaire ;
- une partie mélodique où quatre notes sont jouées en boucle, chacune avec une période tirée au hasard.

Le réseau homéostatique produit les hauteurs des notes des trois parties. Il est perturbé à chaque fois qu'il se stabilise, pour relancer une nouvelle progression harmonique. La partition d'un exemple de résultat est reproduite à la figure Fig. 3.4.

3.2.4 Programmation par contrainte

L'approche de Martin Henz, Stefan Lauer et Detlev Zimmermann [Henz et al., 1996] se base sur la programmation par contrainte. Ils utilisent le système de programmation par contrainte *Oz* pour la production automatique d'œuvres musicales à quatre voix. Le but

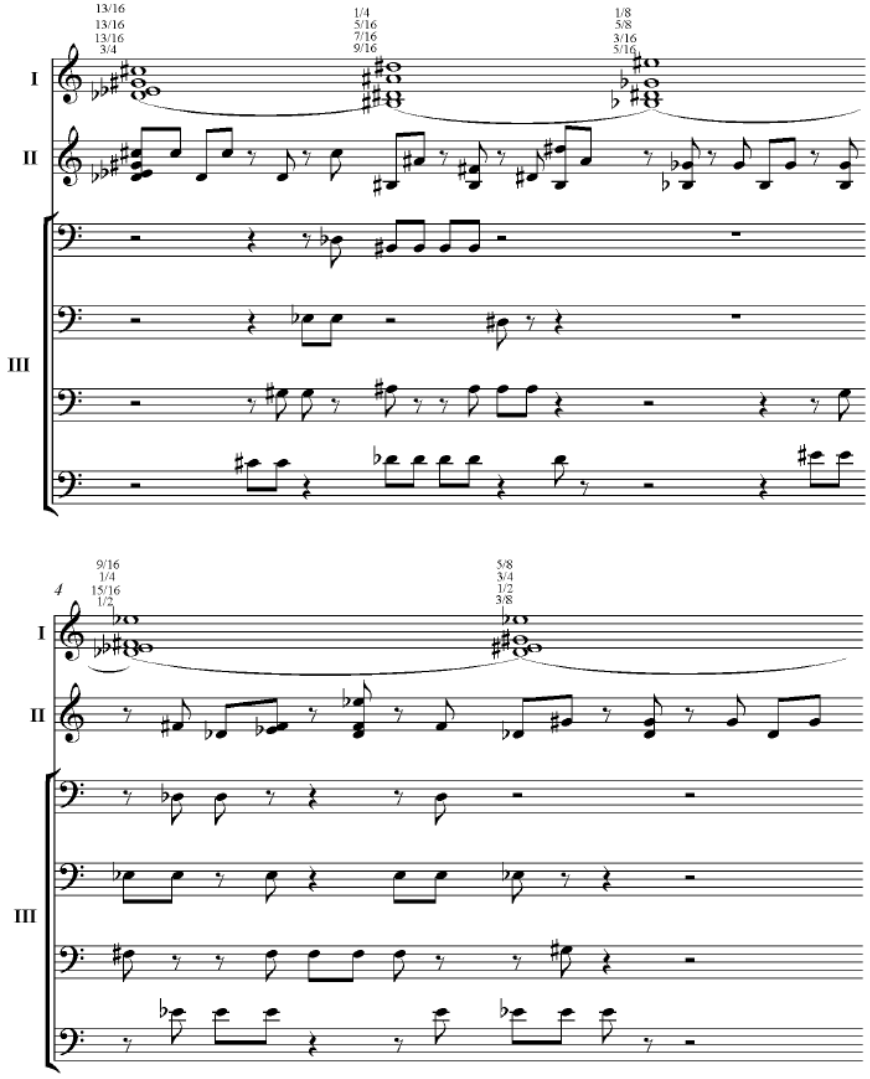


FIG. 3.4 – Partition produite par *Adaptative Systems Music*

du projet est de produire une musique, pas entièrement automatique, mais conforme à des intentions formulées par un utilisateur humain. Le moteur génératif *COMPOzE* est couplé à *AARON* qui génère les arrangements. *AARON* est en charge de produire la progression harmonique (métrique, structure harmonique et rythmique). *COMPOzE*, lui, produit les évènements MIDI à partir de cette structure.

Les intentions de l'utilisateur prennent la forme d'un plan musical, qui consiste en une suite de directives dites fonctions harmoniques. Ces fonctions harmoniques peuvent être, par exemple, l'utilisation d'une gamme majeure, ou mineure. En partant de ce plan musical, *COMPOzE* produit un ensemble de règles de composition qui vont être traduites

sous forme d'un problème de satisfaction de contraintes dont :
- les variables sont le nombre d'accords multipliés par le nombre de voix ;
- le domaine de chaque variable est la tessiture de chaque voix ;
- les contraintes sont les fonctions harmoniques et les règles de composition.

Ce problème est résolu par le solveur *Oz*.

L'utilisation de la programmation par contrainte revient donc à formuler le processus de composition sous la forme d'un ensemble de règles et de contraintes. De l'avis des auteurs, les résultats obtenus sont harmoniquement plaisants, mais d'un style très rigide. La génération d'une solution, donc d'un morceau est assez rapide, mais il n'est pas envisageable de générer des morceaux en temps réel avec ce genre de système.

3.2.5 Réseaux de neurones artificiels

Les réseaux de neurones artificiels sont fréquemment utilisés dans le contexte de la génération automatique de pièces musicales. On peut citer, entre autres, les travaux de Michael C. Mozer [Mozer, 1994]. Partant d'une étude des méthodes mathématiques pour la composition automatique, notamment de tables de transition de Kohonen, et du constat qu'elles nécessitent la manipulation de données très volumineuses (d'un rapport exponentiel avec la taille du problème) et qu'elles sont incapables de traiter les dépendances temporelles entre les évènements, il propose une autre approche. Son système, *CONCERT*, est basé sur un réseau de neurones récurrents. On lui présente une note en entrée, et il fournit la note suivante. Chaque note est représentée par sa hauteur, sa durée et l'accord qui l'accompagne. Une fois l'apprentissage effectué, on peut entrer une séquence de départ, puis brancher la sortie de *CONCERT* sur son entrée pour le laisser générer automatiquement une suite de notes. En pratique, la sortie de *CONCERT* fournit une série de notes probables, il s'assortit donc d'un processus de sélection parmi ses différentes sorties.

L'approche de Chun-Chi J. Chen et Risto Miikkulainen [Chen and Miikkulainen, 2001] est assez proche. Mais leur réseau de neurones est produit par un algorithme évolutionnaire où l'opérateur de sélection est basé sur des règles de compositions harmoniques et rythmiques. Le réseau utilisé est récurrent, l'utilisation d'une mémoire permettant l'apparition de structures répétitives. La hauteur des notes y est relative, plutôt qu'absolue, pour deux raisons. La première est une analogie avec les capacités humaines, l'oreille est beaucoup plus sensible aux intervalles qu'aux hauteurs absolues. L'oreille absolue est, d'ailleurs, extrêmement rare. La seconde est l'expérience personnelle des auteurs, dont les premiers résultats sont plus satisfaisants musicalement avec cette approche relative. Les neurones de sortie du réseau sont scindés en deux parties. La première partie où chaque neurone représente chacun intervalle. On examine la valeur associée à chacun d'entre eux. L'intervalle entre la prochaine note jouée et la précédente est celui qui correspond au neurone de plus forte valeur. La seconde partie où chaque neurone représente une durée de note. La durée de la prochaine note est celle qui correspond au neurone ayant la plus forte valeur. La représentation des mesures se fait par la concaténation de 16 paires de durées et hauteurs jusqu'à ce que le temps total de la mesure soit atteint, le cas échéant, la durée de la dernière note est ajustée. L'algorithme évolutionnaire *SANE* (Symbiotic, Adaptive Neuro-Evolution), spécialisé dans l'évolution de réseaux neuronaux est utilisé. L'évaluation se fait en suivant deux types de règles :

– Celles qui caractérisent le style de composition
– Celles qui proviennent de la théorie musicale

Douglas Eck [Eck and Schmidhuber, 2002], dans son étude sur la composition musicale par réseaux de neurones artificiels, affirme que les réseaux de neurones récurrents classiques ne sont pas capables d'une organisation temporelle suffisante à la construction d'un morceau cohérent. Il propose l'utilisation de réseaux de neurones de type LSTM (Long Short-Term Memory) pour résoudre ce problème et améliorer la composition automatique, notamment pour offrir plus de structure aux morceaux produits.

De manière générale, les réseaux de neurones nécessitent beaucoup d'informations en entrée et il est délicat de faire le tri entre ce qui est pertinent et ce qui l'est moins. Même s'ils peuvent produire une structure locale, ils ne peuvent que difficilement développer de véritables structures de haut niveau. Plus encore que dans d'autres contextes, la phase d'apprentissage peut poser problème. De par sa durée et la technicité qu'elle demande, elle ne peut être effectuée, le plus souvent, que par le concepteur du système. Or, cette phase conditionne grandement les résultats obtenus. Dans le cadre d'un outil de composition qui ne soit pas qu'à l'usage de son concepteur, cela peut engendrer une frustration chez l'utilisateur de l'outil. D'autant qu'on rencontre, comme ailleurs, les problèmes classiques de sur-apprentissage et de difficulté à généraliser. Bien qu'assez complexes, les réseaux de neurones peuvent s'adapter à la production en temps réel. Malgré tout, ces méthodes produisent des résultats plus surprenants et originaux que ceux qu'on obtient avec les méthodes étudiées précédemment.

3.2.6 Algorithmes évolutionnaires

Les algorithmes évolutionnaires apparaissent comme les outils les plus utilisés pour la génération musicale. John Biles [Biles, 2007b] propose quelques règles d'adaptation des algorithmes génétiques à la composition musicale automatique.

La phase d'évaluation peut être automatique, basée sur des règles. Ces règles peuvent, ainsi, être définies par le concepteur, telles que celles utilisées par le logiciel *AMUSE* [Özcan and Erçal, 2007], qui sont décrites plus loin. Néanmoins, établir un ensemble de règles pouvant définir une *bonne* musique semble un peu audacieux, aussi envisage-t-on le passage à une évaluation interactive, en requérant l'avis d'un auditeur. Cette évaluation interactive, plus pertinente, peut constituer un potentiel goulot d'étranglement temporel, aussi faut-il la concevoir efficacement. En effet, l'écoute de passages, ou de morceaux de musique complets est longue et peut vite devenir ennuyeuse. Il apparaît donc exclu de proposer l'écoute d'un nombre important d'individus musicaux.

L'opérateur de croisement a pour but de combiner les meilleures parties de chaque individu pour produire un individu fils meilleur au regard de l'évaluation. Habituellement, c'est le hasard qui détermine les parties des parents choisies. Mais on ne génère pas forcément un individu meilleur dans ce cas-là. Il apparaît donc nécessaire d'avoir un opérateur de croisement avec une aptitude musicale dans le choix du point de croisement comme dans la façon d'échanger les matériaux des parents.

Contrairement à ce qui se passe en optimisation, il est préférable d'avoir un opérateur de mutation intelligent, musicalement bien sûr. Selon Biles, une approche interactive de la

mutation est une excellente solution. L'utilisateur altère un individu à la main.

Pour augmenter la diversité de la population, il faut s'assurer que les remplacements vérifient que des différences significatives existent entre les individus.

Voyons maintenant comment sont construits quelques systèmes basés sur des algorithmes génétiques.

GenJam (Genetic Jammer) [Biles, 1994] est un système interactif de production de solo de jazz basé sur un algorithme génétique. Il fonctionne dans un cadre préalablement défini, à partir d'une grille d'accords donnée, pour une signature rythmique donnée, en se basant sur des phrases de solos pré-construites. La population a une structure hiérarchique : une population de phrases construites à partir d'une population de mesures.

– 64 individus mesures ;
– 48 individus phrases, chaque phrase étant composée de 4 mesures.

En temps réel, un mentor humain évalue la mesure et la phrase jouée sur une échelle simple. Soit la phrase est considérée comme bonne et on incrémente son score, soit elle est mauvaise et son score est décrémenté.

Un individu phrase est codé sous la forme des 4 indices des mesures qui le composent auxquels on accole son score. Un individu mesure est composé d'un nombre d'évènements MIDI qui correspond à une pleine mesure de croches (8 évènements en 4/4). La valeur d'une note correspond à l'une des 14 hauteurs possibles ou à l'une des deux valeurs particulières pour les silences ou pour continuer l'évènement précédent. *GenJam* dispose, ainsi, d'une tessiture de presque deux octaves.

Dans son mode évolutionnaire, *GenJam* remplace 50% de la population à chaque génération, selon une méthode élitiste. Les mutations sur les mesures sont différentes de l'opérateur habituel. Ici, on change totalement la mesure selon une des quelques méthodes mises en place. Elles peuvent inverser la progression harmonique, transposer la phrase, allonger une note, etc. Les mutations sur les phrases inversent ou permutent le sens des phrases, procèdent à la répétition d'une des mesures en choisissant celle de plus haute évaluation, etc.

L'opérateur de croisement de mesure choisit un point de croisement, forcément entre deux notes, et inverse les parties finales des parents. Ce point de croisement est choisi suivant plusieurs critères (inclusion du maximum d'apport humain, ne pas récréer des mesures existantes, etc.). Sur les phrases, les croisements peuvent seulement se faire entre les mesures. L'initialisation de la population est faite par un générateur aléatoire guidé (un générateur fractal dans un premier temps, remplacé dans un second temps par des chaînes de Markov) pour éviter une longue phase d'évaluation de phrases aléatoires. Finalement, dans les dernières versions, l'initialisation est faite par des phrases de solo pré-construites

En effet, *GenJam* a beaucoup évolué depuis ses premières versions. Dans une version plus récente [Biles, 2002], pour développer l'interactivité, un nouveau mode de fonctionnement est apparu, reprenant le concept jazz des *trading fours* : 4 mesures jouées par le musicien humain, 4 mesures jouées par *GenJam*. Un convertisseur MIDI permet au système d'enregistrer ce que fait le soliste qui joue avec lui. Il transforme ces 4 mesures à l'aide de son opérateur de mutation et joue le résultat. Dans ces dernières évolutions, les fonctions de croisement et de mutation ont évolué pour intégrer de plus en plus de connaissances. L'objectif est de pouvoir se passer de la phase d'évaluation. Il ne s'agit donc plus à proprement

parler d'un algorithme génétique.

Peter M. Todd et Gregory M. Werner [Todd and Werner, 1999] proposent un système de composition musical automatique où les mécanismes évolutionnaires permettent de générer des compositions musicales, mais aussi de générer le système qui produit ces compositions. Ainsi, un premier processus évolutionnaire produit un second qui génère, lui, les morceaux de musique. Les auteurs identifient, eux aussi, le goulot d'étranglement que représente une évaluation humaine dans le processus évolutionnaire. Ils insistent d'ailleurs sur la nécessité de contraindre les individus générés à des règles musicales fortes, pour faciliter cette étape, car l'écoute d'une grande quantité de suites incohérentes de sons est particulièrement pénible. Finalement, la solution qu'ils choisissent est une évaluation automatique qui fait intervenir dans le processus évolutionnaire des individus mâles qui constituent les compositions générées et des individus femelles qui les évaluent. Les femelles choisissent les mâles qui deviennent les parents de la génération suivante. L'objectif atteint de ce système est d'obtenir une population de composition présentant de notables différences entre elles. Néanmoins, les auteurs concluent que cette diversité est au prix de la structure musicale et que les résultats obtenus ne présentent pas un grand intérêt artistique.

GenDash de Rodney Waschka II [Waschka II, 2007] est lui aussi un outil de composition, il ne s'agit pas ici d'improviser de la musique, les morceaux sont produits par l'algorithme pour être joués ultérieurement. Nous pouvons résumer *GenDash* en quelques caractéristiques :

1. un individu est une mesure ;
2. tous les individus d'une génération sont joués ;
3. la fonction d'évaluation utilisée est aléatoire ;
4. il n'y a qu'un point de croisement, mais sa position change à chaque génération ;
5. on conserve les individus qui n'ont pu être écoutés à une génération pour leur permettre d'apparaître plus tard ;
6. les individus peuvent se croiser avec plusieurs autres individus, même plusieurs fois ;
7. il y a des mutations ;
8. c'est un modèle à deux parents, mais sans considération de sexe ;
9. la population initiale est choisie par l'utilisateur.

GenOrchestra [De Felice and Abbattista, 2002] est, quant à lui, pourvu de deux mécanismes d'évaluation, une automatique et une interactive. Les meilleures propositions sélectionnées par l'algorithme sont publiées en ligne pour que les internautes les évaluent. Chaque individu est composé de plusieurs chromosomes correspondants chacun à une partie : basse, mélodique ou harmonique. Les opérateurs consistent en des transpositions, inversions de séquence, rotations vers la droite, inversions harmoniques d'évènements ou tris des notes d'une séquence. L'évaluation automatique se base sur la consonance des différentes parties et sur la validité métrique, qui caractérise la présence de notes sur les temps forts.

Tatsuo Unemi et Eiichi Nakada [Unemi and Nakada, 2001] proposent, eux aussi, un système de composition basé sur des techniques évolutionnaires, appelé *Sbeat*. Ils utilisent un algorithme évolutionnaire interactif dont les individus sont des tableaux d'octets, à 3

lignes et 16 colonnes, représentants de courts extraits musicaux à trois voix. Chaque octet comporte deux parties de 4 bits, une pour coder la durée, et l'autre pour la hauteur de la note. Les individus générés sont présentés à l'utilisateur qui choisit les parents de la prochaine génération. Ce choix s'effectue à partir de la partition ou d'une écoute de la phrase produite, comme on peut le voir sur la capture d'écran FIG. 3.5. Les opérateurs de croisement et de mutation sont ceux d'algorithmes génétiques classiques. Le croisement opère un échange d'une partie de chaque parent pour obtenir les fils. La mutation modifie un bit d'un individu. Leur programme permet ainsi de produire une sélection de courtes phrases musicales se rapprochant par étapes successives des préférences de l'utilisateur.

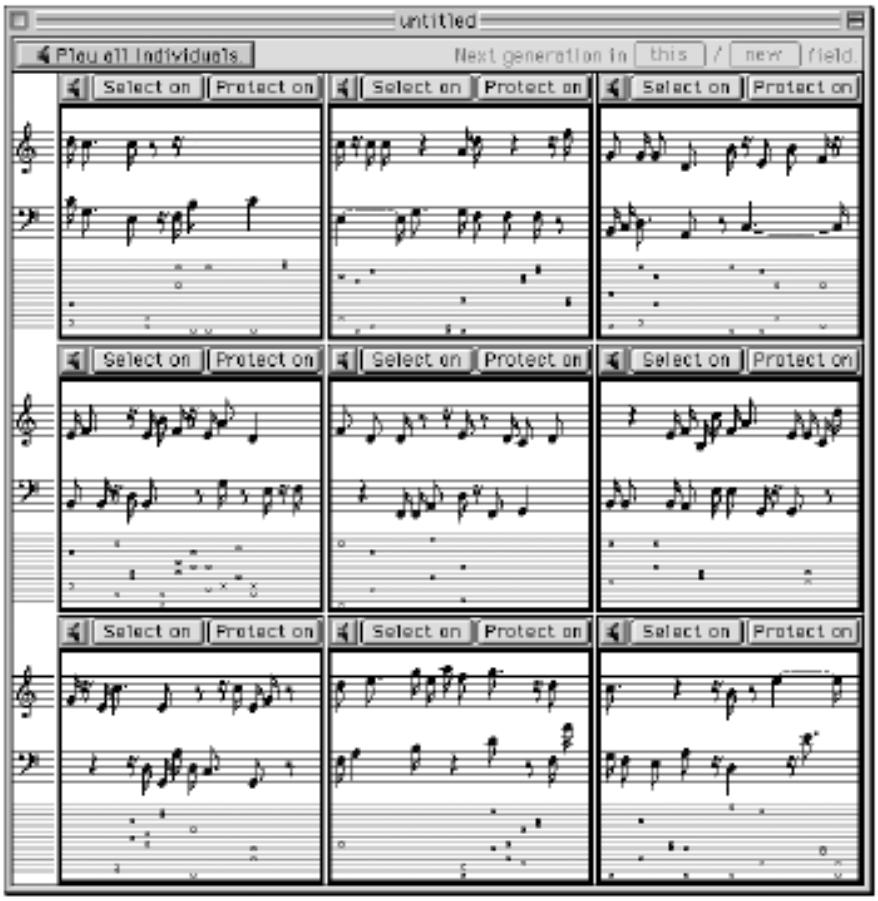


FIG. 3.5 – Interface de séléction de *Sbeat*

Autre exemple, *AMUSE* [Özcan and Erçal, 2007] génère des improvisations, à l'aide d'un algorithme évolutionnaire, dans un contexte harmonique donné. L'idée est d'offrir un outil créatif, où les choix de l'utilisateur paramètrent la création. Ici, ces choix s'effectuent en amont, puis le programme fournit de façon autonome le résultat. *AMUSE* prend en paramètre un fichier MIDI qui contient l'accompagnement sur lequel un solo sera généré. Le programme réclame aussi que lui soit fournit, soit la suite d'accords qui compose le morceau, avec la durée de chaque accord, soit la gamme qu'il doit utiliser pour improviser le solo.

La fonction d'évaluation dépend de multiples critères, auxquels sont associés des poids. Certains sont internes au système : correspondance avec les notes de l'accord, relations entre les notes, direction des notes, note de début, note de fin, intervalle supérieur à la quinte, changements brutaux de durée, etc. D'autres sont dits ajustables car l'utilisateur peut en définir le poids : la proportion de silence, de pause, la similarité à un patron, etc. Les résultats produits sont très contraints, notamment par les fichiers d'entrées. Ils sont donc assez mélodieux, mais finalement peu surprenants.

Palle Dahlstedt [Dahlstedt, 2001] pour son installation *MutaSynth* couple le synthétiseur sonore éponyme à un algorithme évolutionnaire. Une population de sons générés est jouée au spectateur, qui choisit ses préférés, entraînant la création d'une nouvelle génération, dans un processus en boucle. Selon Dahlstedt, une des principales difficultés de la composition automatique est le codage du génotype, ou, comment modéliser le phénomène d'une façon informatiquement efficace, mais sur laquelle les variations peuvent être signifiantes. On peut alors parler de représentations objets quand on représente les informations structurelles, voire de représentations génératives si on formalise les relations entre les objets. Pour le son en lui-même, on peut coder les paramètres d'un synthétiseur, ou encore l'utilisation d'une batterie de synthétiseurs. MutaSynth fait usage de cette catégorie de méthodes et de représentations.

MusicBlox [Gartland-Jones, 2003] est un système de composition en temps réel, pouvant être contrôlé par des objets réels. Il est constitué d'un ensemble de cubes interagissant avec une table, lorsqu'ils y sont posés. Ces cubes permettent de construire des structures en trois dimensions en s'accolant les uns aux autres. À chaque cube est associée une phrase musicale. Si ce cube est placé sur la table de jeu, cette phrase est jouée. Si un autre cube vient s'accoler, les deux phrases sont jouées simultanément. Les cubes disposent d'un bouton qui déclenche l'envoi de leur phrase musicale aux cubes proches sur la table. Ceux-ci vont faire évoluer leurs propres phrases en les combinant avec celles qu'ils reçoivent en utilisant un algorithme évolutionnaire inspiré de *GenJam*.

Zong Woo Geem et Jeong-Yoon Choi [Geem and Choi, 2007] utilisent un algorithme évolutionnaire de recherche harmonique. Ils s'inspirent des techniques d'improvisation de la musique médiévale. Une fonction d'évaluation objective, basée sur des règles harmoniques, leur sert de critère d'optimisation. Chaque voix dispose d'un ensemble de notes possibles, elle va choisir parmi celles-ci au hasard, ce choix étant néanmoins guidé par ce qui a été joué précédemment. Une fois la note choisie, elle peut être substituée par une de ses voisines, selon une probabilité fixe. L'accord créé par toutes les voix est évalué en regard de considérations harmoniques, qui viendront ajuster les probabilités de chaque note de l'accord d'être choisie. Le nombre d'accords possibles est fixe, et chaque nouvel accord créé, viendra remplacer celui de plus faible valeur, s'il est meilleur bien sûr.

Des algorithmes évolutionnaires sont aussi utilisés, comme on l'a déjà vu, pour faire évoluer un réseau de neurones artificiels [Chen and Miikkulainen, 2001] ou pour d'autres cas. Qijun Zhang et Edouardo R. Miranda [Zhang and Miranda, 2007] les utilisent, par exemple, pour la génération de profils d'interprétation. Il ne s'agit donc pas ici de produire ce qui sera joué, mais de produire la façon dont ce sera joué. Ils s'intéressent donc à générer l'intensité de jeu des notes ainsi que de petites variations de rythmes ou d'amplitudes dans l'interprétation d'un morceau, ainsi que le ferait un véritable interprète humain.

Comme on vient de le voir, les méthodes basées sur les algorithmes évolutionnaires pour la création musicale automatique sont nombreuses et diverses. On trouve même des études [Gartland-Jones, 2002, Gartland-Jones and Copley, 2003] visant à montrer leur pertinence pour cette tâche. Elles concluent sur l'efficacité des méthodes évolutionnaires, en mettant en exergue les difficultés classiques de ces méthodes, à savoir :

- les résultats convaincants mais très formatés obtenus avec des évaluations automatiques ;
- la mise en place d'une évaluation interactive qui ne soit pas un goulot d'étranglement temporel ;
- le choix des individus de la population qu'il convient de jouer réellement.

La composition musicale peut s'apparenter à une forme de recherche dirigée. On compose en transformant une idée musicale pour l'amener à une idée neuve et qui correspond à ce qu'aime l'auteur. Le mécanisme de recherche des algorithmes évolutionnaires, basé sur des mutations et des recombinaisons, est donc particulièrement adapté. Un compositeur peut utiliser ces mécanismes de parcours d'espace de recherche pour développer ses idées.

Les difficultés dans l'utilisation de ces méthodes viennent d'abord du modèle que l'on adopte pour représenter la musique, et donc définir les individus et la population de l'algorithme. C'est sur cette question particulière, de la représentation de la musique que se différencient le plus les solutions proposées dans la littérature des algorithmes évolutionnaires utilisés pour la génération musicale. Les opérateurs génétiques, fortement dépendants du modèle, constituent, eux aussi, un point particulier de divergence entre les auteurs. Il apparait néanmoins important d'utiliser des opérateurs ayant des affinités musicales plutôt que de simples mélanges et transformations de bits dans la représentation des individus. La fonction d'évaluation est le plus souvent interactive, même des tentatives pour automatiser cette évaluation des bons individus de la population existent. Cette évaluation a un impact majeur sur les résultats obtenus, aussi une évaluation automatique est-elle une limite aux possibilités créatives de ces systèmes. L'évaluation interactive n'est pourtant pas exempte de défaut, principalement en ce qu'elle constitue un fort goulot d'étranglement pour ces méthodes. En effet, l'évaluation d'un individu par l'utilisateur du système peut prendre du temps, car cela dépend de l'aptitude de l'utilisateur à écouter puis évaluer une séquence musicale, aussi n'est-il pas envisageable d'en faire évaluer un grand nombre.

3.2.7 Systèmes multi-agents

Rodolfo Daniel Wulfhorst et al. [Wulfhorst et al., 2001] posent, dans leur article, les bases d'un système multi-agents basé sur la norme MIDI. Des agents musicaux interagissent ensemble via des messages MIDI pour improviser. Ils mettent en évidence les problèmes de contraintes temporelles et la difficulté de la détection de tempo. Les expérimentations rapportées font état de solutions imparfaites, mais néanmoins efficace dans la majorité des cas pour résoudre ces difficultés.

Les travaux de Tim Blackwell [Blackwell and Young, 2004], avec comme premier objectif d'utiliser des essaims de particules comme système de paramétrage automatique de synthétiseurs sonores, évoluent ensuite [Blackwell, 2006, Blackwell, 2007] vers de l'improvisation musicale automatique. Sa représentation de la musique sous forme d'essaim pose l'agent comme une note évoluant dans un espace musical. Les déplacements de ces notes

font évoluer leur hauteur, volume, durée et position dans la mélodie, et s'effectuent par attractions mutuelles des individus, en évitant toutefois les collisions. Les attractions et répulsions peuvent suivre des règles harmoniques de consonances et dissonances, ou rythmiques, de placement sur des divisions justes de temps. La structure globale provient de ces mécanismes d'interactions locaux.

Daniel Bisig travaille lui aussi sur les essaims de particules [Bisig et al., 2007]. Le projet *Interactive Swarm Orchestra* fait interagir, avec les spectateurs, des particules définissant les paramètres d'un synthétiseur sonore, ainsi que les positions de ces sons dans un espace 3D. Un projecteur affiche les positions des particules et une caméra capte les mouvements de l'audience, qui peuvent ainsi repousser les particules.

Les résultats obtenus avec les systèmes multi-agents sont divers et souvent surprenants. Le principe de structure émergente est très prometteur dans le contexte de la composition musicale. Ces méthodes sont d'ailleurs parfaitement adaptées à l'improvisation en temps réel. Il est par contre nécessaire de guider en partie le comportement des agents pour éviter un résultat d'aspect trop aléatoire.

3.2.8 Autres méthodes bio-inspirées

AntMusic [Monmarché et al., 2003, Guéret et al., 2004, Monmarché et al., 2007] vise à produire des compositions polyphoniques MIDI en utilisant le comportement de déplacement collectif des fourmis. Une colonie de fourmis se déplace sur un graphe d'évènements midi. Chaque sommet correspond à une note, c'est-à-dire à une hauteur, une durée, un volume, etc. Les arcs entre ces sommets sont associés à une valeur : la quantité de phéromones sur cet arc. La probabilité qu'une fourmi choisisse un arc est liée à cette quantité de phéromones, mais aussi à la distance, en nombre de demi-tons entre les hauteurs de la note du sommet où elle se trouve et celle du sommet où elle se propose d'aller. Quand une fourmi arrive sur un sommet, elle joue l'évènement associé. La structure musicale globale du morceau émerge de l'intelligence collective de la colonie. Un graphe contenant toutes les notes possibles est de taille colossale, il apparait nécessaire de limiter cette taille. La première approche proposée consiste à offrir au compositeur qui utilise le programme, un ensemble de paramètres à choisir, qui limite la tessiture, les durées possibles, qui fixe le tempo, etc. Le graphe sera généré en fonction de ces paramètres. La deuxième approche décrite consiste à construire le graphe à partir d'une partition MIDI existante, en récupérant toutes les notes qui s'y trouvent. *AntMusic* pouvant ainsi composer des morceaux dans le style de celui qui aura servi à cette construction.

Daichi Ando et Hitoshi Iba [Ando and Iba, 2005] proposent d'utiliser le comportement d'un système bio-inspiré pour la génération de mélodies en relation avec un véritable pianiste. La performance d'un artiste contient un grand nombre d'informations dont il est possible de tirer parti pour influencer un système automatique. Le jeu d'un pianiste est capturé, traité, analysé puis envoyé au système génératif. Celui-ci produit des commandes musicales qui seront transmises à un synthétiseur sonore. Après un essai à base d'automates cellulaires à une ou deux dimensions, les auteurs ont expérimenté des systèmes imitant le comportement d'une nuée d'oiseaux ou de colonies de fourmis artificielles.

Alice Eldridge [Eldridge, 2005] a développé un système d'accompagnement automa-

tique en temps réel qu'elle utilise pour se produire dans un duo violoncelle et processus génératif. Il se base sur l'enregistrement du son produit par son instrument, traité par des algorithmes de vie artificielle en bassin. Le système fait intervenir un réseau homéostatique générant la recomposition des échantillons enregistrés, qui sont les objets que le simulateur de mouvements et de collisions fait interagir entre eux.

Ces méthodes sont parmi les plus récentes à être utilisées comme générateur musicaux. Dans les cas de méthodes à comportement émergent, telles que les algorithmes de colonie de fourmis artificielles, la structure des pièces produites résulte de ce comportement.

3.3 Étude de méthodes de génération de dessins

Nous décrivons maintenant quelques méthodes, souvent proches des précédentes, utilisées cette fois-ci pour de la génération de dessins. On y trouvera des techniques de création de formes, abstraites ou non, des méthodes de transformation d'éléments existants, etc.

3.3.1 Grammaires génératives

De nombreuses méthodes de création graphique basées sur des grammaires formelles s'inspirent du DOL-System de Prusinkiewicz [Prusinkiewicz and Lindenmayer, 1990]. Il s'agit d'un dérivé de L-System. Les symboles de la grammaire se voient associés à une signification graphique, telle que des commandes pour le langage Turtle. Elles permettent de produire des structures très organisées, gagnant en complexité à chaque nouvelle application des règles.

Les travaux de Bernd Lintermann [Lintermann and Deussen, 1999], fondés sur ces principes, permettent de modéliser et de générer des images de plantes tout à fait convaincantes. L'interface de son logiciel, basée sur la description d'une structure de type graphe où les nœuds sont les éléments graphiques et les arcs les règles les structurant, permet de créer intuitivement les règles du L-system sous-jacent.

Jon McCormack [McCormack, 2003] reproduit, lui aussi, des structures de plantes à l'aide de L-System. Les éléments produits par le L-System sont placés sur une surface de révolution pour obtenir une construction en 3D.

3.3.2 Automates cellulaires

Katie A. Bentley [Bentley, 2002] explore des mécanismes de formation de motifs. Elle propose notamment une étude de résultats produits par un automate cellulaire et un système décrit comme des agents phéromonaux. Un algorithme évolutionnaire est utilisé pour faire évoluer les paramètres de ces deux méthodes. L'automate cellulaire est de la taille du motif produit et comporte quatre états correspondant à quatre couleurs possibles. Elle propose différentes méthodes d'évolution de l'automate, soit cellule par cellule, soit plus classiquement toutes les cellules à la fois. Les huit règles de transformation constituent le génotype que fait évoluer l'algorithme évolutionnaire. Le système d'agents phénoménaux pose des agents au hasard sur le tableau motif. Elle change l'état de la case où elle se trouve en fonction des cases voisines puis se déplace en fonction d'une combinaison de l'influence

phéromonale et du contenu des cases voisines. Les phéromones de diffusent aux cases voisines au cours du temps. Les paramètres évolués par l'algorithme évolutionnaire comportent les règles de remplissage d'une case en fonction des voisines, la réaction aux phéromones des agents : suivi ou répulsion, et la capacité pour l'agent à déposer des phéromones ou non.

Des exemples de résultats obtenus figurent sur la figure FIG. 3.6. Les motifs en haut au centre et en bas à droite sont produits par les agents phéromonaux, les autres, par des automates cellulaires.

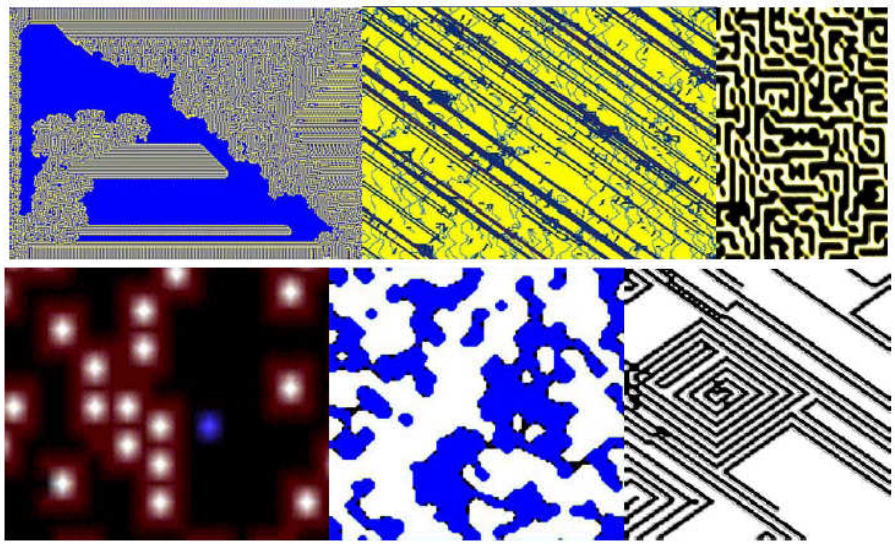


FIG. 3.6 – Exemple de motifs produits par K. Bentley

3.3.3 Algorithmes évolutionnaires

Hans E. Dehlinger [Dehlinger, 1998] explore les possibilités créatives des algorithmes génétiques. Dans son approche, les individus sont des lignes, ou plutôt des coups de crayons. Il génère une population en étant guidé par des règles de survie : une fonction d'évaluation qui permet de supprimer des individus de la population. Ces règles sont ce qui permet à l'artiste de guider l'outil génératif.

Les représentations de fractales ne sont pas dénuées d'un certain caractère esthétique. Dans ses travaux, Evelyne Lutton [Lutton, 1999] utilise des méthodes évolutionnaires pour modeler ces fractales. Le problème consiste à déterminer l'ensemble des paramètres d'entrée qui produit une sortie donnée. Dans un processus d'optimisation, cette sortie devient l'objectif dont la méthode cherche à se rapprocher. L'application à la résolution du problème inverse pour des systèmes de fonctions itératives permet ainsi de générer des images fractales se rapprochant d'images « objectifs ». Une version interactive de cette méthode permet de générer des images fractales guidées par un utilisateur. Un article plus récent [Lutton, 2006] décrit l'évolution de ces travaux vers le logiciel *ArtiE-Fract* et présente l'expérimentation de ce logiciel par deux artistes, ayant permis la production d'œuvres tout

à fait convaincantes. Les figures Fig. 3.7 et Fig. 3.8 reproduisent deux exemples de ces productions.



Fig. 3.7 – Danseuse d'Emmanuel Cayla produit avec *ArtiE-Fract*

Se concentrant sur les fractales de Mandelbrot, Jeffrey J. Ventrella [Ventrella, 2008] suit une approche similaire. Les individus de son algorithme évolutionnaire sont chacun un ensemble de paramètres qui permettent de configurer un rendu modifié d'un ensemble de Mandelbrot. L'algorithme est interactif, les individus nouvellement générés sont présentés à l'artiste qui leur attribue une note. La population est de taille constante, les nouveaux individus remplaçant ceux de plus faible évaluation. La particularité de cette approche est qu'il n'y a qu'un croisement par génération, et donc un unique nouvel individu, les deux parents étant choisis par tournoi au sein d'une sélection aléatoire. L'évaluation des individus baisse progressivement au cours du temps. Une seconde version de l'algorithme remplace l'évaluation interactive par une comparaison avec une image cible.

Certaines approches associent ces méthodes avec la robotique. Jon Bird et Dustin Stokes [Bird and Stokes, 2006] disposent de robots capables de détecter des marques au sol, et d'en ajouter. Des réseaux de neurones artificiels permettent le contrôle des robots. Ces contrôleurs constituent les individus d'un algorithme qui les fait évoluer. L'évaluation d'un

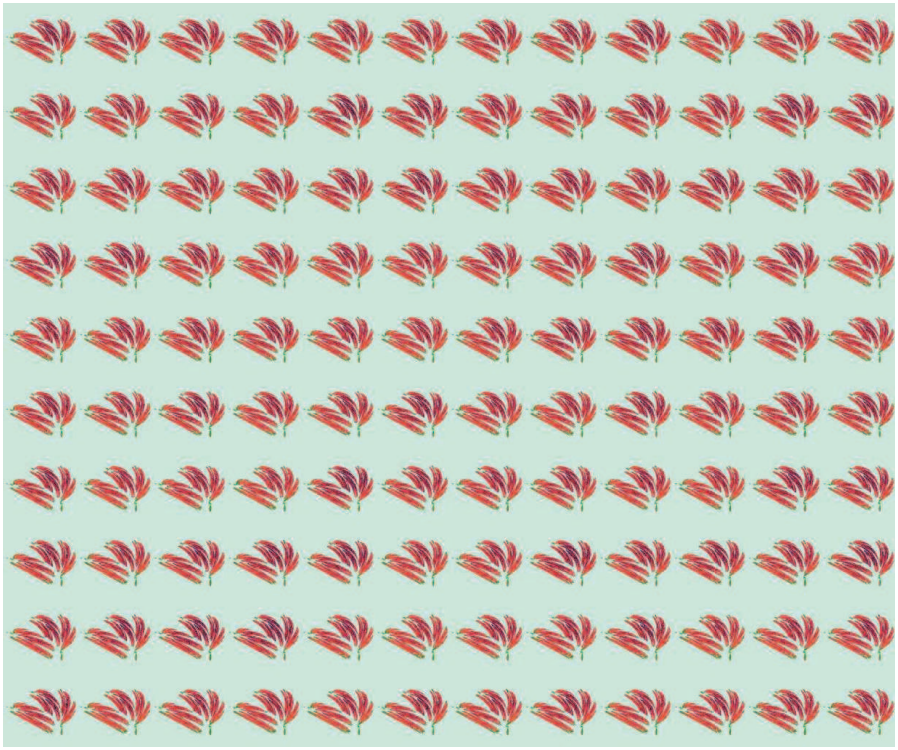


FIG. 3.8 – Papier peint de Marie-Amélie Porcher produit avec *ArtiE-Fract*

individu se fait à partir du comportement du robot observé.

3.3.4 Système multi-agents

La visualisation d'essaims de particules [Boyd et al., 2004, Nguyen et al., 2006] est une autre piste intéressante. Des particules mobiles évoluent dans un espace fermé. Cet espace fermé est une image, un fond noir. Les particules sont représentées par des points lumineux s'affichant sur ce fond noir.À l'aide d'un projecteur, on affiche cette image dans une zone pouvant accueillir des personnes. En interaction avec un système de capture de mouvements, les particules suivent les mouvements de l'artiste qui se trouve dans la zone de projection, ou des spectateurs dans le cas d'une projection au sein du public. Le comportement des particules est de suivre le mouvement des éléments proches de lui, tout en évitant les collisions avec les bords de la scène et les autres particules. Les mouvements capturés sont insérés dans l'espace des particules permettant, ainsi, de guider les particules par ce jeu d'attractions et de répulsions.

3.3.5 Algorithmes de colonie de fourmis artificielles

Associant l'artiste et le scientifique dans la conception du système, Leonel Moura et Henrique Garcia Pereira [Moura and Pereira, 2004] ont, eux aussi, créé des robots-peintres,

utilisant, cette fois-ci ; des mécanismes de déplacement inspirés du comportement des fourmis.

Sans passer par la robotique comme acteur du dessin, les fourmis peintres de Nicolas Monmarché [Monmarché et al., 2003] explorent les capacités créatives du comportement de déplacement collectif d'une colonie de fourmis. Une image initiale, qui peut être un fond uni où une image existante importée, constitue l'environnement sur lequel les fourmis se déplacent. À l'initialisation du programme, chaque fourmi est associée à une couleur, tirée au hasard. Lors des déplacements des fourmis sur l'image, elles déposent derrière elles une couleur. Ces couleurs font office de phéromones, ainsi chaque fourmi se voit attribuer une couleur recherchée. Elle a tendance à plutôt suivre cette couleur lors de ses déplacements. On assiste alors à une compétition en temps réel entre des fourmis qui évoluent sur un même environnement, et remplacent la couleur des pixels sur lesquels elles passent par leur propre couleur. L'utilisateur peut à tout moment arrêter ce processus, lorsqu'il obtient une image qui le satisfait. L'ensemble des paramètres de l'algorithme est très important, et influence grandement le résultat obtenu. Dans un premier temps, ces valeurs étaient simplement tirées au hasard. Une seconde version [Aupetit et al., 2003] introduit un mécanisme évolutionnaire interactif pour parcourir l'espace des paramètres en prenant en compte les goûts de l'utilisateur. Dans ce système, neuf images sont générées à partir de valeurs aléatoires. Ces images sont alors présentées à un utilisateur, qui choisit celles qui lui conviennent le mieux. L'ensemble des paramètres ayant créé ces images constitue les individus sélectionnés. Ils sont alors croisés ensemble pour donner naissance à une nouvelle génération de neuf images, plus conformes, du moins on l'espère, aux désirs de l'utilisateur.

Gary Greenfield [Greenfield, 2005] reprend cet algorithme de déplacement de fourmis classique en séparant la gestion des couleurs et celle des phéromones. L'idée est d'éviter qu'une fourmi n'en vienne à suivre son propre tracé, et surtout, de pouvoir faire réapparaître le concept d'évaporation, sans que le dessin n'en soit modifié. Dans le second article [Greenfield, 2006], il propose une gestion encore plus poussée des phéromones. Elles se diffusent sur la toile et sont consommées par les fourmis qui les suivent. Il obtient ainsi des résultats sensiblement différents.

Toujours sur le même modèle, Paulo Urbano [Urbano, 2007] exploite la notion de mimétisme chez les agents fourmis. Quand deux agents se rencontrent, ils échangent des informations sur leurs couleurs respectives et tendent à s'imiter.

La méthode de génération de peintures numériques, proposée par Elpida Tzafestas [Tzafestas, 2000], se base sur un autre comportement. Les fourmis se déplacent toujours sur une toile numérique, mais elles peuvent, ici, ramasser ou déposer de la couleur, comme lors du tri du couvain par les véritables colonies de fourmis.

La conclusion de cette étude bibliographique est, tout d'abord, que l'art génératif est manifestement un sujet foisonnant et en pleine effervescence. Un grand nombre de techniques différentes produisent des résultats très variés. Ces variations sont la conséquence de la liberté offerte par le champs d'application artistique. La musique peut, d'ailleurs, sembler être un domaine plus développé. Il n'en est rien, le déséquilibre entre les parties musicale et graphique n'étant dû qu'à la nécessité temporelle de privilégier un domaine par rapport à l'autre.

Chapitre 4

Accessibilité des logiciels

D'un point de vue informatique, l'accessibilité se traduit par un ensemble de bonnes pratiques servant de règles de conception, ou d'indicateurs pour l'évaluation d'applications dites accessibles.

Dans le cadre du web, la WAI édite un ensemble de bonnes pratiques : les *Web Content Accessibility Guidelines*. Ces WCAG 2.0 [Caldwell et al., 2008], dans leur dernière version, s'articulent autour de quatre principes :

Principe 1 : perceptible

Principe 2 : utilisable

Principe 3 : compréhensible

Principe 4 : robuste

Le premier principe, perceptible, s'intéresse à la transmission de l'information du site web à l'utilisateur. Il suggère la séparation du fond et de la forme, ainsi l'information contenue dans le site peut être présentée de plusieurs façons, certaines pouvant notamment être adaptées à des utilisateurs ne pouvant efficacement profiter de la mise en forme principale. Partant du postulat que l'information sous forme de texte est accessible car pouvant être, soit lue directement, soit adaptée par une loupe, un lecteur vocal, un afficheur braille, ou tout autre dispositif adapté, cette directive conseille de proposer des équivalents textuels à tout contenu non-textuel, tel que des images par exemple. De la même façon, les vidéos, extraits sonores, animations ou tout autre élément nécessitant une bonne audition ou une attention constante doivent disposer de substituts. Les informations importantes doivent être efficacement mises en exergue du contexte dans lequel elles sont, en pensant par exemple à adopter un contraste fort entre le fond et l'arrière plan.

Un site web doit être utilisable. Ce deuxième principe adresse les difficultés associées aux commandes nécessaires à l'usage du site. Il préconise que la souris ne doit pas être l'unique dispositif pour naviguer sur le site. Toutes les fonctionnalités doivent aussi pouvoir être activées au clavier. De plus, l'ensemble des éléments nécessaires à l'usage du site, c'est-à-dire à naviguer, à trouver un contenu donné ou à se repérer dans le site, doivent être disponibles en permanence. Par ailleurs, il faut faire particulièrement attention aux contenus mobiles ou éphémères. Ils doivent laisser suffisamment de temps pour que n'importe quel utilisateur puisse les lire et les utiliser. Cette directive recommande aussi d'éviter les éléments qui

clignotent à une fréquence de plus de trois fois par secondes, pour ne pas provoquer de crises chez les individus sensibles.

Le principe de compréhension vient compléter ceux liés à la perception et l'utilisabilité du contenu. Les textes doivent ainsi être compréhensibles. Il faut donc penser à en fournir des traductions et s'assurer qu'ils sont dans le langage le plus clair possible. La conception d'un site accessible doit s'assurer que son comportement est prévisible et conforme à l'intuition de l'utilisateur. Une assistance visant à l'aider à saisir et corriger les informations qui lui sont demandées fait aussi partie des recommandations de ce principe.

La robustesse concerne principalement la compatibilité avec différents matériels et logiciels. Une application web accessible se doit d'être fonctionnelle avec n'importe quel navigateur, quel que soit le dispositif pour y accéder, y compris dans le cas d'utilisation de technologies d'assistance spécifiques.

Ces directives liées au web peuvent servir à établir des recommandations à l'usage des concepteurs d'applications informatiques au sens large.

Sortant du strict domaine du web, et avant de s'atteler à suivre des règles de conception supplémentaire, l'utilisation des outils déjà existants est opportune. Un support de l'accessibilité est disponible dans certains langages de programmation. Java, par exemple, fournit un ensemble de classes appelé interface de programmation d'accessibilité Java ou *Java accessibility API*. Ces classes ont pour vocation d'assurer la communication entre l'application et des dispositifs d'assistance, quels qu'ils soient. Il incombe au programmeur de l'application d'associer les informations d'accessibilité aux éléments de son interface.

À un plus haut niveau, des framework pour le développement d'applications accessibles existent, tel que celui développé dans le cadre du projet TiM (*Tactile Interactive Multimedia*) pour le développement de jeux pour enfants malvoyants [Bertelle et al., 2003].

D'un point de vue général, les solutions d'accessibilité sont basées sur la multi-modalité, dont le principe général est de toujours fournir plusieurs moyens différents d'accéder à une information ou d'effectuer une action. Elles supposent l'utilisation de périphériques adaptés. Elles s'articulent autour de problèmes d'interfaces pour afficher ou contrôler l'application, de considérations liées aux modèles utilisés et d'accompagnement de l'utilisateur.

4.1 Interface Homme-Machine

Les interfaces considérées ici sont les moyens de communication entre l'utilisateur et le programme, en général appelée IHM, pour Interfaces Homme-Machine. Trois approches distinctes peuvent être identifiées dans leur conception [Copper, 1995]. L'approche la plus simple est de calquer l'interface sur le fonctionnement réel du programme. Elle est, de nos jours, peu utilisée car elle demande une grande connaissance de la technique pour pouvoir être efficacement appréhendée. L'approche métaphorique, elle, consiste à proposer une interface ressemblant à des éléments de la vie de tous les jours. C'est d'ailleurs ce qui est en général utilisé pour les interfaces des systèmes d'exploitation modernes, avec les notions de bureau, de documents, de dossiers, etc. La dernière approche, dite idiomatique, consiste à associer aux éléments d'interfaces des comportements simples, qui ne font pas nécessairement référence à des éléments réels, mais sont cohérents, stéréotypés et reproductibles.

Dans l'ensemble, ces interfaces regroupent les moyens mis en œuvre pour afficher de l'information, de l'application vers l'utilisateur, ainsi que ceux permettant le contrôle de l'application, de l'utilisateur vers l'application.

4.1.1 Affichage de l'information

La transmission de l'information à destination de l'utilisateur d'un programme s'appuie sur différents types d'équipements, qualifiés de périphériques de sortie. Le plus commun est bien sûr l'écran, mais il n'est pas le seul : les enceintes, ou les casques audio font aussi partie des équipements standards de communication vers l'utilisateur.

D'autres, en général moins standards, répondent à des besoins spécifiques. On peut citer, par exemple, ceux dédiés spécifiquement à l'accessibilité, tels que les terminaux braille. Il s'agit de dispositifs munis de cellules comportant huit points en relief rétractables. Ils permettent ainsi d'afficher des caractères braille éphémères qui seront remplacés par d'autres au fil de la lecture. Le braille informatique utilisé avec ces terminaux est une version enrichie du braille classique à six points, les deux points supplémentaires permettant notamment de coder l'usage de majuscules ou de minuscules.

Autre type de périphérique spécifique, les imprimantes permettent une sortie d'informations sur papier. Il existe d'ailleurs des imprimantes permettant la sortie de documents tactiles en braille.

Les programmes informatiques actuels utilisent principalement des interfaces graphiques (GUI pour *Graphical User Interface*), bien que les interfaces en ligne de commandes (CLI pour *Command Line Interface*) subsistent néanmoins. Les seconds procèdent à un affichage pas à pas, en général linéaire, de l'information. Les premières, elles, sont organisées selon une structure graphique en deux dimensions, souvent signifiante, qui pose un problème majeur aux malvoyants. En effet, lorsque l'information textuelle est séquentielle, comme c'est le cas en CLI, l'agrandissement, la transcription en braille ou par un synthétiseur vocal permet d'obtenir la même information. Dans le cas où l'agencement des éléments graphiques les uns par rapport aux autres organise les éléments textuels, cette organisation ne transparaît pas nécessairement à la transcription, pouvant occasionner des confusions, mélanges ou pertes d'information. Les mises en page en colonnes, par exemple, permettent d'identifier visuellement des paragraphes regroupés en blocs séparés par des espaces. Lors d'une lecture séquentielle du texte, la première ligne de chaque paragraphe est lue avant de passer à la seconde, rendant impossible la compréhension des informations. Il convient de s'assurer qu'une interface graphique est totalement transposable séquentiellement, ce qui est rarement le cas, ou plus simplement de proposer des interfaces alternatives sonores, ou en texte non structuré graphiquement. Une conception efficace d'applications, de par la séparation entre l'interface et le moteur du programme, permet d'ailleurs assez simplement de proposer ces multiples affichages. L'article de Dominique Burger sur «Les Nouvelles Technologies et l'accès à l'information pour les personnes handicapées visuelles» [Burger, 2001] permet d'approfondir les obstacles liés aux handicaps visuels et les solutions pouvant y remédier.

De la même façon, bien que le problème soit moins fréquent, une application accessible ne doit pas reposer uniquement sur le médium sonore pour la transmission d'informations,

un complément visuel à cette information sonore est indispensable.

Ainsi, l'idée générale de toujours fournir plusieurs moyens distincts et différents d'accéder à une information permet de répondre à un certain nombre de besoins d'une application accessible. Qualifiée d'approche multi-modale, elle s'illustre notamment dans les travaux d'Alexis Sepchat sur des jeux vidéo accessibles avec l'utilisation complémentaire de la modalité tactile [Sepchat et al., 2006] ou sonore [Sepchat et al., 2008d]. Des recherches sur des correspondances modales automatiques existent, comme celle de Mirjana Devetakovic Radojevic et Raewyn Turner [Devetakovic Radojevic and Turner, 2002], qui explorent les relations possibles entre la musique et l'architecture. Ils proposent, ainsi des correspondances entre le monde des sons et celui des images. Néanmoins, une mise en place de la multi-modalité dès la conception reste, encore, une solution plus simple et efficace.

Le problème de la langue est plus simple et couramment résolu. Une application ne doit pas faire usage uniquement de la langue de son auteur. Elle peut s'appuyer sur des pictogrammes ou des traductions, au moins dans une langue vernaculaire, si ce n'est dans l'ensemble des langues des publics présumés.

Comme pour les applications web, les éléments mobiles, éphémères ou clignotants sont à utiliser avec parcimonie et en tenant bien compte qu'ils ne seront vraisemblablement pas totalement perçus par tout le monde.

Les considérations sur la charge cognitive ou mémorielle de l'utilisateur impactent, elles aussi, l'organisation de l'interface. Tout le monde n'est pas à même de comprendre aussi vite et aussi facilement. Idéalement, l'intégralité du contexte nécessaire à la compréhension d'un élément doit être affichée en même temps que lui : dans le cadre de l'accessibilité, il n'est pas judicieux de s'appuyer sur la mémoire de l'utilisateur. Une surcharge d'informations, des difficultés à reconnaître des motifs ou des visages, sont des exemples de difficultés, rencontrées notamment dans le cadre des sites web récents, pour lesquels Peter Fairweather et Shari Trewin [Fairweather and Trewin, 2009] proposent des solutions. Ces solutions, qui s'adaptent d'ailleurs au cas de programmes informatiques quelconques, consistent en une organisation de l'information en catégories, une association systématique de textes descriptifs aux images, etc. Plus généralement, l'adaptation automatique, à l'utilisateur, de l'interface et des possibilités qu'elle offre, constitue une excellente approche. Ainsi l'utilisateur débutant ou en situation de handicap a, face à lui, une interface plus légère et didactique, alors qu'un expert, habitué, se voit proposer une interface plus riche et plus rapide.

4.1.2 Contrôle de l'application

La transmission d'informations dans l'autre sens, permettant à un utilisateur de dialoguer avec le programme, est confrontée à des difficultés parfois différentes. La quantité de matériels disponibles à cette fin est bien plus importante. Regroupés dans la catégorie des périphériques d'entrée, ces équipements vont des plus communs aux plus spécifiques, dédiés à un usage particulier, ou à une catégorie d'utilisateurs, comme dans le cas des périphériques adaptés. Plusieurs classifications peuvent être adoptées pour les regrouper. Dans sa thèse [Dragicevic, 2004], Pierre Dragicevic propose, notamment des différentiations suivant la tâche à laquelle ils sont destinés, les utilisateurs auxquels ils s'adressent, l'environnement

dans lequel ils seront utilisés.

Dans notre cas centré sur l'accessibilité, un regroupement en trois grandes catégories apparaît pertinent :

Les dispositifs de pointage regroupent tous les équipements qui ont un comportement similaire à une souris. Ils permettent de déplacer un pointeur dans une fenêtre graphique et de déclencher un certain nombre d'actions, en général à l'aide de boutons en nombre réduit.

Les claviers sont des équipements munis d'un nombre important de touches souvent de l'ordre de la centaine, chacune étant associée à une action particulière. Ils permettent en général de taper du texte.

Les dispositifs spécifiques, dédiés à des applications particulières, sont trop nombreux pour en faire une liste exhaustive, les plus courants sont ceux associés au jeu, tels que les joystick, joypad, volants, etc., ceux pour les applications audio-video, comme les tablettes graphiques, les webcam et autres dispositifs de capture video, les micros, etc. On peut aussi citer les scanners et lecteurs de codes barres qui permettent l'interaction avec des supports papier. L'interface développée par Luigi Pagliarini et Henrik Hautop Lund [Pagliarini and Hautop Lund, 2002] est à noter ici. Elle est l'une des approches visant à développer un matériel spécifique pour la production de musique à l'aide d'un ordinateur. Elle est constituée d'un gant et du logiciel associé, le tout permettant de faire correspondre des sons aux positions des doigts et du poignet.

Les dispositifs de pointage peuvent prendre de nombreuses formes qui diffèrent sensiblement dans la façon de les utiliser. Cette variété permet à chacun d'adopter le dispositif qui lui convient le mieux. La souris est le plus répandu de ces périphériques, mais elle n'est pas le seul. Ainsi, les trackballs et télécommandes de présentation constituent une approche différente du pointage qui convient mieux à certaines personnes, comme dans le cas de difficultés motrices au niveau du bras. Les cas de handicaps moteurs sévères bénéficient de périphériques spécifiques leur permettant de pointer et cliquer via des mini-joystick manipulables avec la bouche ou la langue, des capteurs de souffle, des boutons contacteurs au pied, des dispositifs de suivi des mouvements de la tête ou de l'œil, etc. L'avènement récent de dispositifs tactiles grand public, désormais courants sur les écrans de nos téléphones portables, ouvre une voie intéressante, souvent plus accessible que les souris classiques, en tout cas plus intuitif. Bien souvent, l'utilisation d'un dispositif de pointage impose un retour visuel. Il faut savoir où se trouve le pointeur, ça n'est, donc, pas toujours adapté pour les non-voyants.

Les claviers existent eux aussi dans des versions adaptées : claviers surdimensionnés, utilisant des pictogrammes, à sensibilité réglable, avec des dispositions adaptées, ou disposant de guide-doigts. Ils peuvent aussi se voir substitués par des claviers virtuels pouvant même être utilisés à l'aide d'un unique bouton, en se basant sur des temporisations, ou plus simplement via un dispositif de pointage. Les techniques de reconnaissance vocale peuvent, dans certains cas, remplacer ou compléter avantageusement un clavier. Des techniques visant à accélérer l'utilisation des claviers virtuels, notamment par un réarrangement de la disposition de leurs touches font encore l'objet de recherches à l'heure actuelle [Raynal, 2005, Colas et al., 2007]

Ainsi, chacun dispose de périphériques d'entrée adaptés à son cas, qui lui permettent

de disposer, sinon d'un clavier et d'une souris, en tout cas de dispositifs équivalents. Néanmoins, cette possible substitution par des équipements accessibles ne résout pas tous les problèmes. L'étude [Donegan et al., 2009] d'un dispositif de pointage par le regard, par Donegan et al., débouche d'ailleurs sur des conclusions similaires. Certains aménagements sont nécessaires lors de la conception des applications qui l'utilisent. En effet, si l'accès au pointage, au clic, ou à l'activation de touches du clavier est assuré par des périphériques adaptés, ces derniers ne permettent pas toujours toutes les possibilités offertes par le dispositif d'origine.

L'usage de commandes multiples, maintenant si répandu, diminue clairement l'accessibilité. Car si la pratique du cliquer-déposer, de l'appui simultané sur deux ou trois touches d'un clavier, ou d'autres combinaisons, n'est pas bien compliquée pour un utilisateur standard, elle présente beaucoup plus de difficultés avec certains périphériques de substitution. La vitesse d'action est un autre élément à prendre en compte car, à l'évidence, tout le monde ne peut pas exécuter les mêmes actions à la même vitesse, surtout dans le cas où l'outil diffère. Une étude sur les performances obtenues à l'aide de divers dispositifs de pointage [Mackenzie et al., 1991], en arrive d'ailleurs à cette même conclusion. L'utilisation de temporisations ajustables permettent d'adapter la vitesse à chaque utilisateur.

La fatigue générée par l'utilisation d'une application peut rendre son usage pénible, parfois même impossible. Les mouvements longs, répétitifs ou le maintien d'une pression sur un bouton demande, dans certains cas, un effort important qu'il faut prendre soin de limiter. L'utilisation combinée de plusieurs périphériques d'entrée est parfois utile, pour exécuter certaines actions complexes dans un même geste. Ce genre d'outil ne doit bien sûr pas être la seule façon d'effectuer ces actions car elles sont impossibles ou très compliquées à effectuer dans certaines configurations.

4.2 Accompagnement et assistance

La prise en compte des problèmes liés aux interfaces, bien qu'indispensable, est insuffisante. Les travaux de Chris Law, et ses collègues sur une évaluation de ressources accessibles [Law et al., 2008a, Law et al., 2008b] insistent, notamment, sur l'intérêt d'une phase de conception rigoureuse et bien menée. Cette phase devant permettre :

- une définition précise des fonctionnalités du logiciel, qui répondent à des besoins clairement identifiés ;
- une intégration des contrôles qualités le plus tôt possible ;
- une hiérarchisation des priorités (accessibilité, coût, délais, respect du cahier des charges, etc.) ;
- une prise en compte des erreurs accompagnant l'utilisateur vers leur résolution ;
- prise de recul vis-à-vis d'utilisateurs pouvant être confrontés à des difficultés différentes des nôtres.

Certaines de ces considérations sont reprises dans le chapitre "Les fourmis artificielles et le handicap : compensation de déficits d'autonomie et jeux pour déficients visuels" du livre de Nicolas Monmarché, Frédéric Guinand et Patrick Siarry [Sepchat et al., 2009].

L'exemple des jeux vidéos accessibles est particulièrement parlant en terme d'adaptation. Une mise en œuvre des solutions aux problèmes d'interfaces que nous avons détaillés

précédemment y est bien sûr nécessaire. Mais ce n'est pas suffisant, comme le montrent les travaux d'Alexis Sepchat [Sepchat et al., 2008b]. Dans le cas de jeux contre une intelligence artificielle, l'adaptation du challenge proposé au joueur est capitale. Le niveau de difficulté doit correspondre au joueur. Un jeu trop difficile, comme un jeu trop facile n'est pas adapté, et dans la pratique n'est d'ailleurs pas utilisé. Pour améliorer l'accessibilité, un système d'adaptation du niveau de jeu à celui du joueur est donc indispensable. Le système proposé dans les articles d'Alexis Sepchat et al. [Sepchat et al., 2007, Sepchat et al., 2008b] permet un équilibrage automatique du niveau de jeu à celui du joueur à l'aide d'un algorithme basé sur le partage des tâches chez les fourmis. Les différentes tâches imposées à la colonie sont les éléments sur lesquels l'intelligence artificielle joue. Le stimulus associé est proportionnel à l'urgence à réaliser la tâche. Les mécanismes d'intelligence collective, spécialement lorsqu'ils permettent aux fourmis d'être inactives, offrent une bonne réactivité et une réponse forte dans le cas d'un joueur efficace. Si le joueur a un impact plus modéré sur le jeu, la colonie s'adapte, et réagit moins violemment. La conclusion de ces travaux est qu'une adaptation automatique du niveau de jeu d'un programme augmente nettement l'accessibilité d'un jeu.

Deuxième partie

Les travaux

Les travaux menés durant cette thèse se sont concrétisés sous la forme de deux logiciels. Il s'agit de logiciels de création artistique, accessibles, utilisant des méthodes génératives. Ils répondent ainsi aux objectifs et aux contraintes associées aux deux axes principaux autour desquels ils s'articulent : l'accessibilité et la création artistique. Ces objectifs et contraintes constituent le cadre général des travaux dont voici une définition.

L'accessibilité se caractérise, comme nous l'avons vu, par un ensemble de contraintes liés à l'IHM du programme, ainsi qu'à l'accompagnement qu'il propose. Du point de vue de l'interface, la transmission d'informations d'un programme vers ses utilisateurs, pour être qualifiée d'accessible, doit se confronter à plusieurs difficultés principales :

- les handicaps sensoriels liés à la vue ou à l'audition ;
- les problèmes liés aux langages utilisés ;
- les problèmes de compréhension et de réactivité.

Les interfaces proposées doivent pouvoir s'adapter au public cible. En cas de déficiences sensorielles, le programme doit proposer des interfaces indépendantes basées sur des modalités différentes. Le logiciel doit s'exprimer en plusieurs langues, ou utiliser un système de pictogramme, qui s'adapte d'ailleurs aux jeunes enfants ou aux personnes ayant des difficultés pour lire. Un soin particulier doit être pris de ne pas surcharger l'interface, pour en faciliter la compréhension, et d'en rendre l'utilisation la plus simple et intuitive possible.

Pour le contrôle de l'application, nous nous appuyons sur les périphériques standards que sont le clavier et la souris. Ils présentent l'avantage d'être très répandus, économiques et de pouvoir être substituer par des périphériques adaptés. Ces substitutions sont à anticiper de façon à fournir une utilisation pratique du programme, lorsqu'elles adviennent. Le programme doit pouvoir être contrôlé avec uniquement l'un ou l'autre de ces périphériques, il ne doit pas être nécessaire de devoir utiliser les deux simultanément. Cela irait à l'encontre de l'idée de multi-modalité qui propose plusieurs moyens de contrôle pour pouvoir choisir le plus adapté à son cas personnel. La conception de ces interfaces de contrôle doit chercher à minimiser la fatigue engendrée par leur utilisation.

L'idée générale est de proposer un logiciel clair, simple, efficace et compréhensible. Il se doit d'assister et d'accompagner l'utilisateur, en résumé, de tout faire pour lui faciliter la vie. L'incorporation d'un maximum de connaissances dans l'outil permet de moins se reposer sur celles de l'utilisateur, et donc de limiter son accès aux seules personnes disposant de ces connaissances. L'aspect didactique est aussi très important, les événements inattendus qui se produisent doivent être expliqués et suivis d'un accompagnement vers une résolution du problème. Puisque nous disposons d'outils de génération de musique ou de dessin, nous proposons de les utiliser, pour fournir des programmes disposant d'outils ayant une part d'autonomie. Ces outils sont une forme d'assistance à l'utilisateur. Ils peuvent ainsi lui permettre de produire des résultats conséquents à partir d'actions réduites.

Dans ce domaine de la conception d'applications, tout est affaire de compromis. Un outil puissant et spécifique est en général difficile à utiliser et nécessite un minimum de formation dédiée à son apprentissage. Il est rare d'avoir les moyens de pouvoir tout faire et de répondre à tous les critères qu'on peut se fixer dans un cas théorique. Le temps, les moyens financiers ou humains, etc. sont autant de limites avec lesquelles il va falloir négocier. Il est donc particulièrement important de garder en tête les besoins auxquels on cherche à subvenir et le public à qui on s'adresse. Dans un cas général, il faudrait, ainsi,

concevoir des interfaces universelles, ce qui n'est pas toujours réaliste. Pour chacun des programmes proposés, nous nous attachons donc à définir le public cible, ce qui permettra de préciser les contraintes à prendre en compte. Ce cadre général a été présenté à la conférence Accessible Design in the Digital World en 2008 [Clair et al., 2008a].

Dans le domaine de la musique, le premier est un programme permettant d'improviser. Il propose un accompagnement automatique en direct de la performance de l'utilisateur.

Le second s'intéresse au dessin. Il permet la création d'images numériques à l'aide d'outils de dessin classiques, que viennent compléter des outils génératifs semi-autonomes.

Chapitre 5

Un instrument de musique virtuel

5.1 Présentation et objectifs

Ces travaux, axés sur la musique, visent à la conception et au développement d'un logiciel offrant à ses utilisateurs une forme de création musicale accessible.

Plus précisément, l'objectif de ce programme est de permettre de créer de la musique, en temps réel. Il se situe donc plutôt dans le champ de l'improvisation que dans celui de la composition. Ainsi, on peut utiliser ce logiciel pour jouer de la musique en direct avec d'autres musiciens. On rencontre habituellement plus d'outils de composition qui permettent, donc, la création de partitions qui seront interprétées plus tard.

Bien sûr, dans le domaine de l'accessibilité, nous essayons de toucher le public le plus large possible. Néanmoins, il convient de préciser quels sont les objectifs prioritaires. Les aveugles et les malvoyants ne semblent pas rencontrer de difficultés majeures à la pratique de la musique. En tout cas, c'est ce qu'on peut penser lorsqu'on regarde la scène musicale actuelle et les nombreux musiciens non-voyants plus qu'efficaces dans ce domaine. L'accessibilité de la musique aux sourds repose sur une étude poussée des gammes de fréquences qu'ils sont à même de percevoir, d'utiliser ces fréquences, ou des correspondances avec des éléments basés sur d'autres sens, tels que des équivalents visuels. Il s'agit de problèmes particulièrement intéressants, mais auxquels nous ne nous intéresserons pas dans ces travaux. Les handicaps sensoriels ne sont donc pas situés au cœur de ces travaux. Le public cible est constitué de personnes en situation de handicap moteur, ayant des difficultés dans l'utilisation d'instruments de musique usuels, ainsi que ceux confrontés à des déficiences cognitives légères. Dans un cadre plus large que celui du handicap, les personnes n'ayant pas accès à une pratique musicale tout simplement parce qu'ils ne connaissent pas la musique, les enfants, et plus généralement, le grand public, font aussi partie des utilisateurs visés.

L'accessibilité du logiciel s'architecture autour de plusieurs points détaillés dans les sections suivantes.

5.1.1 Accessibilité des interfaces

À l'évidence, l'accessibilité d'un logiciel commence par l'accessibilité de ses interfaces. En terme d'affichage, une première approche consiste à rester le plus simple et sobre possible. L'utilisateur ne doit pas être submergé d'informations superflues. L'interface ainsi obtenue pourra être complétée dans un second temps, en fonction de besoins identifiés auprès des utilisateurs tests. À terme, l'idéal étant d'obtenir une interface dont la complexité, les fonctionnalités offertes et le niveau d'aide associé s'adaptent automatiquement à l'utilisateur.

Pour les interfaces de contrôle, nous utiliserons le clavier ou la souris, de façon indépendante. Ces périphériques disposent d'équivalents adaptés aux utilisateurs dans l'incapacité de les utiliser pleinement. Néanmoins, une attention particulière est apportée à offrir une utilisation simple et efficace de ces périphériques, y compris en cas de substitution. Le contrôle du programme ne nécessite pas de mouvement rapide, de combinaison d'actions, de maintien de clic, etc. L'accessibilité recouvre aussi des considérations économiques, aussi l'utilisation de ces périphériques très grand public, donc peu coûteux est particulièrement indiquée.

5.1.2 Pas de pré-requis technique

Le programme doit être capable de se configurer tout seul. Aucun paramétrage n'est demandé à l'utilisateur qui n'a ainsi pas besoin de connaître l'informatique ou l'architecture technique particulière utilisée pour le logiciel. Les messages d'erreur ou situations bloquantes pouvant survenir sont traités automatiquement, et aucune information sur la couche technique sur laquelle s'appuie le logiciel pour fonctionner ne vient interférer avec l'expérience musicale de l'utilisateur.

Bien sûr, il n'est pas non plus requis de savoir musical particulier, comme nous allons le voir au point suivant.

5.1.3 Pas de connaissances musicales requises

Un des enjeux est ainsi de développer un logiciel utilisable sans formation musicale préalable. Le programme doit pouvoir guider l'utilisateur, sans lui demander de connaître la musique. C'est pourquoi, le programme doit disposer de connaissances musicales embarquées.

La notion de tempo est déduite de la musique produite par l'utilisateur. Il ne lui est pas demandé de choisir a priori le tempo auquel il va jouer.

Les considérations harmoniques, pouvant être prises comme des règles musicales, sont liées au contexte culturel et à la pratique personnelle de chacun. L'approche adoptée dans ces travaux consiste à imposer des contraintes fortes, dans un premier temps, quitte à les relâcher par la suite si le résultat obtenu n'est pas assez créatif. Pour conserver une consonance globale à la musique produite, l'instrument virtuel proposé sera diatonique. Il n'utilise que les notes d'une seule gamme. Effectivement, l'éventail de possibilités créatrices est ainsi réduit, mais il n'est pas dans les objectifs premiers de cet outil de servir à des

musiciens chevronnés, plutôt de permettre à des néophytes d'obtenir facilement un résultat satisfaisant.

5.1.4 Accompagnement automatique

Une des idées fondatrices de ce projet est d'évaluer la pertinence de l'utilisation de méthodes génératives. Le logiciel se voit ainsi doté d'un moteur de génération automatique d'accompagnements. L'idée d'accompagner l'utilisateur dans sa propre création plutôt que de proposer un outil de création autonome nous apparaît comme un point crucial de cette approche. Cet outil doit servir à valoriser des intentions créatrices d'un utilisateur, éventuellement handicapé. Il est donc primordial qu'il soit au cœur de la dynamique créative mise en place. Sans lui, rien ne se fait, c'est donc bien lui l'auteur de la musique produite, même s'il est aidé dans cette démarche. Le moteur génératif brode donc un accompagnement à partir des thèmes de l'utilisateur, en interaction avec ce qu'il joue.

5.2 Les solutions apportées

Le logiciel est conçu en suivant l'architecture Modèle - Vue - Contrôleur. Cette méthode permet de développer plusieurs interfaces alternatives aisément, qu'il s'agisse de modalités de diffusion de l'information, comme de contrôleur de l'application, elle est donc particulièrement indiquée dans notre cas.

Le modèle permet d'analyser les commandes envoyées par le clavier et la souris, se modifie en conséquence, et applique les effets résultant sur les éléments qui lui sont associés. Ces éléments, regroupés sur le schéma Fig. 5.1, sont :

- une interface graphique
- un instrument qui va produire les sons
- un ou des accompagnements automatiques

La fenêtre graphique est associée à une série de contrôleurs qui s'assurent du suivi des modifications qu'elle subit. D'autres contrôleurs sont dédiés au suivi des pressions de touches du clavier, des boutons et des mouvements de la souris. Ces éléments sont abordés dans la suite, de même que le concept d'instrument utilisé et le moteur d'accompagnement automatique. Ces deux derniers points reposent sur l'utilisation de la norme midi détaillée tout d'abord.

5.2.1 Utilisation de la norme midi

Le langage Java sert de plateforme technique d'implémentation du programme. Les possibilités qu'il offre, ainsi que les contraintes qu'il impose, ont guidé certains des aspects de cette réalisation. Java fournit un ensemble de classes permettant la manipulation et le traitement de messages et d'évènements midi. Regroupées dans *javax.sound.midi*, ces classes et interfaces offrent notamment les services suivants :

Produire le son

Des synthétiseurs permettant de produire les sons associés aux messages midi qu'ils

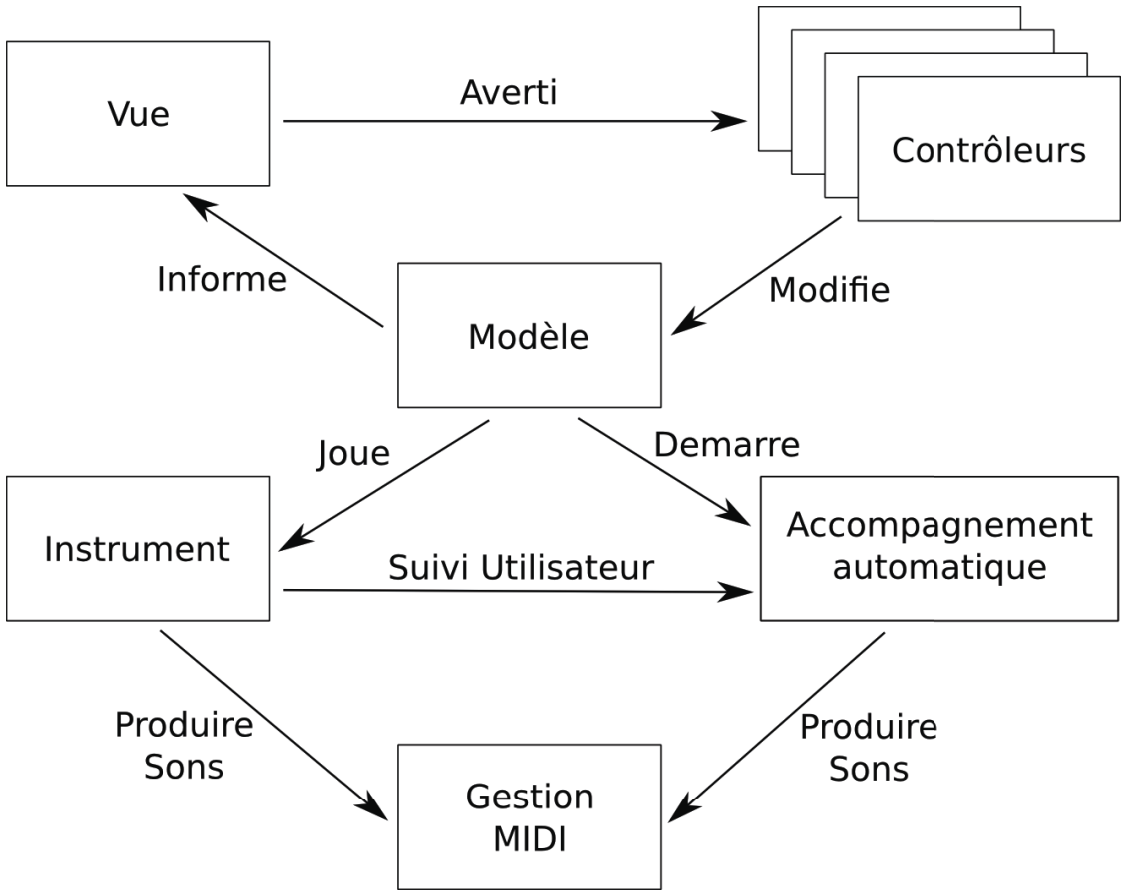


FIG. 5.1 – Schéma des modules du logiciel

reçoivent existent. La fonction java *MidiSystem.getSynthesizer()* permet de récupérer le synthétiseur par défaut disponible sur le système, sous la forme d'un objet *Synthesizer*. Le JRE (Java Runetime Environment) fournit un synthétiseur. Pour des questions de performances, il peut être judicieux de lui substituer une alternative plus efficace. Tritonus est une bibliothèque qui implémente la Java Sound API avec de meilleurs résultats. Il y est notamment possible de profiter d'une carte son via ALSA (Advanced Linux Sound Architecture) et d'obtenir une latence nettement inférieure, et donc plus de réactivité dans le programme.

Les synthétiseurs gèrent des messages midi tels que définis précédemment au chapitre 1.1.1. Java définit la classe *ShortMessage* pour les manipuler. Un synthétiseur peut aussi recevoir des évènements midi. Selon les synthétiseurs, l'information temporelle associée est considérée comme un délai entre le moment de réception de l'évènement et celui de sa restitution. Dans la plupart des cas, elle est simplement négligée et l'évènement est traité immédiatement. Ce dernier comportement est celui du synthétiseur du JRE.

Le timbre, en d'autres termes, l'instrument utilisé pour restituer les notes, est spécifique à chacun des seize canaux midi. Ces canaux, manipulables via des objets MidiChannel sont

configurés par le message particulier *Program Change* (cf. TAB. 1.4) qui impose l'utilisation pour ce canal de l'instrument en paramètre. La référence de cet instrument est appelée *Patch*. Elle est constituée d'un identifiant de banque de sons, suivi de l'identifiant d'un programme spécifique de cette banque. Une référence complète des instruments disponibles en *General MIDI* se trouve en annexe page 122.

L'utilisation de synthétiseurs midi permet ainsi de s'abstraire des problèmes d'enregistrement de son, de traitement du signal, ou autres.

Une approche symbolique de la musique

Comme nous l'avons vu, les hauteurs des notes s'expriment en nombre de demis-tons. Une correspondance directe s'effectue entre les notes du solfège et leur valeur midi. Le tableau TAB. 5.1 référence les valeurs des notes utilisées pour l'instrument.

TAB. 5.1 – Table des notes disponibles

Notes	Numéro midi	Octave	Numéro midi
si♭	58	si♭	70
do	60	do	72
ré	62	ré	74
mi♭	63	mi♭	75
fa	65	fa	77
sol	67	sol	79
la	69	la	81
si♭	70	si♭	82

De la même manière, les durées des notes en midi s'expriment en nombre de ticks. Le tick est l'unité élémentaire de mesure du temps en midi, sa durée dépend du tempo, comme nous allons le voir.

Gérer les contraintes temporelles.

Les séquenceurs sont des entités midi permettant de jouer des séquences complètes d'évènements midi ordonnés dans les temps les uns par rapport aux autres. Le JRE fournit un séquenceur qui, de la même façon que le synthétiseur, peut être remplacé par un autre plus performant tel que celui de Tritonus. Un séquenceur prend la forme d'un objet *Sequencer* auquel on associe un objet *Sequence*. Cette séquence comporte une ou plusieurs pistes, des objets *Track*. Chaque piste est une liste d'évènements midi, ces *MidiEvent* étant des messages midi auxquels a été associée, à chacun, une étiquette temporelle, exprimée en tick. Ces éléments et leur structure sont regroupés sur le schéma FIG. 5.2.

Le tick est donc l'unité temporelle d'un séquenceur. La cadence à laquelle ces ticks sont produits dépend de deux paramètres. Le premier est le tempo qu'on associe au séquenceur. Il peut être exprimé en battements par minute (*i.e.* : noires par minutes), et donné au séquenceur à l'aide de la méthode *SetTempoInBPM(float tempo)*, ou en microsecondes par noire, via *SetTempoInMPQ(float tempo)*. Le second paramètre est la résolution de

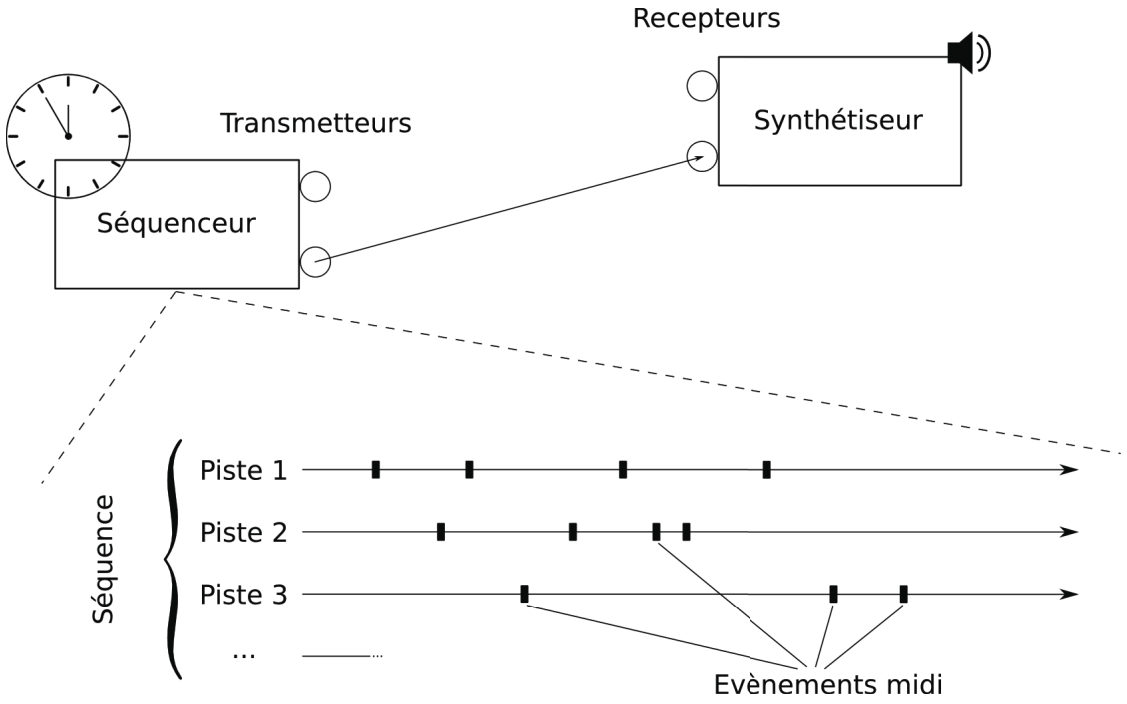


FIG. 5.2 – Schéma du système MIDI

la séquence. Elle s'exprime en tick par noire et doit être spécifiée à la construction de la séquence. Un autre type de résolution existe, plutôt dédié à la synchronisation audio-video, dite SMPTE, exprimée en tick par frame. Elle se décline en 24, 25, 29,97 et 30 frames par seconde. Le programme utilise un tempo en BPM.

La mise en mode lecture d'un séquenceur entraîne alors l'envoi des messages midi prévus à chaque tick à un synthétiseur pour qu'il produise les sons. Ce mécanisme d'association passe par deux objets particuliers : les *Transmitter* et les *Receiver*. Un séquenceur dispose de transmetteurs et de récepteurs, il peut en effet aussi servir à enregistrer les commandes midi qu'il reçoit. Un synthétiseur, lui, se contente de fournir des récepteurs. Ce mécanisme permet de faire transiter des messages midi entre différents composants, qu'ils soient matériels ou logiciels.

L'utilisation d'un séquenceur permet ainsi de s'affranchir de la gestion des contraintes temporelles pour la restitution des sons.

5.2.2 Connaissances musicales embarquées

Le modèle de l'application utilise un objet instrument qui est en charge de communiquer avec les modules midi. Il permet ainsi l'abstraction de la technique de restitution du son au reste de l'application, et surtout d'intégrer de manière transparente des contraintes musicales.

L'instrument est doté d'un paramètre de tessiture qui donne le nombre de notes qu'il est

capable de jouer. On l'utilise ensuite avec des indices de notes propres à l'instrument, qui se charge de la conversion avec les codes midi correspondant. Ces indices s'étendent de 0 à la valeur de tessiture -1. Dans cette version du logiciel, l'instrument proposé est diatonique. Plus précisément, il offre une gamme de si♭ Majeure s'étalant sur deux octaves, soit 15 notes, reprises dans le tableau Tab. 5.1. Cette approche basée sur un objet instrument permet d'étendre facilement les possibilités offertes, par exemple proposer d'autres gammes, plus de notes, passer sur une gamme chromatique, etc.

Les représentations rythmiques des notes jouées sont calculées automatiquement par rapport à un tempo donné. L'instrument fonctionne en interne à un tempo de 60 battements par minute. La conversion est donnée par le rapport 5.1.

$$D_r = \frac{D_m \times T \times R}{60000} \tag{5.1}$$

avec :

D_r : durée en nombre de ticks

D_m : durée en millisecondes

T : tempo

R : résolution du séquenceur

5.2.3 Les interfaces

Le programme dispose d'une interface graphique minimale. L'un des objectifs est d'abstraire la technique. Il n'y a ainsi rien à configurer ou régler. N'ayant donc rien d'important à afficher, dans un premier temps, l'option choisie consiste à n'utiliser qu'une fenêtre vide. On pourrait évidemment afficher des retours graphiques aux interactions avec l'utilisateur, mais à fin de test, ces éléments graphiques pourraient parasiter l'évaluation de la production sonore du logiciel. Il ne s'agit pas d'avoir des utilisateurs satisfaits utilisant l'outil sans le son ! Donc, le seul retour que l'application donne est sonore.

Deux moyens distincts sont fournis pour contrôler l'application. Le clavier est le premier de ceux-ci. Il s'inspire du fonctionnement d'un clavier de type piano. Cette interface relativement intuitive consiste donc à associer à la pression d'une touche le déclenchement d'une note, celle-ci s'arrêtant dès qu'on relâche la touche. La répartition des touches sur le clavier part du constat que le clavier informatique nous offre plus de touches que ce dont on a strictement besoin. On peut donc avoir plusieurs touches pour produire le même son. La disposition de ces répétitions vise à faciliter les intervalles courants. Les touches qui se suivent de gauche à droite permettent de monter la gamme note à note. Au dessus de chaque note se trouve sa tierce dans la gamme. Les trois touches alignées verticalement forment donc une triade classique tonique - tierce - quinte, et offrent une harmonisation en accords de trois sons de la gamme. Le schéma Fig. 5.3 récapitule la disposition des notes sur le clavier.

Le second s'appuie sur un dispositif de pointage. Le postulat de départ est le suivant : il s'agit d'un instrument virtuel. Les interfaces de contrôle peuvent, certes, reproduire le fonctionnement d'instruments réels, mais elles peuvent aussi s'inscrire dans un cadre plus original et explorer les possibilités d'expression d'une interface novatrice. Les contraintes

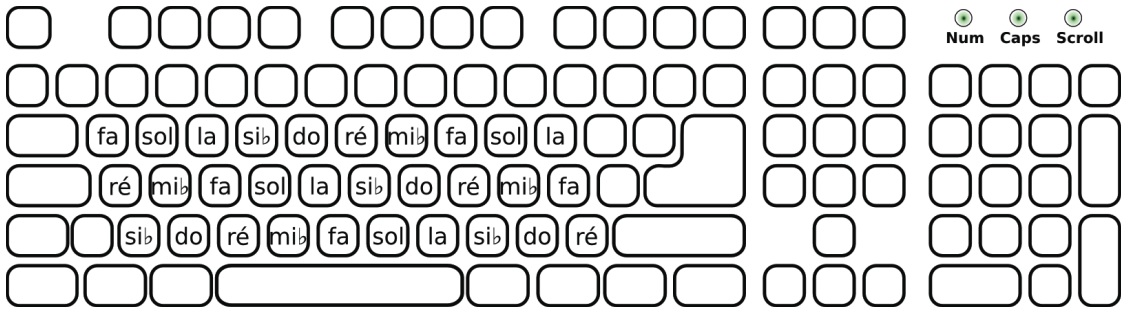


FIG. 5.3 – Schéma de l'utilisation du clavier

adoptées cherchent à limiter la fatigue et assurer un maximum de compatibilité en cas de substitution par un périphérique adapté. Pour ce faire, les boutons ne sont pas utilisés et seul le déplacement du pointeur est pris en compte.

Cette interface se base sur la réalité physique du son, où de grandes vibrations produisent des sons graves, quand des vibrations plus courtes en produisent de plus aigus. Le fonctionnement se base, donc, sur la distance parcourue par le pointeur. On récupère la position d'origine du pointeur sur la fenêtre et lorsqu'il se déplace, on mesure la distance qui le sépare de cette position d'origine. Tant que le pointeur s'éloigne, il ne se passe rien, mais dès que la distance depuis la position d'origine précédemment définie diminue, une note est déclenchée. La hauteur de cette note dépend directement de la distance parcourue : plus cette distance est grande, plus la note est grave. A chaque note jouée, le point de départ du pointeur est redéfini à la position actuelle du pointeur et on commence à préparer la note suivante.

Les notes les plus graves s'obtiennent donc en parcourant la distance la plus longue possible, soit la diagonale de la fenêtre du programme. L'algorithme de ce fonctionnement est décrit par l'algorithme 1.

5.2.4 Accompagnement automatique

Un des objectifs de ces travaux est de fournir un accompagnement automatique de l'utilisateur, plus précisément, de générer, en parallèle de la production de l'utilisateur, des phrases musicales s'accordant avec cette dernière. Les détails de ce système d'accompagnement automatique ont été présentés à la conférence Évolution Artificielle en 2009 [Clair et al., 2009b].

5.2.4.1 Principes de base

Le moteur de cet accompagnement consiste en un algorithme de colonie de fourmis artificielles. Basé sur le comportement de déplacement collectif des fourmis, il utilise le parcours des fourmis comme générateur musical. Il s'inscrit en cela dans la continuité des travaux de Nicolas Monmarché sur l'étude des capacités créatives des algorithmes de fourmis artificielles, cf. [Monmarché and Clair, 2009], par exemple.

Algorithm 1 Algorithme de suivi des déplacements de la souris

distanceMax ← $\sqrt{(\text{tailleFenetreX})^2 + (\text{tailleFenetreY})^2}$
tessiture ← Instrument.recupereTessiture()
origineX ← Position de la souris en X
origineY ← Position de la souris en Y
distance ← 0
loop
  positionX ← Position de la souris en X
  positionY ← Position de la souris en Y
  nouvelleDistance ← $\sqrt{(\text{origineX} - \text{positionX})^2 + (\text{origineY} - \text{positionY})^2}$
  if nouvelleDistance > distance **then**
    distance ← nouvelleDistance
  else
    jouerNote(arrondi($(1 - \frac{\text{distance}}{\text{distanceMax}}) \times$ tessiture))
    origineX ← Position de la souris en X
    origineY ← Position de la souris en Y
    distance ← 0
  end if
end loop

Le principe est d'associer une signification musicale à l'environnement sur lequel les fourmis évoluent. Cet environnement prend la forme d'un graphe pour lequel on associe une note à chaque sommet. Les fourmis se déplacent alors de sommet en sommet en empruntant les arcs du graphe. À son arrivée sur un sommet, une fourmi « joue » la note correspondante.

Le choix de la direction qu'une fourmi emprunte dépend uniquement de l'information locale qu'elle récolte, et s'effectue, bien sûr, sans contrôle centralisé. Cette information, basée sur les phéromones pour les fourmis réelles, est stockée sur les arcs du graphe. Ces phéromones mesurent, en quelque sorte, l'attirance qu'exerce ce chemin sur nos insectes virtuels. Ces phéromones sont déposées par les fourmis chaque fois qu'elles parcourent un arc. La valeur déposée par chaque fourmi lors de son passage est un des paramètres de l'algorithme, ces paramètres sont récapitulés dans le tableau Tab. 5.2. L'incrémentation de la quantité de phéromones est systématique, chaque fourmi augmente cette valeur à chacun de ses passages sur un arc.

Néanmoins, le choix d'une direction à prendre conserve une part de hasard, capitale dans ce genre de méthode. Le rapport entre la quantité de phéromones présente sur un arc et la quantité présente sur tous les arcs environnant donne la probabilité qu'une fourmi choisisse cet arc. On peut la calculer à l'aide de l'équation 5.2.

$$P(x) = \frac{S(x)}{\sum_{i=1}^{n}(S(i))} \tag{5.2}$$

Avec :

x : un arc partant du sommet courant, avec $x \in [1..n]$

n : le nombre d'arcs partant du sommet courant

$P(x)$: la probabilité de choisir l'arc x

$S(i)$: la valeur (intensité des phéromones) de l'arc i

On peut noter que chaque fourmi choisit toujours une direction à chaque pas de temps. Elles sont contraintes de bouger. Aucune notion de distance n'apparaît dans cet algorithme, aussi les déplacements des fourmis sont-ils synchrones. Toutes les fourmis parcourent un arc à chaque pas de temps.

L'émergence d'une intelligence collective apparaît, dans le cadre d'une colonie de fourmis, par les interactions indirectes de plusieurs individus. Aussi, le nombre de fourmis présentes est un autre paramètre important de cette méthode. Dans le fonctionnement décrit précédemment, la multiplication des agents fourmis pose un problème. Plus il y a de fourmis, et plus il y a de notes jouées simultanément. Le risque de surcharge sonore, conduisant à produire un bruit désagréable, augmente très vite. Pour limiter ce phénomène, les fourmis sont divisées en deux catégories. La première sont les fourmis, dites musiciennes, qui jouent effectivement des notes. Leur nombre caractérise directement la capacité polyphonique de l'accompagnement automatique. Dans le cas d'une seule fourmi, l'accompagnement ne peut ainsi jouer qu'un son à la fois. La seconde catégorie est constituée de fourmis silencieuses. Elles agissent exactement comme les autres sauf qu'elles ne jouent jamais de note. Leur apport consiste uniquement à favoriser l'émergence de l'intelligence collective. Ce sont finalement ces deux paramètres qui figurent sur le tableau récapitulatif Tab. 5.2, leur somme étant le nombre total de fourmis de l'algorithme.

Un mécanisme d'évaporation vient compléter cet algorithme de fourmis. Il consiste à faire progressivement diminuer la quantité de phéromones présente sur chaque arc au fil du temps. Ce mécanisme permet à la colonie une meilleure adaptabilité aux changements de l'environnement, et participe donc grandement à la robustesse de ces méthodes. À chaque pas de temps, un certain pourcentage de la valeur stockée sur chaque arc disparaît. Ce coefficient d'évaporation vient donc compléter notre tableau Tab. 5.2 des paramètres de l'algorithme.

5.2.4.2 Interactivité

La mise en place d'interactions entre l'algorithme et l'utilisateur est fondée sur deux mécanismes.

Le graphe de notes sur lequel les fourmis évoluent est construit par l'utilisateur. En fait, nous n'utilisons pas un, mais deux graphes. Un graphe dont les sommets sont les hauteurs des notes, l'autre contenant les durées des notes. Le programme est muni d'un système de suivi de la musique jouée par l'utilisateur. À l'initialisation, les graphes sont vides, l'accompagnement automatique ne peut donc rien faire seul. Au cours du jeu de l'utilisateur, le programme récolte les notes jouées, et les ajoute aux deux graphes. Les fourmis sont placées simultanément sur les deux graphes. Lorsque les graphes disposent d'au moins un sommet et que les fourmis ne sont pas encore placées, elles se voient chacune affectée un sommet de chaque graphe au hasard. Les fourmis musiciennes fabriquent ainsi leur note en récoltant l'information harmonique du premier graphe et l'information rythmique du second. C'est l'association de cette hauteur et de cette durée qui déclenche une note. Le moteur d'accompagnement automatique se base donc uniquement sur les notes jouées par

l'utilisateur. Il n'est capable que de variations autour des thèmes du musicien qui l'utilise. Cette séparation en deux graphes permet une amélioration des performances que nous aborderons plus tard, mais permet aussi, d'augmenter la créativité. En effet, les fourmis ont tendance à reproduire les schémas harmoniques et les schémas rythmiques joués par l'utilisateur, mais ont la capacité de les associer différemment.

Les deux graphes sont orientés, pour leur donner une signification plus musicale. En effet, les arcs de ces graphes modélisent implicitement les intervalles entre les notes. Or, en musique, un intervalle ascendant, tel que do – mi, n'est pas équivalent à un intervalle descendant, dans notre exemple mi – do. En effet, le premier est une tierce, l'autre conduit à la sixte inférieure. L'homme est d'ailleurs beaucoup plus sensible aux intervalles qu'aux notes elles-mêmes. Une expérience simple permet de s'en rendre compte. Il est aisé de reconnaître une mélodie, quelle que soit la hauteur à laquelle elle est jouée, alors que reconnaître d'oreille une note seule demande beaucoup d'entraînement. Plus que des ajouts de sommets aux graphes, ce sont plutôt des arcs orientés qui sont ajoutés, les sommets n'étant rajoutés que lorsqu'un arc pointe vers un sommet inexistant. L'initialisation de la quantité de phéromones présente sur un arc est un paramètre important de l'algorithme, puisqu'un arc de valeur zéro ne sera tout simplement jamais emprunté.

Le second mécanisme consiste à doter le musicien de la capacité de guider les fourmis. Laissées à elles-mêmes, les fourmis ont tendance à créer au fil du temps un chemin préférentiel qu'elles finissent par emprunter très majoritairement. Ce phénomène est décrit plus en détail dans la partie 5.3. Pour permettre à l'utilisateur de l'instrument d'influer sur le comportement des fourmis, il dépose lui aussi des phéromones sur les graphes pendant son jeu. La quantité qu'il dépose est une mesure de son impact sur le comportement de la colonie. Cette valeur s'ajoute donc aux paramètres du tableau Tab. 5.2. L'algorithme complet de suivi de l'utilisateur, permettant de construire les graphes et de déposer des phéromones est détaillé dans l'algorithme 2.

Tab. 5.2 – Paramètres de l'algorithme

Paramètres	Notation
Nombre de fourmis	N_m
Nombre de fourmis silencieuses	N_s
Dépôt standard	τ_f
Dépôt de l'utilisateur	τ_u
Quantité initiale	τ_i
Coefficient d'évaporation	ρ

5.2.4.3 Contraintes du temps réel

L'objectif de ce projet est de permettre de l'improvisation musicale, ce qui signifie donc, de jouer de la musique en temps réel. Cette approche impose des contraintes supplémentaires que ne rencontrent pas les outils de composition. La notion de latence, déjà très rapidement abordée, et notamment de son maintien à une très faible valeur est de première

Algorithm 2 Algorithme de suivi de l'utilisateur

```
/* Initialisation */
sommetCourant(hauteur) ← null
sommetCourant(durée) ← null
grapheHauteur ← null
grapheDurée ← null
for Chaque note jouée do
   On récupère la note jouée,
   correspondant aux nouveaux sommets dans chaque graphe
   for Chaque graphe do
      if sommetCourant n'est pas dans le graphe then
         graphe.ajoutSommet(nouveauSommet)
      else
         arcCourant ← arc(sommetCourant, nouveauSommet)
         if arcCourant ∃ then
            arcCourant.valeur ← arcCourant.valeur + depotUtilistateur
         else if nouveauSommet ∃ then
            arcCourant ← graphe.ajoutArc(sommetCourant, nouveauSommet)
            arcCourant.valeur ← arcCourant.valeur + depotUtilistateur
         else
            graphe.ajoutSommet(nouveauSommet)
            arcCourant ← graphe.ajoutArc(sommetCourant, nouveauSommet)
            arcCourant.valeur ← arcCourant.valeur + depotUtilistateur
         end if
         sommetCourant ← nouveauSommet
      end if
   end for
end for
```

importance dans ce cadre. Elle correspond à la durée entre l'action déclenchant un évènement, ici une note, et l'apparition de cet évènement. En musique, on considère qu'une latence de l'ordre de la dizaine de millisecondes est correcte. Lorsqu'elle dépasse 100ms, la gène occasionnée empêche de jouer.

Pour respecter la politique d'accessibilité du logiciel, et notamment la possibilité pour le programme de s'exécuter sur une machine standard, nous faisons le choix de ne pas utiliser de système garantissant des contraintes temps réel. En l'absence de système d'exploitation véritablement temps réel sur lequel nous appuyer, le programme doit donc faire de son mieux pour s'exécuter le plus rapidement possible.

Un des atouts des algorithmes de fourmis est leur rapidité d'exécution. Ils sont donc particulièrement indiqués ici. Il convient néanmoins de s'assurer que celui utilisé pour notre accompagnement automatique reste rapide. C'est la deuxième raison de l'utilisation de deux graphes au lieu d'un seul : limiter leur taille. Le logiciel permet de jouer sur deux octaves d'une gamme diatonique, soit 15 hauteurs de notes possibles. Le graphe de notes fait une taille maximum de 15 sommets et 225 arcs. Le graphe des durées peut être un peu plus

grand. Les durées sont mesurées en nombre de ticks. La résolution fixée dans le logiciel est de 24 ticks par noire. Cette valeur est calculée pour permette de jouer des triples-croches, d'une valeur d'un huitième de noire, ainsi que des triolets d'une valeur d'un tiers de noire. En limitant la durée maximale des notes à celle d'une ronde, soit quatre noires, on obtient déjà un graphe maximal de 96 sommets et jusqu'à 9216 arcs. Si les graphes n'étaient pas séparés, on pourrait avoir à manipuler 1440 sommets reliés par plus de 2 millions d'arcs, et ce malgré les contraintes fortes appliquées aux possibilités musicales de notre instrument. Dans un cadre plus général, le tableau Tab. 5.3 donne une idée des tailles de graphes pour quelques exemples.

Tab. 5.3 – Taille des graphes

Hauteurs	Durée	Nombre de sommets		Nombre d'arcs	
		1 graphe	2 graphes	1 graphe	2 graphes
1 octave diatonique (8)	5	40	13	1600	89
2 octaves diatoniques (15)	96	1440	111	2 073 600	9 441
2 octaves chromatiques (23)	96	2208	119	4 875 264	9 745
Maximum en midi (128)	128	16 384	256	268 435 456	32 768

Le moteur génératif est exécuté en parallèle de l'instrument, il n'est bien sûr pas question de jouer chacun à son tour. Cette utilisation de *threads* crée une difficulté supplémentaire. En effet, l'accompagnement automatique construit, à sa vitesse, des notes en se basant sur les graphes créés par l'utilisateur. Celui-ci enrichit le graphe au fur et à mesure qu'il joue. Il se trouve donc deux problèmes envisageables. En fait, on s'appuie sur le séquenceur pour l'accompagnement automatique, ainsi chaque note générée prend la forme de deux évènements midi. Le premier, débutant la note, ayant pour date, la date de fin de l'évènement précédent. Le second évènement, de fin de note, prend pour date la somme de la date du premier évènement et de la durée de la note. Soit, l'accompagnement automatique prend du retard et ne génère pas les notes assez rapidement par rapport à la vitesse de lecture. Soit, il génère des notes trop vite et n'est plus en accord avec l'état actuel du graphe.

Pour le premier problème, la solution est effectivement déjà traitée, il faut être rapide. Le logiciel a néanmoins été muni d'un mécanisme de recalage, en cas de retard. Ainsi, si l'accompagnement automatique n'est pas assez véloce, les notes générées sont associées à une date antérieure à la position actuelle de lecture du séquenceur. Dans ces cas-là, la date de début de la prochaine note générée est calée sur la position de lecture actuelle du séquenceur.

Le second problème est rencontré beaucoup plus fréquemment. Il apparaît justement lorsque le moteur génératif prend trop d'avance. En effet, sans mécanisme de régulation, l'accompagnement automatique est capable de produire plusieurs minutes d'accompagnement dès les premières secondes de jeu. Aussi, n'est-il plus du tout réactif par rapport aux changements que lui impose l'utilisateur. Il est donc muni d'un paramètre d'avance maximum autorisé, illustré par le schéma Fig. 5.4. Si la génération automatique dépasse

cette avance, le moteur est mis en sommeil pour un moment, afin de l'empêcher de trop pré-calculer.

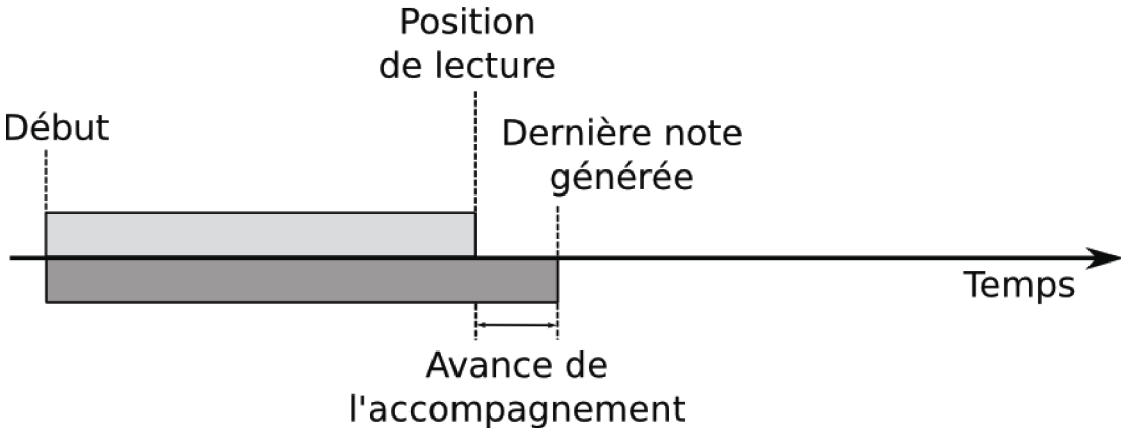


FIG. 5.4 – Schéma du mécanisme d'avance accompagnement-lecture

Concrètement, l'accompagnement automatique proposé joue une basse. Elle est donc monophonique, il n'y a qu'une seule fourmi musicienne. Les notes jouées par l'accompagnement sont transposées de deux octaves vers le bas. L'instrument de musique virtuel a fait l'objet d'une présentation à la conférence Generative Art en 2008 [Clair et al., 2008b].

5.3 Évaluation et résultats

5.3.1 Influence des paramètres

Le comportement général de l'algorithme est de créer des chemins cycliques stables dans le graphe. Il se traduit par des boucles musicales qui se créent naturellement au cours du déroulement de l'accompagnement automatique. Ce résultat est évident à l'écoute et diffère clairement d'un algorithme purement aléatoire. Il est d'ailleurs possible, pour s'en convaincre d'utiliser l'algorithme de fourmis pour obtenir un résultat totalement aléatoire, en fixant les dépôts phéromonaux ainsi que l'évaporation à zéro.

Le rapport entre la quantité de phéromones déposée par les fourmis et la quantité fixée à l'initialisation du graphe, $\frac{\tau_f}{\tau_i}$, permet de faire varier l'influence de l'intelligence collective sur le comportement global. Plus ce rapport est petit, plus l'algorithme aura un comportement aléatoire. Cette organisation émergente est la raison principale de l'utilisation d'un algorithme de ce type dans le cadre de ce projet. La rapidité d'apparition de ces cycles et leur stabilité dans le temps dépendent des paramètres de l'algorithme.

Procédons maintenant à une évaluation de l'influence de ces paramètres sur le comportement du moteur génératif. Dans cette analyse, seules les hauteurs des notes sont prises en compte. Il est en effet beaucoup plus facile de reproduire une suite de notes de même hauteur, plutôt qu'une même figure rythmique. Les effets observés sur le graphe de hauteur peuvent s'extrapoler sur celui de durées.

5.3.1.1 Le nombre de fourmis

L'impact du nombre de fourmis sur le système est évalué avec les paramètres repris au tableau Tab. 5.4. Pour annuler l'influence de l'utilisateur, la quantité de phéromones qu'il dépose est à zéro. L'expérience consiste à jouer, à chaque fois, une même suite de notes, en l'occurrence la séquence fournie par la figure Fig. 5.5. L'utilisation d'une séquence prédéfinie permet de construire un graphe donné. On mesure la fréquence de chaque note dans la mélodie obtenue. Les résultats du tableau Tab. 5.5 sont obtenus sur plusieurs expériences ayant permis la génération de quelques milliers de notes.

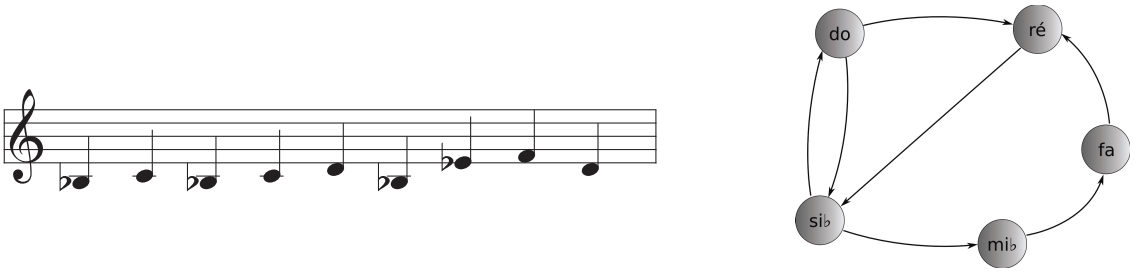


Fig. 5.5 – Suite expérimentale et graphe résultant

Tab. 5.4 – Influence du nombre de fourmis : les paramètres utilisés.

Paramètre	Valeur
N_m	1
N_s	1 - 9 - 99 - 999
τ_f	1
τ_u	0
ρ	0.1

Tab. 5.5 – Influence du nombre de fourmis : proportion de chaque note dans la mélodie générée.

$N_m + N_s$	2	10	100	1000
si♭	48.81%	45.92%	48.61%	50%
do	35.71%	45.92%	51.39%	50%
ré	4.76%	4.08%	0%	0%
mi♭	9.52%	1.02%	0%	0%
fa	1.19%	3.06%	0%	0%

On observe qu'avec 100 fourmis ou plus, une boucle entre si♭ et do est immédiatement créée et qu'elle ne varie jamais. En effet, plus les fourmis sont nombreuses, plus le comportement résultant est rapidement cyclique et stable. Ce résultat est conforme à l'idée que l'organisation émergente est dépendante du nombre de fourmis. Dans le cas de notre instrument, il n'est pas recommandé d'utiliser un trop grand nombre de fourmis, pour ne

pas dégrader les performances, mais aussi pour ne pas avoir systématiquement un résultat trop prévisible. Dans la pratique, ce nombre est fixé à dix, pour permettre le mécanisme d'intelligence collective sans surcharger la machine et en conservant une part d'imprévu.

5.3.1.2 Le coefficient d'évaporation

L'influence de l'évaporation des phéromones sur le comportement global de la colonie est évaluée à l'aide des paramètres du tableau Tab. 5.6. Là aussi, l'influence phéromonale de l'utilisateur est nulle, et les conditions de l'expérience sont identiques à l'expérience précédente. Les mesures du tableau Tab. 5.7 correspondent là encore aux fréquences d'apparition de chaque note dans la mélodie générée.

Tab. 5.6 – Influence du coefficient d'évaporation ρ : les paramètres utilisés.

Parameter	Value
N_m	1
N_s	9
τ_f	1
τ_u	0
ρ	0 - 0.25 - 0,5 - 0,75 - 0,9

Tab. 5.7 – Influence du coefficient d'évaporation ρ : proportion de chaque note dans la mélodie générée.

ρ	0	0,25	0,5	0,75	0,9
si♭	50%	46.81%	48.10%	36.36%	40.54%
do	50%	48.94%	46.84%	28.57%	33.78%
ré	0%	4.26%	2.53%	15.58%	18.92%
mi♭	0%	0%	2.53%	7.79%	0%
fa	0%	0%	0%	11.69%	6.76%

L'évaporation des phéromones a un impact important sur l'organisation produite par la colonie. Plus la valeur du coefficient d'évaporation est élevée, plus les résultats obtenus sont dispersés. L'organisation globale est plus stable dans le temps avec une évaporation faible. De la même manière que pour le nombre de fourmis, pour permettre d'obtenir cette création de boucle mélodique, sans pour autant la rendre trop immuable, le coefficient d'évaporation utilisé pour le logiciel est $\rho = 0,25$.

5.3.2 Efficacité des interactions

L'impact du jeu de l'utilisateur sur le moteur génératif est évalué par l'expérience suivante : une première séquence de notes est jouée, suivie d'une pause de quelques secondes permettant à l'accompagnement de se stabiliser. Cette séquence très simple, si♭ – do – si♭, induit une boucle dans la mélodie générée. Une seconde séquence, si♭ – ré – si♭ – ré, vient

tenter de perturber le cycle. Cette expérience est répétée de nombreuses fois, en utilisant différentes valeurs de τ_u. Les paramètres utilisés sont récapitulés dans les tableau TAB. 5.8. Les résultats obtenus, repris sur le tableau TAB. 5.9, mesurent le pourcentage d'expériences au cours desquelles l'accompagnement est sorti de sa boucle initiale.

TAB. 5.8 – Influence de τ_u, interaction moteur-utilisateur : les paramètres utilisés.

Parameter	Value
N_m	1
N_s	9
τ_f	1
τ_u	1 - 5 - 10 - 20
ρ	0,25

TAB. 5.9 – Influence de τ_u, interaction moteur-utilisateur : proportion d'expériences avec une influence notable de l'utilistateur.

τ_u	1	5	10	20
Influence	40%	60%	80%	100%
Pas d'influence notable	60%	40%	20%	0%

Ainsi, permettre à l'utilisateur de déposer lui aussi des phéromones sur le graphe a un impact majeur sur le comportement de l'algorithme. En fait, plus que la quantité déposée, τ_u, en elle-même, c'est le rapport entre celle-ci et celle déposée par les fourmis, $\frac{\tau_u}{N \times \tau_f}$, qui caractérise l'influence de l'utilisateur. Des expériences avec des ratio similaires pour des quantités différentes produisent, d'ailleurs, les mêmes résultats.

5.3.3 Évaluation de terrain

L'outil a pour objectif de jouer de la musique. Une évaluation objective des résultats obtenus est, au moins périlleuse, pour ne pas dire impossible. L'évaluation de la pertinence de l'outil passe donc par des enquêtes subjectives auprès de véritables utilisateurs.

Le public cible dans le cadre de cette évaluation peut être regroupé en plusieurs grandes catégories :

- personnes handicapés ;
- thérapeutes ;
- enfants ;
- musiciens ;
- grand public.

Les rencontres avec ces utilisateurs tests sont de deux types. Les premières sont des entretiens, individuels, d'une durée d'une à deux heures suivant les cas. Ces entretiens concernent des personnes avec qui un rendez-vous a été fixé auparavant, et qui ont donc déjà consenti à procéder à un test de l'outil. Ils débutent par une présentation générale de l'instrument de musique virtuel comportant :

– une présentation de l'équipe de recherche ;
– le contexte dans lequel le programme a été conçu ;
– ses objectifs généraux ;
– les principes de base du fonctionnement de l'outil.

À la suite de cette présentation, une première proposition est faite de procéder à un essai. Dans certain cas, une demande de renseignements complémentaires amène à aborder le fonctionnement interne de l'accompagnement automatique, c'est-à-dire l'algorithme de colonie de fourmis, ainsi que le fonctionnement des deux interfaces de contrôles.

Le second type de test s'effectue lors de manifestations ouvertes au public et prend la forme de présentations sur un stand. Le temps disponible étant moindre, la présentation et les discussions qui s'ensuivent sont donc beaucoup plus succinctes. Le programme a, ainsi, été présenté lors de la Fête de la science 2009, des Journées portes ouvertes 2009 de l'université François Rabelais de Tours, du festival Univercités 2010 organisé par l'association les Petits débrouillards de Région Centre. Tourné vers un public plus spécifique, le Challenge Handicap et Technologie[1] qui a eu lieu à l'université de Metz les 15 et 16 mai 2009, a aussi servi de terrain d'expérimentation. On a d'ailleurs pu y découvrir que, même s'il n'était pas prévu à cet effet, le programme était tout à fait utilisable par une personne non-voyante, principalement via l'utilisation du clavier, la souris étant plus délicate à utiliser en l'absence du retour visuel de sa position sur l'écran. Le challenge regroupait 24 projets, l'instrument de musique virtuel accessible y a remporté le premier prix « Défi Loisirs ».

Les conférences scientifiques ayant permis des communications sur le projet ont souvent été, elles aussi, des lieux de tests. La rencontre avec une personne handicapée d'un bras a permis de constater la possibilité d'utiliser l'outil à l'aide d'une seule main. Il s'agit d'ailleurs d'un point d'intérêt fort du logiciel car les instruments de musiques habituels utilisables avec une seule main sont très rares, aussi les personnes ne pouvant utiliser pleinement leurs deux mains y trouvent une façon aisée de pratiquer la musique.

Les essais du logiciel se font systématiquement à l'aide d'un casque audio. En effet, le test ne se fait jamais tout seul, puisque il se trouve toujours, au moins, l'expérimentateur à côté de l'utilisateur. L'idée est d'évaluer si la musique jouée par l'utilisateur lui plaît. Il n'est donc pas souhaitable qu'interviennent des rapports sociaux implicites. Les interrogations d'un utilisateur, sur le fait que ce qu'il joue plaît aux autres auditeurs en plus de lui-même, risquent d'interférer avec le rapport personnel de ce testeur avec l'outil. C'est d'ailleurs pour cette même raison que nous avons conservé une version du programme qui affiche uniquement le cadre dans lequel la souris bouge, afin que les résultats graphiques des actions de l'utilisateur n'empiètent pas sur son expérience sonore.

L'observation de testeur au cours de la phase d'essai constitue une importante partie de l'évaluation de l'outil. Poursuit-il son essai dans le temps ? Quelle attitude a-t-il ? Quelles sont les expressions de son visage ? Ce sont des questions de ce genre qui nous intéressent à ce moment. Cette phase est, en général, suivie d'une phase de questions informelles, si possible, en invitant l'expérimentateur à faire des retours sur l'expérience, sa satisfaction globale, les points qui lui semblent à améliorer, et les usages possibles du programme.

Ces observations subjectives et les échanges qui suivent nous permettent de distinguer

[1] http ://challenge-ht.univ-metz.fr/

trois catégories d'utilisateurs. Les premiers sont ceux qui ne sont pas immédiatement intéressés et qui n'ont pas la curiosité de pousser l'expérience plus loin. Leur essai est très court, de l'ordre de quelques secondes, et ils ne sont manifestement pas convaincus par l'outil. Parmi les raisons de cet échec, on trouve sans doute le fait que le résultat qu'ils obtiennent ne correspond pas à l'idée qu'ils se sont faite a priori, ces personnes ayant souvent voulu savoir de quoi il s'agissait précisément avant d'essayer. La seconde partie des testeurs, environ deux tiers d'entre eux, est celle des personnes qui, intéressées par l'outil, poussent leur expérience plus longuement. En général, ils ne cèdent leur place que parce que d'autres utilisateurs attendent, et parfois même, demandent à récupérer le programme. Lors des rencontres avec un jeune public, la proportion de ces utilisateurs intéressés devient très majoritaire, de l'ordre de 90%. En effet, les enfants sont plus prompts à expérimenter et trouvent immédiatement un côté ludique dans l'utilisation du logiciel. La troisième partie des utilisateurs englobe ceux qui restent indécis. Nous considérons qu'ils ne sont pas pleinement satisfaits par l'outil et les assimilons à la catégorie des personnes déçues. Les résultats obtenus sont regroupé dans le tableau Tab. 5.10.

Tab. 5.10 – Résultats des tests de terrain

	Grand public	Enfants	Musiciens	Thérapeutes	Personnes handicapées	Total
Entretiens	0	10	10	6	3	29
Stand	100	150	20	2	10	282
Apparemment satisfaits	65%	89,4%	46,6%	100%	76,9%	77,2%

Ces chiffres sont complétés par quelques mots sur certains des entretiens que nous avons menés. Nous sommes ainsi allé à la rencontre de thérapeutes pour recueillir leurs sentiments sur ce projet. Dans l'ensemble très enthousiastes, ces personnes m'ont accordé souvent plusieurs heures de leur temps. Le docteur Pascal Lenoir, Médecin chef du centre de ressources sur l'autisme de Région Centre, par ailleurs musicien, voit dans une utilisation tranquille de l'outil un côté relaxant. Sans lui avoir précisément décrit le comportement du moteur génératif, il a naturellement eu l'impression d'être « suivi par une ombre musicale ».

Gaël Bouket, un musicothérapeute avec qui j'ai longuement discuté s'est montré particulièrement intéressé. Dans sa pratique, il organise des séances de musique collective avec des patients en situation de handicap mental. Certains de ces patients, qui présentent des troubles de l'apprentissage, ont de grandes difficultés à participer à ces séances. Seuls des instruments très simples, type percussions, sont à leur portée et ils ont rapidement tendance à faire plus de bruits désordonnés que de réelle musique, ce qui nuit à la communication de l'ensemble du groupe. Les contraintes intégrées au logiciel permettent à ce type d'utilisateurs, même lorsqu'ils font des mouvements désordonnés, de conserver un résultat « musical », et donc de rester intégrés au groupe.

Le 9 février 2010, une présentation à l'équipe d'encadrement du centre d'accueil spécialisé de Ballan Miré (37) a permis de déployer le logiciel sur leur parc d'ordinateurs et le mettre ainsi à la disposition de leur vingtaine de résidents. Parmi les constats de cette rencontre, le manque d'information du personnel sur les moyens d'adaptation au handicap

d'ordinateurs standards est flagrant. Néanmoins, la connaissance du public handicapé de l'équipe a permis de proposer des améliorations du programme.

Les résultats de cette évaluation ont été présentés lors du congrès Handicap 2010 [Clair et al., 2010]. D'une manière générale, les conclusions en sont les suivantes. Les objectifs de ces travaux récoltent l'adhésion quasi générale. Le logiciel en lui-même, sans être complètement consensuel, suscite majoritairement l'intérêt de ceux qui l'essaient. La principale raison d'utiliser le programme semble tenir à son coté ludique. La curiosité et l'étonnement laissent rapidement place à l'amusement sur le visage des gens qui tentent l'expérience. D'autres usages émergent de cette campagne de test. L'utilisation de l'outil paraît opportune dans le cadre de séance de relaxation. Un maniement lent de l'instrument tend à produire des sons calmes et reposants. Il s'adapte, aussi, aux personnes ayant des troubles de l'apprentissage, qui peuvent ainsi accéder à une création musicale particulièrement chaotique par ailleurs. Ces rencontres ont aussi permis de dégager un ensemble d'améliorations ou d'évolutions possibles du logiciel. Parmi ces amélioration, certaines sont déjà intégrées, tels que le démarrage automatique de l'accompagnement lorsque l'utilisateur se met à jouer, ou la possibilité d'utiliser simultanément le clavier et la souris.

5.3.4 Perspectives et évolutions

D'autres améliorations restent encore à développer. Parmi les évolutions possibles, la plus évidente est l'ajout d'un retour graphique aux actions de l'utilisateur. Il peut prendre de nombreuses formes (affichages des graphes et des fourmis, tracé des mouvements de la souris, affichage de la partition, etc.), ayant chacune des objectifs et un intérêt différent.

Citons aussi celles qui ont trait aux capacités musicales de l'outil. À l'heure actuelle, l'instrument utilise toujours le même timbre. Il joue un son de violon car c'est un son qui peut durer aussi longtemps que nécessaire. La possibilité de choisir le patch midi peut être ajouté pour permettre à l'utilisateur de choisir son son. Dans l'éventualité d'augmenter ou de diversifier les possibilités musicales, il est envisageable d'offrir plusieurs types de gammes (mineure, pentatonique, chromatique, etc.) et un moyen de choisir parmi elles. De la même façon, le clavier peut proposer des touches pour produire directement des accords. Ce genre d'évolutions vient rapidement complexifier l'interface et nuire à la simplicité d'utilisation, aussi ouvrent-elles la porte à une interface adaptative qui se personnalise automatiquement en fonction des besoins de l'utilisateur.

En se basant sur une conception modulaire de l'interface, regroupant des fonctions connexes au sein d'éléments d'interfaces pouvant être affichés ou masqués, une interface qui s'enrichit progressivement lorsque l'utilisateur s'habitue au logiciel constitue un exemple d'interface adaptative. L'utilisateur débute alors avec une interface minimale, qui offre de plus en plus d'options avec le temps. L'avis de l'utilisateur sur l'interface peut être collecté, en récupérant la façon dont il dispose les différents éléments, et en dotant ces éléments d'un moyen pour l'utilisateur de les valider ou de les désapprouver. Un algorithme peut alors choisir, à partir de ces informations, quels autres éléments proposer dans l'interface.

D'autres évolutions sont envisageables au niveau du moteur du logiciel. Une détection automatique du tempo permet une adaptation plus fine au jeu de l'utilisateur. L'accompagnement automatique peut être enrichi, en lui ajoutant d'autres instruments, ou en testant

d'autres méthodes génératives. L'algorithme de fourmis artificielles déjà développé peut incorporer plus de règles musicales, par exemple par l'utilisation de seuils. Cette forme d'hybridation avec les mécanismes de partage de tâches chez les fourmis permet d'inclure un mécanisme de préférence indépendant des phéromones, et pouvant favoriser certains intervalles ou certaines durées, comme par exemple une tendance à compléter exactement une mesure.

Ce ne sont là que quelques pistes de développement d'un outil déjà prometteur qui ouvrent la voie à plus d'adaptations et de personnalisation.

Chapitre 6

Atelier de dessin

Les objectifs de ces travaux sont globalement similaires aux précédents. Toutefois, il s'appliquent, ici, au domaine de la création d'images numériques.

6.1 Présentation et objectifs

Ces travaux consistent à concevoir un programme permettant la réalisation simple de dessins. On peut le rapprocher d'outils graphiques existants tels que *Tux Paint*[1], *Paint*[2], *Art Rage*[3], par exemple. Il propose une palette d'outils permettant la création de formes sur un fond uni ou à partir d'images existantes. Ces outils vont des plus classiques, comme le pinceau ou la gomme, jusqu'à des outils plus originaux basés sur des méthodes génératives semi-autonomes.

S'inscrivant, bien sûr, dans le cadre de l'accessibilité, ce logiciel cible, lui aussi, le public le plus large possible. Il s'intéresse en priorité aux porteurs de handicaps moteurs au niveau des membres supérieurs, pour qui la pratique du dessin peut être particulièrement difficile. Les personnes souffrant de déficiences cognitives légères constituent l'autre cible principale. Comme pour la conception de l'outil de génération musicale, les personnes en situation de handicap sensoriel ne font pas partie du public prioritaire. L'accès pour les aveugles à la création graphique doit être traitée spécifiquement. Les autres handicaps sensoriels ne constituent pas un frein majeur à l'utilisation de ce type d'outil, aussi ils ne nécessitent pas de traitement particulier.

Comme nous l'avons vu précédemment, l'accessibilité se traduit, notamment, par des contraintes au niveau des interfaces. Dans le cas d'un atelier de dessin, une IHM minimale ne peut être suffisante. Nous avons en effet besoin d'une zone de dessin, de boutons d'accès aux outils, etc. Toutefois, les stratégies habituelles de conception d'interfaces ne sont pas pertinentes dans notre cas. Les habituels menus déroulants peuvent être délicats à utiliser sans une bonne motricité fine dans le maniement de la souris. D'ailleurs, ils sont souvent très chargés et usent fréquemment de termes techniques, comme on peut le constater avec

[1] http ://www.tuxpaint.org
[2] http ://windows.microsoft.com/fr-FR/windows7/products/features/paint
[3] http ://www.artrage.com/

l'exemple du logiciel *Inkscape*[4], sur la capture d'écran Fig. 6.1.

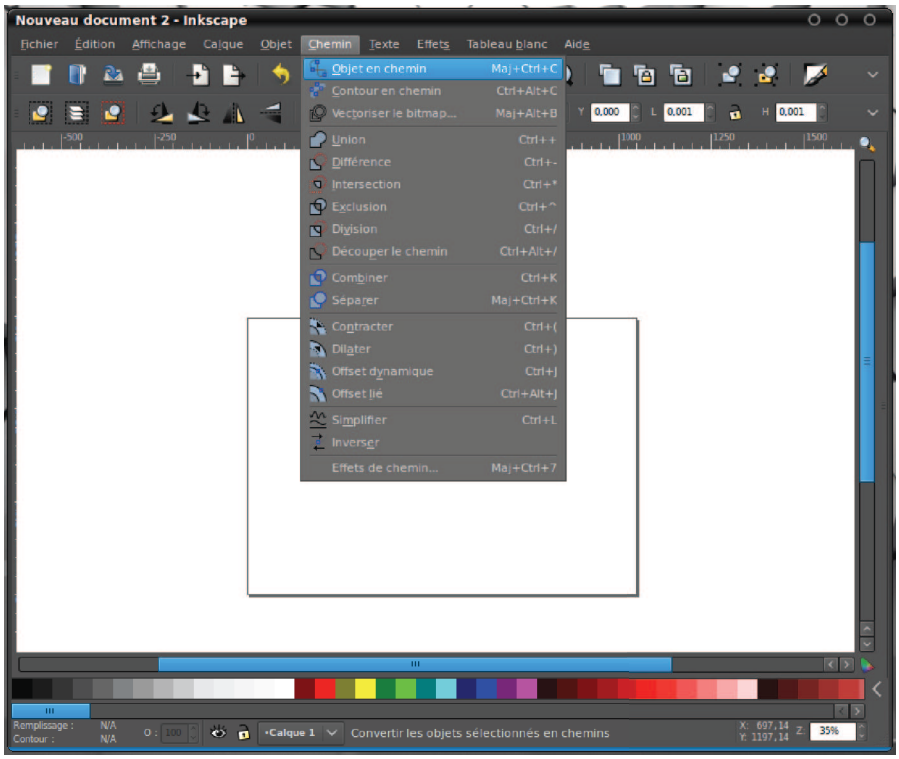


Fig. 6.1 – Exemple de menu déroulant

L'approche adoptée pour la conception de notre interface se base sur les principes suivants :

- utiliser des icônes plutôt que du texte, notamment pour permettre un accès aux enfants et aux personnes ayant des difficultés de lecture ;
- afficher constamment toute l'interface, pour ne pas solliciter la mémoire de l'utilisateur ;
- minimiser le nombre d'actions nécessaire à l'accès à un outil ;
- permettre le repositionnement de tous les éléments de l'interface, pour maximiser la surface utilisable pour le dessin et permettre une personnalisation importante de l'interface ;
- abstraire au maximum les considérations techniques à l'utilisateur, qui n'a, ainsi, pas besoin de savoir comment ça marche ;
- le logiciel doit fonctionner sur un ordinateur standard.

Le logiciel est multimodal : toutes les actions sont déclenchables soit à l'aide de la souris, soit à l'aide du clavier. Pour limiter la fatigue liée à l'utilisation prolongée de la souris, il n'est pas nécessaire de maintenir des boutons appuyés, et un mécanisme de menu contextuel permet de limiter les mouvements requis pour accéder aux outils, en rapprochant les boutons de la position actuelle du pointeur. La priorité est donnée, lors de la conception

[4]http ://inkscape.org/

des contrôles, à l'intuitivité et la limitation de la fatigue induite.

L'utilisation de méthodes automatiques de création graphique permet d'ajouter à l'atelier des outils de dessin ayant une part d'autonomie. Ils permettent avec peu de mouvements de la part de l'utilisateur d'obtenir des effets importants. Ils sont donc particulièrement indiqués dans notre cadre accessible où les utilisateurs ne sont pas toujours en mesure de réaliser des actions complexes. La palette des outils disponibles pouvant être relativement importante, plusieurs méthodes sont proposées :

- un algorithme de colonie de fourmis artificielles, proche de celui conçu par Nicolas Monmarché [Monmarché and Clair, 2009] ;
- plusieurs déclinaisons de L-System ;
- des fractales.

6.2 Solutions adoptées

Ce logiciel a été développé en utilisant le langage de programmation Java et la bibliothèque graphique Qt. Il est donc multi-plateforme ce qui permet de l'exécuter indépendamment du système d'exploitation utilisé.

L'interface est constituée d'une fenêtre principale qui héberge la zone de dessin. Pour être la plus grande possible, la zone de dessin recouvre l'intégralité de la fenêtre. Un ensemble de menus et barres d'outils vient la compléter. Ces éléments s'insèrent au dessus de la zone et peuvent tous être déplacés. D'une exécution à l'autre, la position de ces éléments est sauvegardée. Les différentes fonctionnalités du logiciel sont accessibles par des boutons circulaires affichant une image ainsi qu'un texte d'aide en pop-up. Le choix de la forme circulaire est due à la volonté d'afficher des images plutôt que du texte dans les boutons. Ces boutons sont regroupés en menus en fonction de leurs rôles. Le premier des éléments de l'interface (voir Fig. 6.2) est un menu en forme de fleur qui permet l'accès aux actions d'ordre général, telles qu'accéder à l'aide, sauvegarder, charger ou recommencer avec une image vierge.



Fig. 6.2 – Interface de l'atelier de dessin : menu principal

Un second menu, circulaire, permet de choisir la couleur utilisée par les outils de dessin.

Il permet de choisir rapidement, en deux clics, la couleur voulue à l'aide de deux zones concentriques. La zone externe permet de choisir la teinte générale, la zone interne ajuste cette teinte, le rond central donne un aperçu de la couleur choisie. La figure Fig. 6.3 montre ce menu. Il utilise un codage de couleur de type Teinte – Saturation – Valeur (TSV) plutôt que l'habituel Rouge – Vert – Bleu (RVB). Le codage TSV est plus proche de la perception humaine, et rends donc plus aisée la sélection de couleurs par notre utilisateur.



Fig. 6.3 – Interface de l'atelier de dessin : rosace de choix des couleurs

Deux autres menus serpentiformes permettent de choisir un outil de dessin. Cet alignement de boutons s'adapte en fonction de sa position. Il passe du vertical à l'horizontal quand des boutons viennent à sortir de l'écran, pour une plus grande adaptation de l'interface aux désirs de l'utilisateur. Le premier de ces menus regroupe les outils de base :

- un outil de sélection ;
- un crayon ;
- une gomme ;
- un aerographe ;
- un traceur de ligne ;
- un traceur de rayons ;
- un constructeur d'ellipse ;
- une constructeur de rectangle.

Le second menu comporte les outils basés sur des méthodes génératives que nous allons détailler ensuite.

Un menu pop-up (cf. Fig. 6.4) vient compléter cette interface. Il apparaît à côté du pointeur en cas d'inactivité, il regroupe les derniers outils utilisés et permet ainsi un accès rapide aux outils habituels de l'utilisateur. Une capture d'écran de l'interface se trouve sur la figure Fig. 6.5.

Ces menus offrent donc un accès rapide à toutes les fonctions du logiciel, bien entendu, une touche du clavier permet l'activation de chacun de ces outils, qui peuvent d'ailleurs être



FIG. 6.4 – Interface de l'atelier de dessin : menu Pop-up

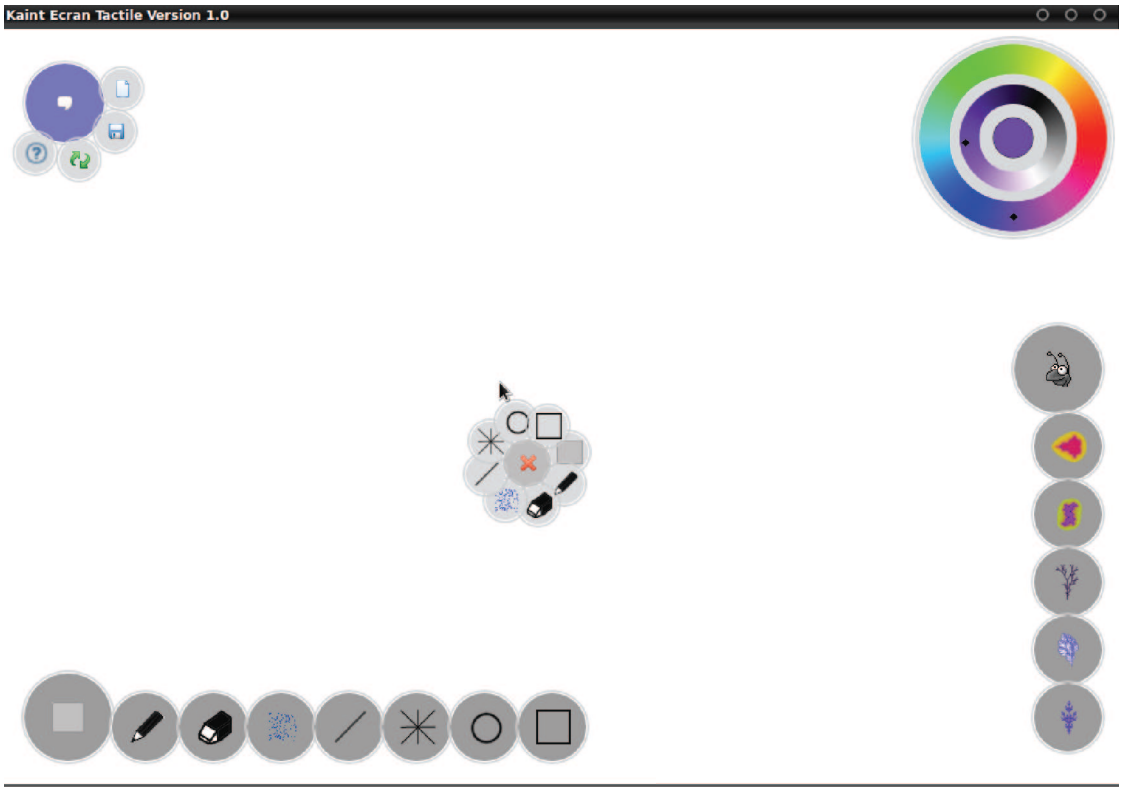


FIG. 6.5 – Interface de l'atelier de dessin

guidés soit à l'aide d'un clavier, soit via un dispositif de pointage. Pour limiter la fatigue liée au maintien de boutons appuyés, les outils de dessin fonctionnent comme des interrupteurs. Un clic de souris, ou une pression d'un bouton active l'outil, les mouvements créent alors les dessins. Une seconde pression le désactive et les déplacements cessent d'avoir un impact sur la zone de dessin.

Les outils avancés disponibles à l'heure actuelle sont au nombre de trois, certains sous plusieurs variantes.

6.2.1 Algorithme de fourmis artificielles

Un algorithme de colonie de fourmis artificielles, basé sur le comportement collectif de déplacement des fourmis, constitue un de ces outils semi-automatiques. Son mécanisme est proche de celui utilisé pour l'instrument de musique. N fourmis se promènent sur le dessin. À chaque pas de temps, elles choisissent chacune une direction en fonction des couleurs présentes sur le dessin et d'un comportement propre. Chaque fourmi est caractérisée par trois probabilités :

- P_f, sa probabilité d'aller en face ;
- P_g, celle d'aller à gauche ;
- P_d, celle d'aller à droite.

Dans cette approche, les couleurs sont assimilées aux phéromones. Chaque fourmi se trouve sur un pixel du dessin et lors de ses déplacements, colore le pixel qu'elle quitte avec une couleur C_l. Elle tend à suivre préférentiellement une couleur donnée : C_s, et est associée à une probabilité P_s de suivre cette couleur préférée. Ainsi, si cette couleur préférée se trouve sur un des pixels voisins, elle à P_s chance d'y aller. Sinon, elle se déplace par tirage au sort en utilisant P_f, P_g et P_d. L'ensemble des paramètres de l'algorithme sont repris au tableau Tab. 6.1.

Les quatre probabilités sont tirées au sort individuellement pour chaque fourmi, la couleur déposée est celle actuellement sélectionnée dans l'atelier, la couleur suivie est l'une de celle déjà présente sur le dessin, au hasard.

Tab. 6.1 – Paramètres de l'outil algorithme de fourmis

Paramètres	Notation	Valeur
Nombre de fourmis	N	interactive
Durée de vie des fourmis	D	interactive
Couleur déposée	C_l	interactive
Couleur suivie	C_s	aléatoire
Probabilité de suivre C_s	P_s	aléatoire
Probabilité d'aller tout droit	P_f	aléatoire
Probabilité d'aller à droite	P_d	aléatoire
Probabilité d'aller à gauche	P_g	aléatoire

Le contrôleur de l'outil permet de fixer les autres paramètres : le nombre de fourmis dépend de la durée entre les deux clics ou pressions de la touche qui enclenchent le lancement des fourmis. Cette durée permet aussi de fixer un autre paramètre de durée de vie des fourmis. Cette durée prend la forme d'un compteur qui est décrémenté à chaque pixel parcouru par une fourmi, une fois à zéro, cette fourmi disparaît. Il est bien sûr possible pour l'utilisateur d'arrêter le dessin avant que cette durée de vie soit atteinte. Les fourmis commencent toutes leur parcours au niveau du pointeur de la souris, mais avec une orienta-

tion de départ tirée au hasard. L'algorithme du fonctionnement de cet outil est décrit plus précisément dans l'algorithme 3. La figure Fig. 6.6 illustre un exemple de résultat obtenu.

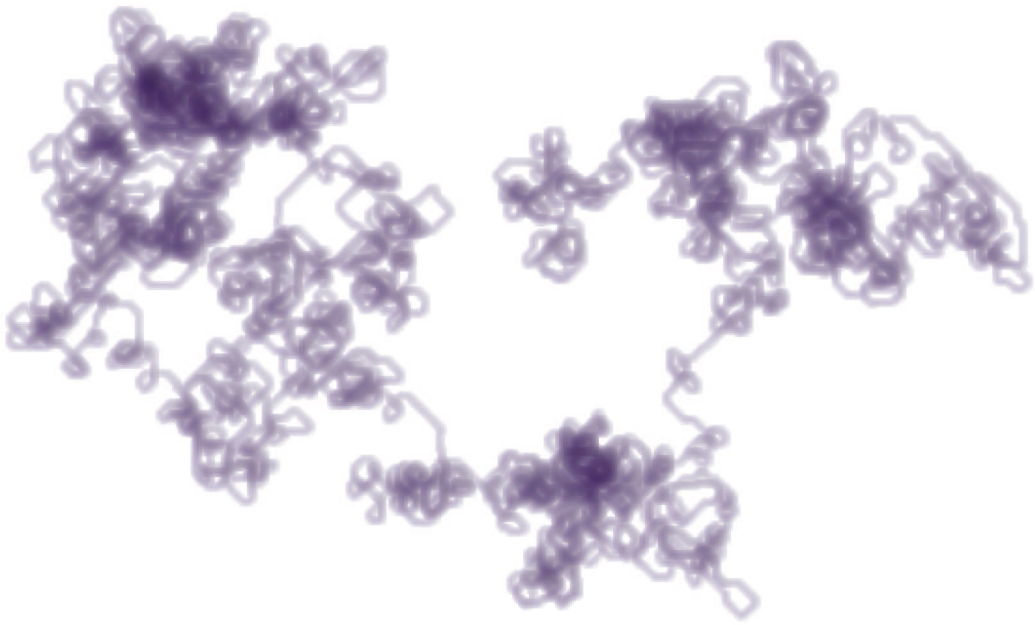


Fig. 6.6 – Outil fourmis : exemple avec deux fourmis déposant du bleu

6.2.2 Fractales

Deux outils permettent d'obtenir des fractales. Le processus itératif de construction de la fractale permet de visualiser sa construction et de l'arrêter à un moment choisi. L'initialisation des paramètres se fait en traçant une ligne sur le dessin qui servira à définir l'orientation, la position de départ de la fractale et la taille de la zone du dessin qu'elle va recouvrir.

Les fractales utilisées ici sont des représentations des ensembles de Mandelbrot et de Julia. Ces deux ensembles visualisent le comportement d'un système dynamique chaotique simple $f(z) = z^2 + c$ sur le plan complexe. Pour les représenter, on associe les z aux points du plan 2D. On calcule, à chaque itération, les points qui ne font pas partie de l'ensemble étudié et on leur affecte une couleur. Les couleurs utilisées pour colorer les fractales dépendent de la couleur sélectionnée au moment d'utiliser l'outil. Cette couleur permet de calculer un tableau de couleurs calculées depuis la couleur de base à l'aide de l'algorithme 4, dont le but est d'obtenir un dégradé de couleurs. Un exemple de résultat est reproduit sur la figure Fig. 6.7. Ainsi, les points ne faisant pas partie de l'ensemble à la $i^{\text{ème}}$ itération prennent la couleur de la case $i + 1$ du tableau. Pour détacher la fractale

du reste du dessin, on attribue une couleur transparente au point de la première itération.

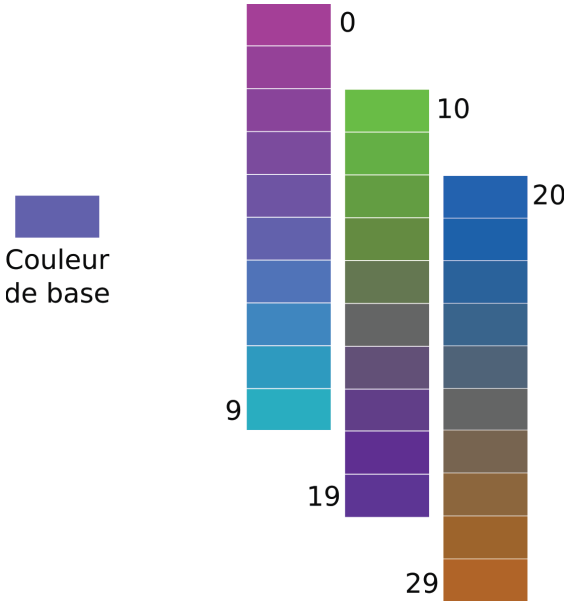


FIG. 6.7 – Exemple de tableau de couleurs de fractales

Le premier de ces outils représente l'ensemble de Mandelbrot. En partant de l'ensemble des points de coordonnées (a, b) correspondant à notre zone de dessin définie, on va déterminer itérativement les valeurs x_n et y_n correspondantes. À chaque itération, les x_n et y_n de chaque point sont calculés à partir de ses coordonnées et des x_{n-1} et y_{n-1} par les équations 6.1.

$$\begin{aligned} x_{n+1} &= x_n^2 - y_n^2 + a \\ y_{n+1} &= 2.x_n.y_n + b \\ \text{avec} \quad x_0 &= 0 \\ \text{et} \quad y_0 &= 0 \end{aligned} \tag{6.1}$$

On attribue aux pixels n'appartenant pas à l'ensemble de Mandelbrot, c'est-à-dire pour lesquels $x_n^2 + y_n^2 > 2^2$, une couleur qui dépend de l'itération en cours. Chaque nouvelle itération enrichit la figure. Un exemple de forme générée par cet outil se trouve sur la figure FIG. 6.8.

Le second outil représente l'ensemble de Julia. Il se calcule de façon similaire, mais cette fois-ci avec des valeurs de a et b fixes. Ce sont les valeurs de x_0 et y_0 qui correspondent à la zone de dessin. Les valeurs fixes utilisées sont reprises dans l'équation 6.2 et le résultat obtenu sur la figure FIG. 6.9.

$$\begin{aligned} x_{n+1} &= x_n^2 - y_n^2 + 0,3 \\ y_{n+1} &= 2.x_n.y_n - 0,5 \end{aligned} \tag{6.2}$$

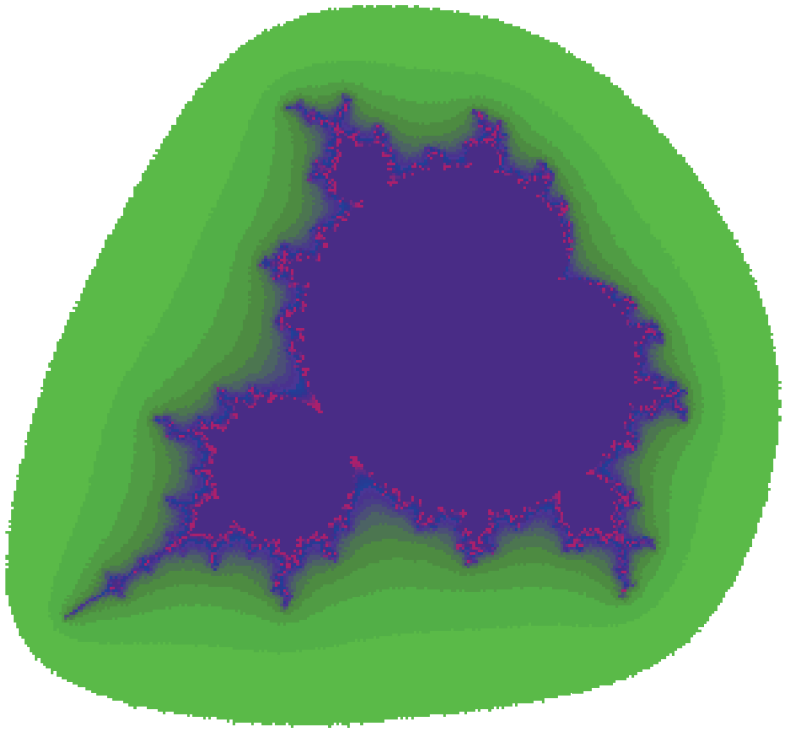


FIG. 6.8 – Outil fractale de Mandelbrot : exemple

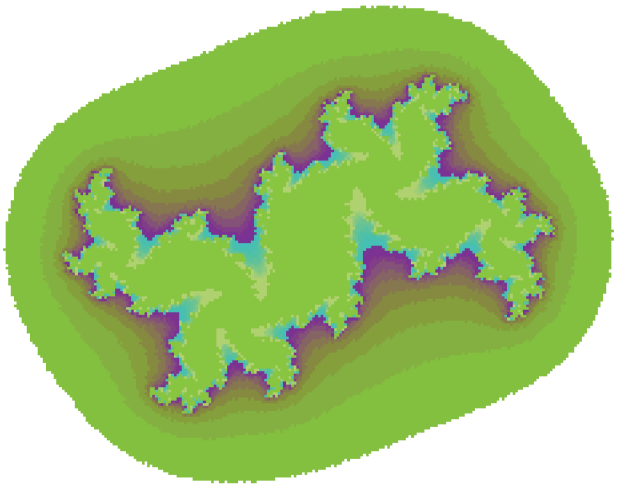


FIG. 6.9 – Outil fractale de Julia : exemple

6.2.3 L-System

Les L-System utilisés comportent cinq symboles, dont voici les significations :

F : trace un trait (une branche) ;

\+ : augmente l'inclinaison courante des branches ;

- : diminue l'inclinaison courante des branches ;

[: débute une sous-branche ;

] : termine une sous-branche.

Ils permettent de dessiner des structures arborescentes évoquant des plantes. Chaque début de branche est accrochée à l'extrémité de la branche dessinée précédemment. Le trait est orienté en fonction de l'angle d'inclinaison courant. Le symbole de début de sous-branche enregistre l'extrémité de la dernière branche dessinée. Le symbole de fin ramène la position de dessin à la position enregistrée à l'ouverture de sous-branche correspondante.

De nombreux outils peuvent être développés sur ce modèle qui diffèrent par la grammaire effectivement utilisée. Deux exemples, 6.3 et 6.4 permettent de générer des formes telles que celles présentes sur les figures Fig. 6.10 et Fig. 6.11.

Grammaire de l'outil L-System 1 :

$$\begin{aligned} S &= X \\ X &\rightarrow F[+X]F[-X] + X \\ F &\rightarrow FF \end{aligned} \tag{6.3}$$

Grammaire de l'outil L-System 2 :

$$\begin{aligned} S &= F \\ F &\rightarrow FF - [-F + F + F] + [+F - F - F] \end{aligned} \tag{6.4}$$

Ces deux outils s'utilisent sensiblement de la même manière que les outils fractals. Le dessin d'un trait sur la zone de dessin permet de déterminer la position de la racine, l'orientation de l'arbre et sa taille maximale. Le motif se forme par itération, en appliquant les règles de la grammaire à chaque étape, jusqu'à ce que la taille maximale soit atteinte, ou que l'utilisateur ait arrêté manuellement la génération.

6.3 Résultats et perspectives

L'atelier de dessin accessible a fait l'objet d'une présentation à la conférence Generative Art en 2009 [Clair et al., 2009a]. Plusieurs versions de l'atelier de dessin ont été présentées à des utilisateurs tests. Ces retours nous ont déjà permis de faire évoluer le programme, de corriger certains comportements de l'interface, etc.

L'utilisation d'un écran tactile semble particulièrement pertinente pour un logiciel de ce genre. Toutefois, même si ces écrans se comportent comme des dispositifs de pointage, ils n'étaient pas directement utilisables pour l'atelier. Le principe d'activation/désactivation

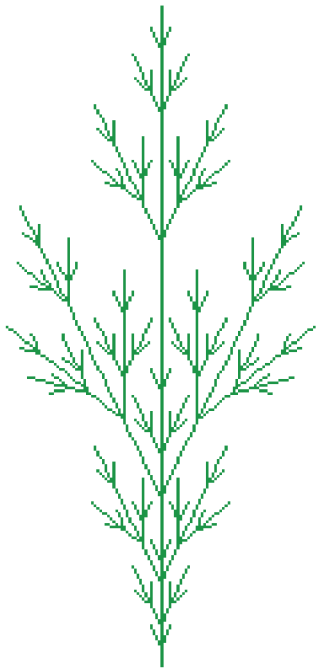


FIG. 6.10 – Outil L-System 1 : exemple

des outils, pour limiter les longs maintiens de pression sur les boutons, est opposé à la manière dont fonctionne un écran tactile. Sur ce type de périphérique, chaque pression du doigt sur l'écran entraîne l'équivalent d'un clic gauche, mais aussi un déplacement du pointeur à la position d'appui. Ainsi, les déplacements du pointeur, contrairement à la souris, se font par saut d'une position à une autre position éventuellement éloignée. Dans notre cas où c'est souvent la trajectoire qui nous intéresse puisque nous cherchons à la dessiner, l'utilisateur se retrouve à déplacer son doigt sur l'écran, en maintenant la pression. C'est un fonctionnement très intuitif, mais qui déclenche un grand nombre de clics. L'atelier a donc évolué, et dans sa seconde version, le programme propose au démarrage, de fonctionner avec un écran tactile ou un dispositif de pointage classique.

Parmi les outils avancés, nous avons aussi fait de nombreux tests de développement basés sur des algorithmes évolutionnaires interactifs. Plusieurs modèles d'individus ont été conçus, notamment basés sur l'idée d'une sélection interactive de zones préférées dans l'image. L'utilisateur sélectionne une zone rectangulaire, cette étape tient lieu de sélection.

L'individu considéré est un tableau à une dimension qui constitue une version linéaire de la sélection. Il contient les couleurs de chaque pixel. Dans les modèles essayés, l'individu sélectionné est croisé avec ses huit voisins directs. Les opérateurs de croisement classiques ne sont clairement pas adaptés pour ce genre de techniques parce qu'ils induisent de simples découpes et déplacements de morceaux d'images, ce qui ne semble pas pertinent, en tout cas, sans une sélection plus fine des individus, par exemple basée sur une détection de

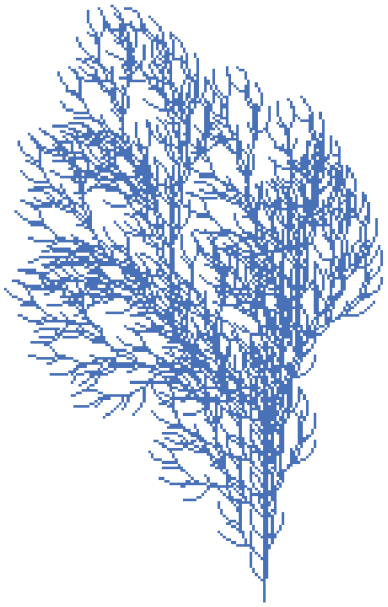


FIG. 6.11 – Outil L-System 2 : exemple

motifs particuliers dans une sélection. Nous avons testé trois méthodes de croisement :

- Le premier opérateur de croisement testé consiste, pour chaque voisin, à tirer au hasard un ensemble de cases du tableau et à remplacer leur couleur par celle de la même case de l'individu sélectionné.
- Le second opérateur vise à remplacer complètement une couleur dans chaque voisin. Pour chaque voisin, on tire au hasard un indice de case i. On récupère la couleur C_n présente dans la case i de l'individu sélectionné. On récupère aussi la couleur C_a présente dans la case i de ce voisin. On remplace ensuite, dans ce voisin, toutes les occurrences de C_a par C_n.
- Le troisième opérateur est proche du second, mais utilise non pas une seule couleur, mais un ensemble de couleurs proches. Ces couleurs sont déterminées par un petit intervalle sur une des composantes RVB.

Ces trois opérateurs produisent des résultats différents, principalement selon le type d'images auxquels ils s'appliquent (photos, formes sur fond blanc, dessin abstrait...). Dans l'ensemble, ces résultats ne nous ont pas semblés satisfaisants, soit du fait de leur tendance à détruire la structure de l'image, sans pour autant en créer une nouvelle, soit parce qu'ils produisent une modification presque invisible de l'image. Si, en l'état, les résultats obtenus en fouillant cette piste des algorithmes évolutionnaires ne sont pas satisfaisants, cette piste n'est pas encore à écarter pour autant. Une des idées restant à essayer consiste à travailler uniquement sur les couleurs en tentant de conserver la structure générale de l'image. Dans cette idée, l'individu est une portion de l'image, donc un tableau, où chaque case ne contient

pas directement sa couleur, mais l'indice de cette couleur dans un second tableau. Cette autre tableau, trié, contient l'ensemble des couleurs qui compose l'image. Un croisement sur ces tableaux de couleurs triés pourrait offrir des résultats intéressants, en modifiant les couleurs tout en conservant la structure globale de l'image.

L'atelier de dessin a lui aussi été présenté lors d'évènements grand public. Il a ainsi été présenté sur les stands de la Fête de la science, des Journées portes ouvertes, au Challenge Handicap et Technologie ainsi que sur certaines conférences scientifiques. Contrairement à l'instrument de musique, l'atelier de dessin se rapproche plus facilement de programmes connus, aussi suscite-t-il moins d'étonnement. On ne rencontre ainsi pas de déception par rapport à une idée préalable. Par contre, ce programme souffre, dans certains cas, de la comparaison avec d'autres, qui n'ont pourtant pas les mêmes objectifs. Au delà des remarques sur les améliorations possibles, les retours obtenus lors de ces démonstrations publiques, témoignent d'un intérêt certain pour cet outil, qui invite à pousser ces travaux plus loin.

Algorithm 3 Algorithme de dessin de fourmis

repeat
  for Chaque fourmi **do**
    $P'_f \leftarrow P_f$
    $P'_g \leftarrow P_g$
    $P'_d \leftarrow P_d$
    On examine les couleurs des cases voisines C_f, C_g et C_d
    On tire un nombre au hasard $R1$ entre 0 et 1
    if $R1 < P_s$ ET $C_s \in$ au moins une des cases voisines **then**
      if $C_s \neq C_f$ **then**
        $P'_f \leftarrow 0$
      end if
      if $C_s \neq C_g$ **then**
        $P'_g \leftarrow 0$
      end if
      if $C_s \neq C_d$ **then**
        $P'_d \leftarrow 0$
      end if
    else
      if $C_s = C_f$ **then**
        $P'_f \leftarrow 0$
      end if
      if $C_s = C_g$ **then**
        $P'_g \leftarrow 0$
      end if
      if $C_s = C_d$ **then**
        $P'_d \leftarrow 0$
      end if
    end if
    $P_t \leftarrow P'_f + P'_g + P'_d$
    On tire un nombre au hasard $R2$ entre 0 et P_t
    if $R2 < P'_f$ **then**
      La fourmi va sur la case en face et la colore avec C_l
    else if $R2 < (P'_f + P'_g)$ **then**
      La fourmi va sur la case de gauche et la colore avec C_l
    else
      La fourmi va sur la case de droite et la colore avec C_l
    end if
  end for
  $D \leftarrow D - 1$
until $D = 0$

Algorithm 4 Algorithme de calcul des couleurs des fractales

On part d'une couleur de base dont les composantes RVB sont : R_b, V_b et B_b
On utilise un tableau de couleurs codées en RVB
for $i \in [0..9]$ **do**
    TableauCouleur$[i] \leftarrow (200 - 20 * i, 20 * i, B_b)$
    TableauCouleur$[i] \leftarrow (R_b, 200 - 20 * i, 20 * i)$
    TableauCouleur$[i] \leftarrow (20 * i, V_b, 200 - 20 * i)$
end for

Conclusion

Cette étude de méthodes de production d'art génératif et de leur application pour la conception d'outils de création artistique accessibles nous a ainsi permis de travailler sur l'amélioration de l'accessibilité des logiciels informatiques. Cette amélioration impose de suivre, dès la conception, un ensemble de règles et de bonnes pratiques que nous avons détaillées dans cette thèse. Ces règles ont servi de cadre à la conception de deux logiciels de création artistique accessibles.

L'étude du domaine des systèmes de production d'arts génératifs, nous a conduit à proposer leur utilisation au sein de ces outils, comme une forme d'assistance à l'utilisateur. En effet, des méthodes génératives, adaptées pour interagir avec l'utilisateur peuvent servir d'outils de création artistique permettant l'obtention de résultats complexes à partir de peu d'actions.

La première des réalisations menées consiste en un logiciel instrument de musique virtuel. Ce programme est structuré autour d'interfaces accessibles et offre un système d'accompagnement automatique basé sur un algorithme de colonie de fourmis artificielles. Nos travaux ont abouti à un prototype utilisable qui a permis de procéder à une réelle évaluation de terrain, donc de mettre le logiciel face à de véritables utilisateurs. Les résultats de cette évaluation sont encourageants.

Les seconds travaux ont suivi la piste d'un atelier de dessin accessible. Ce programme applique lui aussi les contraintes d'accessibilités mises en évidence, et propose, entre autres, des outils de dessin génératif, basés sur un algorithme de colonie de fourmis artificielles, des grammaires génératives et des représentations d'espace fractale. Ce logiciel a été réalisé en collaboration avec des étudiants, ce qui a permis d'explorer de nombreuses pistes, notamment sur l'idée d'algorithmes évolutionnaires interactifs. Cet outil a lui aussi été confronté aux retours de véritables utilisateurs.

Ces travaux nous ont permis de valider plusieurs points :
- Il est possible de développer des outils informatiques accessibles de création artistique.
- Des algorithmes d'art génératif, adaptés pour être interactifs, s'intègrent bien à ce type d'outils.
- Les recommandations proposées permettent d'obtenir des applications plus accessibles.
- Les résultats des tests montrent un vif enthousiasme du public pour ces deux réalisations.

Le programme instrument de musique a d'ailleurs été réclamé et est utilisé à l'heure actuelle par plusieurs personnes, certaines souffrant de handicap.

Plus généralement, ces deux pistes ouvrent des perspectives prometteuses sur des logiciels utilisant des algorithmes interactifs d'art génératif. Sans doute peut-on, ainsi, offrir une forme nouvelle de création artistique, accessible au plus grand nombre.

Les recommandations proposées dans cette thèse, pour le développement d'applications accessibles s'appliquent, en majeure partie, à n'importe quel logiciel. Leur application résulte, d'ailleurs, en des améliorations utilisables par tous et peuvent amener des réflexions particulièrement bénéfiques lors d'une phase de conception logicielle. Le principal frein à une large application de ces bonnes pratiques est la méconnaissance générale que les concepteurs d'applications en ont. Aussi, l'implication dans une démarche d'amélioration globale de l'accessibilité informatique est un travail de tous les jours, pour sensibiliser et informer les acteurs de l'informatique. Une position entre l'enseignement et la recherche est dans ce cas particulièrement intéressante, puisqu'elle permet de bien appréhender ces problèmes et d'intervenir pleinement dans la formation des concepteurs d'application de demain.

Ces travaux se sont confrontés à la nécessaire évaluation, propre à tout travaux scientifiques. Deux principales difficultés ont alors été rencontrées. La première est liée au domaine de l'accessibilité. En effet, dans un domaine ou les problèmes du public visé sont souvent différents des nôtres, il faut s'assurer de bien répondre à leurs attentes. La seconde est liée au domaine artistique, où l'évaluation d'une œuvre est en grande partie subjective. Loin des habituels preuves et tests de performances du domaine informatique, des évaluations de terrain, en contact avec les utilisateurs finaux, nous ont permis de pallier ces difficultés. Néanmoins, ces évaluations sont très gourmandes en temps et demande de trouver des volontaires pour les tests, d'établir des protocoles d'évaluation, d'organiser des rencontres, etc. Il s'agit là de difficultés supplémentaires qu'il faut anticiper lorsqu'on est confronté à la conception d'applications dans ce cadre.

Ces travaux ont bien sûr vocation à être poursuivis dans l'avenir, pour aboutir un jour, du moins nous l'espérons, à leur totale mise à disposition du public. Les pistes de développement de chacune de ces deux réalisations, déjà évoquées précédemment, devraient aboutir à des outils riches et adaptables tant aux capacités qu'aux désirs de chacun.

Des rapprochements entre ces outils accessibles et des outils pédagogiques ont été fréquemment faits par les évaluateurs rencontrés. En effet, puisqu'un des objectifs est d'être accessible à ceux ne disposant pas des connaissances nécessaires à la pratique d'une forme de création artistique, on peut profiter des connaissances incorporées pour faire évoluer ces programmes vers des outils d'apprentissage. Il consisterait alors à faire progresser l'utilisateur en lui expliquant progressivement les contraintes mises en place par l'outil, qui lui servent de guides, quitte à les faire disparaitre pour lui permettre une plus grande liberté. Il s'agirait alors d'une forme de pédagogie par la pratique, proche de la façon dont les enfants apprennent à faire du vélo avec les petites roues stabilisatrices.

Du point de vue des interfaces de contrôle, nous avons fait des propositions qui, soit s'inspirent d'instruments réels, éventuellement en leur ajoutant des mécanismes nouveaux, soit profitent des possibilités offertes par le coté virtuel pour explorer des pistes nouvelles. De nombreuses autres interfaces peuvent être développées. Devant l'immensité des possibilités, la collaboration avec des ergonomes peut alors servir à guider la conception de ces interfaces. Par ailleurs, de nombreuses interfaces novatrices sont disponibles, développées

par des passionnés, parfois utilisant des périphériques différents. Une collecte et une étude de toutes ces idées peut être riche en enseignements.

D'autres travaux sur des logiciels d'aide à la création artistique peuvent, bien sûr, être lancés. En espérant que le cadre défini ici, comme les deux réalisations déjà menées, puissent servir à la fois de guide et de source d'inspiration.

En restant dans les domaines déjà abordés ici, des outils de composition, qui permettent donc d'écrire de la musique sans la jouer immédiatement, ouvrent la voie à l'exploration d'autres méthodes. En effet, puisqu'il est ainsi possible de revenir en arrière, d'essayer plusieurs options, finalement, de concevoir plus finement et surtout au préalable la musique qui sera interprétée plus tard, des programmes de composition assistée peuvent, ainsi, faire appel à d'autres méthodes que celle que nous avons pu tester ici. Les algorithmes évolutionnaires, notamment, s'adaptent beaucoup mieux à ce genre de contexte où il est possible de procéder à des évaluations successives.

Dans d'autres domaines artistiques, des programmes de création accessibles, faisant appel à des méthodes génératives peuvent être développer. Les machines de prototypage rapide, telles que les imprimantes 3D, par exemple, pourraient permettre de réaliser des sculptures réelles conçues à l'aide d'outils de création artistique accessibles.

La littérature pourrait, elle aussi, se voir dotée d'outils d'aide à la création. Néanmoins il s'agit là d'un domaine un peu particulier où les règles sont nombreuses et souvent ambigües. La conception de tels outils, en association avec des techniques de traitement automatique de langues naturelles, est un défi particulièrement intéressant.

Annexes

Annexe A

Références techniques

Tab. A.1 – Instruments *General MIDI*-1

Banque	Programme	Nom
0	000	Grand piano acoustique
0	001	Piano acoustique
0	002	Grand piano électrique
0	003	Piano bastringue (Honky-Tonk Piano)
0	004	Piano électrique (Rhodes)
0	005	Piano - effet chorus
0	006	Clavecin
0	007	Clavinet
0	008	Celesta
0	009	Glockenspiel
0	010	Boîte à musique
0	011	Vibraphone
0	012	Marimba
0	013	Xylophone
0	014	Cloches tubulaires
0	015	Tympanon (Dulcimer)
0	016	Orgue Hammond
0	017	Orgue à percussion
0	018	Orgue - Rock
0	019	Grandes orgues
0	020	Harmonium
0	021	Accordéon
0	022	Harmonica
0	023	Accordéon tango (Bandonéon)
0	024	Guitare classique
0	025	Guitare folk
0	026	Guitare électrique - Jazz
0	027	Guitare électrique - son clair
0	028	Guitare électrique - sourdine
0	029	Guitare saturée (Overdrive)
0	030	Guitare avec distorsion
0	031	Harmoniques de guitare
0	032	Basse acoustique
0	033	Basse électrique
0	034	Basse électrique - médiator
0	035	Basse fretless
0	036	Basse - slap 1
0	037	Basse - slap 2
0	038	Basse synthé 1
0	039	Basse synthé 2

TAB. A.2 – Instruments *General MIDI*-2

Banque	Programme	Nom
0	040	Violon
0	041	Violon alto
0	042	Violoncelle
0	043	Contrebasse
0	044	Cordes - trémolo
0	045	Cordes - pizzicato
0	046	Harpe
0	047	Timbales
0	048	Ensemble acoustique à Cordes 1
0	049	Ensemble acoustique à Cordes 2
0	050	Cordes synthé 1
0	051	Cordes synthé 2
0	052	Chœur - "Aah"
0	053	Chœur - "Ooh"
0	054	Voix synthétique
0	055	Coup d'orchestre
0	056	trompette
0	057	Trombone
0	058	Tuba
0	059	trompette en sourdine
0	060	Cor d'harmonie
0	061	Section de cuivres
0	062	Cuivres synthé 1
0	063	Cuivres synthé 2
0	064	Saxophone soprano
0	065	Saxophone alto
0	066	Saxophone ténor
0	067	Saxophone baryton
0	068	Hautbois
0	069	Cor anglais
0	070	Basson
0	071	Clarinette
0	072	Piccolo
0	073	Flûte
0	074	Flûte à bec
0	075	Flûte de pan
0	076	Bouteille - souffle
0	077	Shakuhachi
0	078	Sifflet
0	079	Ocarina

Tab. A.3 – Instruments *General MIDI*-3

Banque	Programme	Nom
0	080	Synthé - Signal carré
0	081	Synthé Signal dents de scie
0	082	Orgue à vapeur
0	083	Chiffer
0	084	Charang
0	085	Voix solo
0	086	Signal dent de scie en quinte
0	087	Basse solo
0	088	Fantaisie (new age)
0	089	Son chaleureux
0	090	Polysynthé
0	091	Chœur
0	092	Archet
0	093	Métallique
0	094	Halo
0	095	Balai
0	096	Pluie de glace
0	097	Trames sonores
0	098	Cristal
0	099	Atmosphère
0	100	Brillance
0	101	Gobelins
0	102	Échos
0	103	Espace
0	104	Sitar
0	105	Banjo
0	106	Shamisen
0	107	Koto
0	108	Kalimba
0	109	Cornemuse
0	110	Viole
0	111	Shehnai
0	112	Clochettes
0	113	Agogo
0	114	Batterie métallique
0	115	Planchettes
0	116	Timbales
0	117	Tom mélodique
0	118	Tambour synthétique
0	119	Cymbale - inversée

Tab. A.4 – Instruments *General MIDI*-4

Banque	Programme	Nom
0	120	Guitare - bruit de frette
0	121	Respiration
0	122	Rivage
0	123	Gazouilli
0	124	Sonnerie de téléphone
0	125	Hélicoptère
0	126	Applaudissements
0	127	Coup de feu

Tab. A.5 – Codes des percussions (canal 10) – 1

Code	Instrument
27	Balais
28	Roulement de balais
29	Roulement de caisse claire
30	Castagnettes
31	Caisse claire 0
32	Coup de baguette
33	Grosse caisse 0
34	Caisse claire avec rimshot
35	Grosse caisse médium
36	Grosse caisse haute
37	Coup de métronome
38	Caisse claire 1
39	Claquement de main
40	Caisse claire 2
41	Tom basse grave
42	Charley frappé
43	Tom basse aigu
44	Charley au pied
45	Tom médium 4
46	Charley ouvert
47	Tom médium 3
48	Tom médium 2
49	Cymbale crash
50	Tom aigu
51	Ride
52	Cymbale china
53	Cymbale ride/cup
54	Tambourin
55	Cymbale splash
56	Meuglement de vache Cow
57	Crash 2
58	Vibraslap
59	Cymbale ride aigue
60	Bongo aigu
61	Bongo grave
62	Congas aigu sourd
63	Congas aigu ouvert
64	Congas grave
65	Timbales aigu
66	Timbales grave

Tab. A.6 – Codes des percussions (canal 10) – 2

Code	Instrument
67	Cloche agogo aiguë
68	Cloche agogo grave
69	Cabasa
70	Maracas
71	Sifflet aigu
72	Sifflet grave
73	Guiro court
74	Guiro long
75	Claves
76	Woodblock aigu
77	Woodblock grave
78	Cuica assourdie
79	Cuica ouverte
80	Triangle tenu
81	Triangle libre
82	Shaker
83	Grappe de clochettes
84	Arbre à cloches

Bibliographie

[Ando and Iba, 2005] Ando, D. and Iba, H. (2005). Real-time musical interaction between musician and multi-agent system. In *8th Generative Art Conference*.

[Art Prime, 2010] Art Prime (2010). Site de l'association art prime. http ://www.art-prime.com/. Site web.

[Aupetit, 2005] Aupetit, S. (2005). *Contribution aux modèles de Markov cachés : Metaheuristique d'apprentissage, nouveaux modèles et visualisation de dissimilarité*. PhD thesis, Université François-Rabelais Tours, Tours, France.

[Aupetit et al., 2003] Aupetit, S., Bordeau, V., Monmarché, N., Slimane, M., and Venturini, G. (2003). Interactive evolution of ant paintings. In *IEEE Congress on Evolutionary Computation*, volume 2, pages 1376–1383, Canberra.

[Aupetit et al., 2009] Aupetit, S., Monmarché, N., and Slimane, M. (2009). Apprentissage des modèles de markov cachés par l'algorithme api. In Monmarché, N., Guinand, F., and Siarry, P., editors, *Fourmis Artificielles, des bases de l'optimisation aux applications industrielles*, volume 1 of *Traité IC2*, chapter 12. Hermès-Lavoisier.

[Azzag et al., 2003] Azzag, H., Monmarché, N., Slimane, M., Venturini, G., and Guinot, C. (2003). Algorithme anttree : Classification non supervisée par des fourmis artificielles. In *Revue des nouvelles technologies et l'information*, pages 75–86, Lyon. Cepadues.

[Bentley, 2002] Bentley, K. A. (2002). Exploring aesthetic pattern formation. In Soddu, C., editor, *5th Generative Art Conference (GA2002)*, Milan, Italie.

[Berners-Lee, 1997] Berners-Lee, T. (1997). Définition de l'accessibilité du web. Site web d'accessiweb : `http://www.accessiweb.org/fr/Label_Accessibilite/`.

[Bertelle et al., 2003] Bertelle, C., Dutot, A., Lerebourg, S., Olivier, D., and Prévost, G. (2003). An actor architecture to develop games for blind children. In *GAME-ON'2003 Conf*, London (United Kingdom). Conférencier invité.

[Biles, 1994] Biles, J. A. (1994). Genjam : A genetic algorithm for generating jazz solos. In *International Computer Music Conference (ICMC'94)*. The Computer Music Association.

[Biles, 2002] Biles, J. A. (2002). Genjam in transition : from genetic jammer to generative jammer. In *Generative Art and Design Conference (GA2002)*, Politecnico di Milano University, Italy, Milan. `http://www.it.rit.edu/~jab/GenJam.html`.

[Biles, 2007a] Biles, J. A. (2007a). Evolutionary computation for musical tasks. In Miranda, E. R. and Biles, J. A., editors, *Evolutionary Computer Music*, number 2 in Computer Science, chapter 2, pages 28–51. Springer, London.

[Biles, 2007b] Biles, J. A. (2007b). Improvizing with genetic algorithms : Genjam. In Miranda, E. R. and Al, B. J., editors, *Evolutionary Computer Music*, chapter 7, pages 137 – 169. Springer, London.

[Bird and Stokes, 2006] Bird, J. and Stokes, D. (2006). Evolving fractal drawings. In Soddu, C., editor, *Generative Art 2006 Proceedings of the 9th International Conference*, pages 317–327, Milano.

[Bisig et al., 2007] Bisig, D., Neukom, M., and Flury, J. (2007). Interactive swarm orchestra. In Soddu, C., editor, *10th Generative art conference (GA2007)*, Milan, Italie.

[Blackwell, 2006] Blackwell, T. (2006). Swarming and music. Technical report, Goldsmiths college, Londres.

[Blackwell, 2007] Blackwell, T. (2007). Swarming and music. In Miranda, E. R. and Biles, J. A., editors, *Evolutionary Computer Music*, chapter 9, pages 194 – 217. Springer, London.

[Blackwell and Young, 2004] Blackwell, T. and Young, M. (2004). Swarm granulator. In *2nd European Workshop on Evolutionary Music and Art (EVOMUSART2004)*, Coimbra, Portugal. Springer.

[Boden, 1996] Boden, M. A., editor (1996). *Dimensions of Creativity.* MIT Press, Cambridge, Massachussets, États-unis.

[Boden, 2007] Boden, M. A. (2007). *Creativity and Artificial Evolution.* MIT Press.

[Bonabeau et al., 1999] Bonabeau, E., Dorigo, M., and Theraulaz, G. (1999). *Swarm Intelligence : From Natural to Artificial Systems.* Santa Fe Institute Studies on the Sciences of Complexity. Oxford University Press.

[Boyd et al., 2004] Boyd, J. E., Hushlak, G., and Jacob, C. J. (2004). Swarmart : Interactive art from swarm intelligence. In *Proceedings of the 12th annual ACM international conference on Multimedia (ACM Multimedia 2004)*, pages 628 – 635, New York, USA. Association for Computing Machinery (ACM).

[Burger, 2001] Burger, D. (2001). Les nouvelles technologies et l'accès à l'information pour les personnes handicapées visuelles. In *29èmes entretiens de médecine physique et de réadaptation de Montpellier*.

[Caldwell et al., 2008] Caldwell, B., Cooper, M., Reid, L. G., and Vanderheiden, G. (2008). Web content accessibility guidelines (wcag) 2.0. http ://www.w3.org/TR/WCAG20/.

[Chareyron, 1990] Chareyron, J. (1990). Digital synthesis of self-modifying waveforms by means of linear automata. In Keislar, D., editor, *Computer Music Journal*, volume 14, pages 25–41. MIT Press, Cambridge, Etats-unis.

[Chen and Miikkulainen, 2001] Chen, C.-C. J. and Miikkulainen, R. (2001). Creating melodies with evolving recurrent neural networks. In *the 2001 International Joint Conference on Neural Networks (IJCNN-01)*, Washington, DC.

[Chomsky, 1957] Chomsky, N. (1957). *Syntactic structures.* De Gruyter, Berlin.

[Ciampi and Lechevallier, 1995] Ciampi, A. and Lechevallier, Y. (1995). Reseaux de neurones et modeles statistiques. *La Revue de Modulad*, 15 :27–46.

[Clair et al., 2009a] Clair, R., Monmarché, N., and Slimane, M. (2009a). Accessible art creation tools, a generative arts application. In Soddu, C., editor, *12th Generative Art Conference (GA2009)*, page 9, Milano, Italy.

[Clair et al., 2009b] Clair, R., Monmarché, N., and Slimane, M. (2009b). An interactive ant algorithm for real-time music accompaniment. In *Évolution Artificielle, 9th International Conference on Artificial Evolution*, page 12, Strasbourg, France.

[Clair et al., 2010] Clair, R., Monmarché, N., and Slimane, M. (2010). Accessibilité de la création musicale par ordinateur : l'exemple d'un instrument virtuel avec accompagnement automatique. In *Handicap 2010*, Paris.

[Clair et al., 2008a] Clair, R., Monmarché, N., and Slimane, M. (2008a). Art, an accessibility issue. In *Accessible Design in the Digital World (ADDW08)*, page 9, York, UK.

[Clair et al., 2008b] Clair, R., Monmarché, N., and Slimane, M. (2008b). Interactions between an artificial colony of musical ants and an impaired human composer : towards accessible generative arts. In Soddu, C., editor, *XI Generative Art conference (GA2008)*, page 10, Milano, Italy.

[Colas et al., 2007] Colas, S., Monmarché, N., Gaucher, P., and Slimane, M. (2007). Artificial ants for the optimization of virtual keyboard arrangement for disable people. In Monmarché, N., Talbi, E.-G., Collet, P., Schoenauer, M., and Lutton, E., editors, *Évolution Artificielle, 8th International Conference on Artificial Evolution*, volume 4926 of *Lecture Notes in Computer Science*, pages 87–98, Tours, France. Springer.

[Conklin, 2003] Conklin, D. (2003). Music generation from statistical models. In *Proceedings of the Symposium on Artificial Intelligence and Creativity in the Arts and Sciences (AISB 2003)*, pages 30–35, Aberystwyth, Wales.

[Copper, 1995] Copper, A. (1995). The myth of metaphor. *Visual Basic Programmer's Journal*. Traduit par Marc Wathieu.

[Cott, 1979] Cott, J. (1979). *Conversations avec Stockhausen*. Jean-Claude Lattès. traduit de l'anglais par Jacques Drillon.

[Dahlstedt, 2001] Dahlstedt, P. (2001). A mutasynth in parameter space : interactive composition through evolution. In Landy, L., editor, *Organised Sound*, volume 6, pages 121–124. Cambridge University Press, Royaume uni.

[De Felice and Abbattista, 2002] De Felice, F. and Abbattista, F. (2002). Genorchestra : An interactive evolutionary agent for musical composition. In *Generative Art*.

[Dehlinger, 1998] Dehlinger, H. E. (1998). The artist's intentions and genetic coding in algorithmically generated drawings. In *Generative Art : Proceedings of the 1st International Conference GA 98*, Milano. Editrice Librerie Dedalo.

[Desainte-Catherine and Marchand, 1999] Desainte-Catherine, M. and Marchand, S. (1999). Vers un modèle pour unifier musique et son dans une composition multiéchelle. In *Proceedings of the Journees dInformatique Musicale (JIM99)*, pages 59–68, Paris, France.

[Devetakovic Radojevic and Turner, 2002] Devetakovic Radojevic, M. and Turner, R. (2002). Spatial forms generated by music – the case study. In Soddu, C., editor, *5th Generative Art Conference (GA2002)*, Milan, Italie.

[Dewdney, 1981] Dewdney, A. K. (1981). Cristallisation en spirale : Un univers de débris, de mosaïques, de défauts et de démons. *Pour la science*, 144.

[Donegan et al., 2009] Donegan, M., Morris, J. D., Corno, Fulvio Signorile, I., Chio, A., Pasian, V., Vignola, A., Buchholz, M., and Holmqvist, E. (2009). Understanding users and their needs. In Stephanidis, C., editor, *Universal Access in the Information Society*, volume 8 (4), chapter 259–275. Springer, Berlin / Heidelberg.

[Dorigo and Di Caro, 1999] Dorigo, M. and Di Caro, G. (1999). Ant colony optimization : A new meta-heuristic. In *Proceedings of the 1999 Congress on Evolutionary Computation (CEC 99)*, volume 2, pages 1470–1477. IEEE Press.

[Dorigo et al., 1996] Dorigo, M., Maniezzo, V., and Colorni, A. (1996). The ant system : Optimization by a colony of cooperating agents. *IEEE Transactions on Systems, Man, and Cybernetics Part B : Cybernetics*, 26(1) :29–41.

[Dorigo and Stützle, 2004] Dorigo, M. and Stützle, T. (2004). *Ant Colony Optimization*. Bradford Books. MIT Press, Cambridge, USA.

[Dragicevic, 2004] Dragicevic, P. (2004). *Un modèle d'interaction en entrée pour des systèmes interactifs multi-dispositifs hautement configurables*. PhD thesis, Université de Nantes, Nantes.

[Dubnov et al., 2003] Dubnov, S., Assayag, G., Lartillot, O., and Bejerano, G. (2003). A system for computer music generation by learning and improvisation in a particular style. In *IEEE Computer*, volume 36, pages 73–80.

[Eck and Schmidhuber, 2002] Eck, D. and Schmidhuber, J. (2002). A first look at music composition using lstm recurrent neural networks. Technical Report IDSIA-07-02, IDSIA, www.idsia.ch/techrep.html.

[Eldridge, 2002] Eldridge, A. C. (2002). Adaptive systems music : Musical structures from algorithmic process. In Soddu, C., editor, *5th Generative Art Conference (GA2002)*, Milan, Italie.

[Eldridge, 2005] Eldridge, A. C. (2005). Fond punctions : Generative processes in live improvised performance. In *Proceedings of Generative Arts Practice*, pages 41–51.

[Fairweather and Trewin, 2009] Fairweather, P. and Trewin, S. (2009). Cognitive impairments and web 2.0. In Stephanidis, C., editor, *Universal Access in the Information Society*. Springer, Berlin / Heidelberg.

[Fédération APAJH de Haute-Marne, 2009] Fédération APAJH de Haute-Marne (2009). Festival art et handicap. http ://art.handicap.free.fr/. Site web.

[Gardner, 1970] Gardner, M. (1970). Mathematical games. the fantastic combinations of john conway's new solitaire game "life". *Scientific American*, 223 :120–123.

[Gartland-Jones, 2002] Gartland-Jones, A. (2002). Can a genetic algorithm think like a composer ? In Soddu, C., editor, *5th Generative Art Conference (GA2002)*, Milan, Italie.

[Gartland-Jones, 2003] Gartland-Jones, A. (2003). Musicblox : A real-time algorithmic composition system incorporating a distributed interactive genetic algorithm. In Raidl, G. R., Cagnoni, S., Cardalda, J. J., Corne, D. W., Gottlieb, J., Guillot, A., Hart, E., Johnson, C. G., Marchiori, E., Meyer, J.-A., and Middendorf, M., editors, *Applications of Evolutionary Computing*, volume 2611/2003 of *Lecture Notes in Computer Science*, pages 145–155, Berlin / Heidelberg. Springer.

[Gartland-Jones and Copley, 2003] Gartland-Jones, A. and Copley, P. (2003). The suitability of genetic algorithms for musical composition. *Contemporary Music Review*, 22(3) :43–55.

[Geem and Choi, 2007] Geem, Z. W. and Choi, J.-Y. (2007). Music composition using harmony search algorithm. In Giacobini, M., Brabazon, A., Cagoni, S., Di Caro, G. A., Drechsler, R., Farooq, M., Fink, A., Lutton, E., Machado, P., Minner, S., O'Neill, M., Romero, J., Rothlauf, F., Squillero, G., Takagi, H., Uyar, A. S., and Yang, S., editors, *Applications of Evolutionary Computing*, volume 4448 of *LNCS*, pages 593—-600, Berlin Heidelberg. EvoWorkshops, Springer-Verlag.

[Goldberg, 2002] Goldberg, D. E. (2002). *The design of Innovation : Lessons from and for Competent Genetic Algorithms*. Number 7 in Genetic algorithms and evolutionary computation. Kluwer Academic Publishers, Boston, Etats-unis.

[Greenfield, 2006] Greenfield, G. (2006). Ant paintings using a multiple pheromone model. In *7th International Conference on Short and Medium Span Bridges (Bridges 2006)*, Montréal, Québec, Canada.

[Greenfield, 2005] Greenfield, G. R. (2005). Evolutionary methods for ant colony paintings. In *EvoWorkshops*, pages 478–487.

[Guéret et al., 2004] Guéret, C., Monmarché, N., and Slimane, M. (2004). Ants can play music. In *Fourth International Workshop on Ant Colony Optimization and Swarm Intelligence (ANTS 2004)*, volume 3172 of *Lecture Notes in Computer Science*, pages 310–317, Université Libre de Bruxelles, Belgique. Springer-Verlag.

[Henry and the Education & Outreach Working Group (EOWG), 2005] Henry, S. L. and the Education & Outreach Working Group (EOWG) (2005). Introduction to web accessibility. W3C's web site.

[Henz et al., 1996] Henz, M., Lauer, S., and Zimmermann, D. (1996). COMPOzE — intention-based music composition through constraint programming. In *Proceedings of the 8th IEEE International Conference on Tools with Artificial Intelligence (ICTAI 1996)*, Tools with Artificial Intelligence, pages 118–121, Toulouse, France. IEEE Computer Society Press.

[Hiller and Isaacson, 1959] Hiller, L. and Isaacson, L. (1959). *Experimental music : Composition with an electronic computer*. McGraw-Hill, New York. réimprimé en 1979 par Greenwood Press.

[Holland, 1975] Holland, J. H. (1975). *Adaptation in natural and artificial systems*. MIT Press, Cambridge. réimprimé en 1992.

[Kemeny and Snell, 1960] Kemeny, J. G. and Snell, J. L. (1960). *Finite Markov chains*. Van Nostrand, Princeton.

[Lambert, 1977] Lambert, J.-C. (1977). *Yvaral*. Le territoire de l'œil. éditions Horay, Paris, France.

[Law et al., 2008a] Law, C. M., Yi, J. S., Choi, Y. S., and Jacko, J. A. (2008a). A systematic examination of universal design resources : part 1, heuristic evaluation. In Stephanidis, C., editor, *Universal Access in the Information Society*, volume 7 (1–2), pages 31–54. Springer, Berlin / Heidelberg.

[Law et al., 2008b] Law, C. M., Yi, J. S., Choi, Y. S., and Jacko, J. A. (2008b). A systematic examination of universal design resources : part 2, analysis of the development process. In Stephanidis, C., editor, *Universal Access in the Information Society*, volume 7 (1–2), pages 55–77. Springer, Berlin / Heidelberg.

[Leleu and Decroupet, 2006] Leleu, J.-L. and Decroupet, P., editors (2006). *Pierre Boulez. Techniques d'écriture et enjeux esthétiques*. Contrechamps Editions, Genève, Suisse.

[Lettvin et al., 1959] Lettvin, J. Y., Maturana, H. R., McCulloch, W. S., and Pitts, W. H. (1959). What the frog's eye tells the frog's brain. In *Proceedings of the Institute of Radio Engineers*, volume 47, pages 1940–51.

[Lintermann and Deussen, 1999] Lintermann, B. and Deussen, O. (1999). Interactive modeling of plants. In *IEEE Computer Graphics*, volume 19, pages 55–65. IEEE Computer Society Press.

[Lorrain, 2003] Lorrain, D. (2003). Réalisation de jeux musicaux du xviiie siècle : Mozart & stadler. In *10ème Journée d'informatique musicale*.

[Loy, 1985] Loy, G. (1985). Musicians make a standard : The midi phenomenon. *Computer Music Journal*, 9(4) :181–198.

[Lumer and Faieta, 1994] Lumer, E. D. and Faieta, B. (1994). Diversity and adaptation in populations of clustering ants. In *Proceedings of the third international conference on Simulation of adaptive behavior : from animals to animats*, pages 501–508, Brighton, United Kingdom. MIT Press.

[Lutton, 1999] Lutton, E. (1999). *Algorithmes Génétiques et Fractales*. Dossier d'habilitation à diriger des recherches, Université Paris XI Orsay.

[Lutton, 2005] Lutton, E. (2005). Darwinisme artificiel : une vue d'ensemble. *Revue Technique et Science Informatique, TSI, Traitement du Signal, numéro spécial "Méthodologie de la gestion intelligente des senseurs"*, 22(4) :339–354.

[Lutton, 2006] Lutton, E. (2006). Evolution of fractal shapes for artists and designers. *International Journal of Artificial Intelligence Tools (IJAIT)*, 15(4) :651–672. Special Issue on AI in Music and Art.

[Mackenzie et al., 1991] Mackenzie, S., Seller, A., and Buxton, W. (1991). A comparison of input devices in element pointing and dragging tasks. In Robertson, S. P., Olson, G. M., and Olson, J. S., editors, *Proceedings of the SIGCHI conference on Human factors in computing systems : Reaching through technology*, pages 161–166, New Orleans, Louisiana, États-unis. ACM.

[Maniezzo, 1999] Maniezzo, V. (1999). *INFORMS Journal on Computing*, volume 11, chapter Exact and approximate nondeterministic tree-search procedures for the quadratic assignment problem, pages 358–369. HighWire Press, Stanford, Etats-unis.

[Markov, 1913] Markov, A. A. (1913). An example of statistical investigation in the text of "eugene onegin" illustrating coupling of "tests" in chains. In *Proceedings of Academic Scientific St. Petersburg*, volume VI, pages 153–162, Saint Petersburg, Russie.

[McCormack, 1996] McCormack, J. (1996). Grammar based music composition. In Stocker, R., Jelinck, H., B., B., and Bossomaier, T., editors, *Complex Systems 96 : From Local Interactions to Global Phenomena*, Complex Systems, pages 321 – 333. IOS Press, Nieuwe Hemweg 6B, 1013 BG Amsterdam, The Netherlands.

[McCormack, 2003] McCormack, J. (2003). A method for generating phyliotaxis over surfaces of revolution. Technical Report No. 2003/149, Monash University, Melbourne.

[Ministère de la culture et de la communication, 2010] Ministère de la culture et de la communication (2010). Culture et handicap. http ://www.culture.gouv.fr/handicap/. Site web.

[Minsky and Papert, 1987] Minsky, M. L. and Papert, S. A. (1987). *Perceptrons - Expanded Edition*. MIT Press, Cambridge, MA.

[Miranda, 2007] Miranda, E. R. (2007). Cellular automata music : From sound synthesis to musical forms. In Miranda, E. R. and Biles, J. A., editors, *Evolutionary Computer Music*, volume 8, pages 170 – 193. Springer, London.

[Monmarché, 2000] Monmarché, N. (2000). *Algorithmes de fourmis artificielles : applications à la classification et à l'optimisation*. Thèse de doctorat, Laboratoire d'Informatique, Université de Tours.

[Monmarché et al., 1998] Monmarché, N., Desbarats, L., Slimane, M., and Venturini, G. (1998). Etude d'un nouvel algorithme d'optimisation et d'apprentissage inspiré d'une colonie de fourmis *Pachycondyla apicalis*. In *Colloque Intelligence Artificielle et Complexité*, pages 114–119, Paris, France.

[Monmarché et al., 2009] Monmarché, N., Guinand, F., and Siarry, P., editors (2009). *Fourmis Artificielles, des bases de l'optimisation aux applications industrielles*, volume 1 of *Traité IC2*. Hermès-Lavoisier.

[Monmarché et al., 2007] Monmarché, N., Mahnich, I., and Slimane, M. (2007). Artificial art made by artificial ants. In Machado, P. and Romero, J., editors, *The Art of Artificial Evolution*, Natural Computing Series, page 227–247. Springer, Berlin/Heidelberg.

[Monmarché et al., 2003] Monmarché, N., Slimane, M., and Venturini, G. (2003). Fourmis artistiques ou l'art artificiel pictural et musical. In *Hypertextes et Hypermédias, Réalisations, Outils & Méthodes (H2PTM03)*, pages 109–118, Paris, France. Hermes.

[Monmarché and Clair, 2009] Monmarché, N. and Clair, R. (2009). Des fourmis artificielles pour l'art artificiel. In Monmarché, N., Guinand, F., and Siarry, P., editors, *Fourmis Artificielles, des bases algorithmiques aux concepts et réalisations avancés*, Traité IC2. Hermès-Lavoisier.

[Moura and Pereira, 2004] Moura, L. and Pereira, H. G. (2004). *Man + Robots : Symbiotic Art*. Institut d'Art Contemporain de Villeurbanne.

[Mozart, 1793] Mozart, W. A. (1793). *Instruction pour composer autant de Walzes ou Schleiser que l'on veut par le moyen de deux Dez sans savoir la musique ou la composition*. Simrock, N., Bonn, Allemagne.

[Mozer, 1994] Mozer, M. C. (1994). Neural network music composition by prediction : Exploring the benefits of psychoacoustic constraints and multiscale processing. *Connection Science*.

[Nguyen et al., 2006] Nguyen, Q., Novakowski, S., Boyd, J. E., Jacob, C., and Hushlak, G. (2006). Motion swarms : video interaction for art in complex environments. In *proceedings of ACM Multimedia 06*, Santa Barbara.

[Nierhaus, 2009] Nierhaus, G. (2009). *Algorithmic composition*. SpringerWienNewYork, Wien.

[ONU, 2010] ONU (2010). Faits et chiffres sur le handicap. `http://www.un.org/french/disabilities/default.asp?navid=35&pid=833`.

[Ortega de la Puente et al., 2002] Ortega de la Puente, A., Alfonso, R. S., and Moreno, M. A. (2002). Automatic composition of music by means of grammatical evolution. In *Proceedings of the 2002 conference on APL : array processing languages : lore, problems, and applications*, pages 148 – 155, Madrid. ACM.

[Pagliarini and Hautop Lund, 2002] Pagliarini, L. and Hautop Lund, H. (2002). The musical glove. In Bilotta, E., Groß, D., Smith, T., Lenaerts, T., Bullock, S., Hautop Lund, H., Bird, J., Watson, R., Pantano, P., Pagliarini, L., Abbass, H., Russell, S., and Bedau, M., editors, *ALifeVIII : Workshop Proceedings*, Sydney, Australie.

[Papadopoulos and Wiggins, 1999] Papadopoulos, G. and Wiggins, G. (1999). Ai methods for algorithmic composition : A survey, a critical view, and future prospects. In *Proceedings of the AISB'99 Symposium on Musical Creativity*.

[Prusinkiewicz and Lindenmayer, 1990] Prusinkiewicz, P. and Lindenmayer, A. (1990). *The Algorithmic Beauty of Plants*. Springer-Verlag.

[Raynal, 2005] Raynal, M. (2005). Optimisation des systèmes de saisie de texte pour un meilleur usage par les personnes handicapées moteur. In *17ème conférence francophones sur l'interaction Homme-Machine*, pages 9–10.

[Roads, 2004] Roads, C. (2004). *Microsound*. MIT Press, Cambridge, MA.

[Rosenblatt, 1958] Rosenblatt, F. (1958). The perceptron : A probabilistic model for information storage and organization in the brain. *Psychological Review*, 65(6) :386–408.

[Réseau National Musique et Handicap, 2010] Réseau National Musique et Handicap (2010). Charte musique et handicap. http ://www.musique-handicap.fr/. Site web.

[Schockaert et al., 2004] Schockaert, S., De Cock, M., Cornelis, C., and Kerre, E. E. (2004). Efficient clustering with fuzzy ants. In *Applied Computational Intelligence*, pages 195–200. World Scientific.

[Sebag and Schoenauer, 1996] Sebag, M. and Schoenauer, M. (1996). Contrôle d'un algorithme génétique. In *Revue d'intelligence artificielle*, volume 10, pages 389–428. Lavoisier, Paris, FRANCE.

[Sepchat et al., 2007] Sepchat, A., Clair, R., Monmarché, N., and Slimane, M. (2007). Jeux vidéo accessibles et fourmis artificielles. In *Première conférence internationale sur l'accessibilité et les systèmes de suppléance aux personnes en situations de handicaps (ASSISTH'2007)*, Toulouse, France. Cépaduès édition.

[Sepchat et al., 2008a] Sepchat, A., Clair, R., Monmarché, N., and Slimane, M. (2008a). Artificial ants and accessible games for impaired players. In *International conference on metaheuristics and nature inspired computing (META'08)*, Hammamet, Tunisia.

[Sepchat et al., 2008b] Sepchat, A., Clair, R., Monmarché, N., and Slimane, M. (2008b). Artificial ants and dynamical adaptation of accessible games level. In *11th International Conference on Computers Helping People with special needs (ICCHP'08)*, volume 5105 of *Lecture Notes in Computer Science*, page 593–600. University of Linz, Austria, Springer Berlin / Heidelberg.

[Sepchat et al., 2008c] Sepchat, A., Clair, R., Monmarché, N., and Slimane, M. (2008c). Task division in ants for better game engines : a contribution to game accessibility for impaired players. In Rudolph, G., Jansen, T., Lucas, S. M., Poloni, C., and Beume, N., editors, *Parallel Problem Solving from Nature - PPSN X*, volume 5199 of *Lecture Notes in Computer Science*, page 961–970, Dortmund, Germany. Springer Berlin / Heidelberg.

[Sepchat et al., 2009] Sepchat, A., Colas, S., Clair, Romain Monmarché, N., Gaucher, P., and Slimane, M. (2009). Les fourmis artificielles et le handicap : compensation de déficits d'autonomie et jeux pour déficients visuels. In Monmarché, N., Guinand, F., and Siarry, P., editors, *Fourmis Artificielles, Des bases algorithmiques aux concepts et réalisations avancés*, Traité IC2. Hermès-Lavoisier.

[Sepchat et al., 2008d] Sepchat, A., Descarpentries, S., Monmarché, N., and Slimane, M. (2008d). Digital audio player and audio games : an alternative to portable console for visually impaired players. *AMSE Journal*.

[Sepchat et al., 2006] Sepchat, A., Monmarché, N., Slimane, M., and Archambault, D. (2006). Jeux vidéo tactiles pour enfants non voyants. In *4-ème Conférence Handicap*, pages 303–308, Paris, Porte de Versailles.

[Shannon, 1948] Shannon, C. E. (1948). A mathematical theory of communication. In *The Bell System Technical Journal*, volume 27, pages 379–423, 623–656. American telephone and telegraph Co., États-unis.

[Stützle and Hoos, 2000] Stützle, T. and Hoos, H. H. (2000). *MAX-MIN Ant System*, volume 16 of *Future Generation Computer Systems*, pages 889–914. Elsevier, elsevier edition.

[Takagi, 1998] Takagi, H. (1998). Interactive evolutionary computation : System optimization based on human subjective evaluation. In *IEEE International Conference on Intelligent Engineering Systems (INES'98)*, pages 1–6, Vienna, Austria.

[The MIDI Manufacturers Association (MMA), 2010] The MIDI Manufacturers Association (MMA) (2010). Specifications midi. http ://www.midi.org/. Site web.

[Todd and Werner, 1999] Todd, P. M. and Werner, G. M. (1999). Frankensteinian methods for evolutionary music composition. In *Musical networks*, pages 313 – 339. MIT Press, Cambridge.

[Tzafestas, 2000] Tzafestas, E. S. (2000). Integrating drawing tools with behavioral modeling in digital painting. In *Proceedings of the 1st International Workshop "Bridging the gap : Bringing together new media artists and multimedia technologists"*, Los Angeles.

[Unemi and Nakada, 2001] Unemi, T. and Nakada, E. (2001). A tool for composing short music pieces by means of breeding. In *IEEE International Conference on Systems, Man, and Cybernetics*, volume 5, pages 3458 – 3463, Tucson, USA.

[Urbano, 2007] Urbano, P. (2007). Mimetic variations on stigmergic swarm paintings. In Monmarché, N., Talbi, E.-G., Collet, P., Schoenauer, M., and Lutton, E., editors, *Artificial Evolution, 8th International Conference, Evolution Artificielle (EA 2007)*, volume 4926 of *Lecture Notes in Computer Science*, pages 62–72, Tours, France. Springer.

[Vandenbroucke, 2000] Vandenbroucke, N. (2000). *Segmentation d'images couleur par classification de pixels dans des espaces d'attributs colorimétriques adaptés. Application à l'analyse d'images de football.* PhD thesis, UNIVERSITÉ DE LILLE 1, Lille, France.

[Ventrella, 2008] Ventrella, J. J. (2008). Evolving the mandelbrot set to imitate figurative art. In Hingston, P. F., Barone, L. C., and Michalewicz, Z., editors, *Design by Evolution*, Natural Computing Series, pages 145–167. Springer, Berlin Heidelberg.

[Von Neumann, 1966] Von Neumann, J. (1966). *Theory of Self-Reproducing Automata*. University of Illinois Press, Champaign, IL, USA, burks, arthur w. edition.

[Waschka II, 2007] Waschka II, R. (2007). Composing with genetic algorithms : Gendash. In Miranda, E. R. and Biles, J. A., editors, *Evolutionary Computer Music*, chapter 6, pages 117 – 136. Springer, London.

[Weiss, 1999] Weiss, G., editor (1999). *A Modern Approach to Distributed Artificial Intelligence*. MIT Press, Cambridge.

[Wooldridge, 1992] Wooldridge, M. (1992). *The Logical Modelling of Computational Multi-Agent Systems*. PhD thesis, Manchester Metropolitan University, Manchester.

[Wulfhorst et al., 2001] Wulfhorst, R. D., Flores, L. V., Flores, L. N. C. D., Alvares, L. O. C., and Vicari, R. M. (2001). An open architecture for a musical multi-agent system. In Ramalho, G. and Aragão, F., editors, *VIII Brazilian Symposium on Computer Music*, Fortaleza, Brésil.

[Xenakis, 1981] Xenakis, I. (1981). *Musiques formelles*. Stock.

[Zhang and Miranda, 2007] Zhang, Q. and Miranda, E. R. (2007). Experiments in generative musical performance with a genetic algorithm. In Miranda, E. R. and Biles, J. A., editors, *Evolutionary Computer Music*, number 2 in Computer Science, pages 100–116. Springer, Londres.

[Özcan and Erçal, 2007] Özcan, E. and Erçal, T. (2007). A genetic algorithm for generating improvised music. In *Evolution Artificielle (EA'07)*.

Résumé :

Partant d'une étude de méthodes algorithmiques de création artistique automatique, nos travaux s'intéressent à des adaptations de ces méthodes pour proposer des formes de création artistique accessibles centrées sur la musique et le dessin par ordinateur.

Le développement d'outils informatiques accessibles impose certaines contraintes. Elles servent de cadre au développement de deux outils qui illustrent la mise en œuvre de ces contraintes, associées à l'adaptation de systèmes de production artistique automatique, notamment basés sur des algorithmes de colonie de fourmis artificielles. Le premier programme est un instrument de musique virtuel permettant au plus grand nombre de jouer de la musique, et fournissant un accompagnement automatique. Le second est un atelier de dessin où des outils basés sur des méthodes génératives offrent un résultat complexe à partir d'actions très simples. Cette thèse détaille le développement de ces deux programmes ainsi que leur évaluation, sur le terrain, à la rencontre d'utilisateurs réels.

Mots clés :

Accessibilité, Handicap, Art génératif, Algorithmes de colonie de fourmis artificielles, Algorithmes évolutionnaires, Musique, Dessin

Abstract :

Starting from a survey of algorithimic methods for automatic artistic creation, our works deals with adaptations of these methods to provide accessible art creation forms, focusing on music and computer drawing.

the development of accessible computer software requires some constraints. Following these constraints and using adapted automatic art production systems, notably based on artificial ant colony algorithms, we developed two computer programs. The first one is a virtual music instrument, allowing most people to play music and providing and automatic accomapniement. The second one is a drawing workshop with generative methods-based tools provide complex results from simple actions. This PhD thesis details the development of this two programs and their evaluations, with real users meetings.

Keywords :

Accessibility, Handicap, Generative art, Artificial ants colony algorithms, Evolutionnary computation, Music, Drawing