Entre Calcul, Programmation et Création

Yann Orlarey

Yann ORLAREY

Entre Calcul, Programmation et Création

Introduction

L'objectif de ce texte est d'essayer de répondre à la question suivante : qu'est-ce qui fait qu'un ordinateur et un langage de programmation deviennent des outils d'aide à la création ?

Quand nous disons « Aide à la création », cela veut dire que nous faisons bien une distinction avec l'ordinateur comme outil de production. Nous ne nous intéressons pas au fait qu'on ait déjà inventé la chose et qu'on se serve de l'ordinateur pour la réaliser. Nous ne nous intéressons pas non plus ici à l'ordinateur comme outil de création automatique, de composition automatique. La question qui nous intéresse, c'est la relation créative qu'un compositeur va avoir avec l'ordinateur. Qu'est-ce qui fait que l'ordinateur va être un aide à la création ? C'est un peu la question à laquelle nous allons essayer de répondre.

Nous allons être amenés à rappeler ce qu'est un ordinateur, une sorte de machine universelle programmable. Nous allons être amenés à rappeler ce qu'est un langage de programmation, le vocabulaire qui nous permet d'instruire cette machine univer-

selle programmable pour qu'elle fasse ce qu'on veut lui faire faire. Nous allons être amenés à rappeler également les types de calculs spécifiquement musicaux qu'on est amené à faire faire à un ordinateur, à essayer aussi de théoriser en quelque sorte l'acte de création en tant que tel et ensuite à essayer de réunir tous les morceaux pour tenter de comprendre en quoi l'ordinateur devient un outil d'aide à la création.

Tous ceux qui ont une expérience de ces logiciels de composition musicale, de ces langages pour la composition musicale, ont une compréhension en fait très fine de l'apport créatif qu'amène ce type d'outil, comme on peut le constater dans les autres textes de cet ouvrage par exemple.

Nous allons essayer de donner des mots, des définitions sur ces différentes choses-là.

Nous serons amenés à parler de *programmation créative*, à savoir l'utilisation de la programmation non pour écrire un programme en tant que tel, comme le ferait un informaticien, mais comme le font beaucoup d'artistes pour décrire des objets artistiques, musicaux par exemple mais aussi graphiques, des systèmes d'interactions. Il y a là un apport spécifique de l'ordinateur que nous aimerions essayer d'expliciter.

Des Cailloux aux Ordinateurs

Pourquoi appelle-t-on calcul aussi bien des calculs rénaux que des calculs mathématiques ? Ce n'est pas parce que c'est douloureux dans les deux cas, s'il est vrai que les calculs mathématiques peuvent souvent être assez douloureux. C'est tout simplement parce que calcul dérive du latin *calculus* qui veut dire cailloux et que les cailloux sont en quelque sorte les lointains ancêtres des ordinateurs.

Nous allons commencer par dresser une sorte d'historique qui nous mène comme ça des cailloux aux ordinateurs. Il est en effet très important de comprendre la spécificité de ce que c'est qu'un ordinateur. Si on ne comprend pas la spécificité de ce que c'est

qu'un ordinateur, on ne peut pas comprendre le rapport créatif qu'on a avec l'informatique.

Il faut se replonger il y a cinq mille cinq cents ans[1] on pouvait alors exprimer les nombres dans le langage parlé, mais on ne savait pas les écrire. Néanmoins, il y avait des échanges commerciaux et beaucoup de situations qui nécessitaient de pouvoir comptabiliser les choses.

Une des techniques extrêmement astucieuse qui a été développée en Mésopotamie c'est l'utilisation de jetons et de bulles. La situation typique était la suivante : le propriétaire d'un troupeau de moutons par exemple qui allait confier à un berger son troupeau de moutons. Il fallait bien quelque part fixer en quelque sorte la quantité de bêtes qui étaient confiées pour vérifier que quand le berger revenait, on avait bien la même quantité de bêtes. Mais en même temps on n'avait pas les nombres, on ne savait pas les écrire, etc. Donc l'idée était d'établir une sorte de bijections, si l'on peut dire en termes mathématiques, de correspondance entre des objets, ici des jetons, des petits cailloux en quelque sorte, des billes en argile et les bêtes qu'on allait confier au berger et on allait enfermer ces billes dans une bulle aussi en argile sur lequel le comptable (il y avait des comptables) et le propriétaire allaient apposer un sceau. Quand le berger allait revenir, on allait pouvoir casser cette bulle pour vérifier qu'il y avait bien le bon nombre de bêtes à l'arrivée comme au départ. Ce qui est très intéressant c'est de voir que l'origine de l'écriture, provient d'un besoin de compter, de comptabilité et c'est la même origine pour l'informatique : aussi un besoin de compter et de comptabiliser.

Ce qui va se passer très rapidement, c'est qu'on va commencer à représenter sur la bulle, le contenu de la bulle (voir figure 1). On obtient donc sur la surface de la bulle, une représentation du contenu de la bulle. Cela permet de vérifier, sans avoir à casser la bulle, ce qu'il y a à l'intérieur. Petit à petit, on a aplati la bulle, en quelque sorte, et on est arrivé aux tablettes avec signes. Ce sont les débuts de l'écriture.

1. Georg Ifrah, *Histoire universelle des chiffres*, éd. Robert Lafont, 1994.

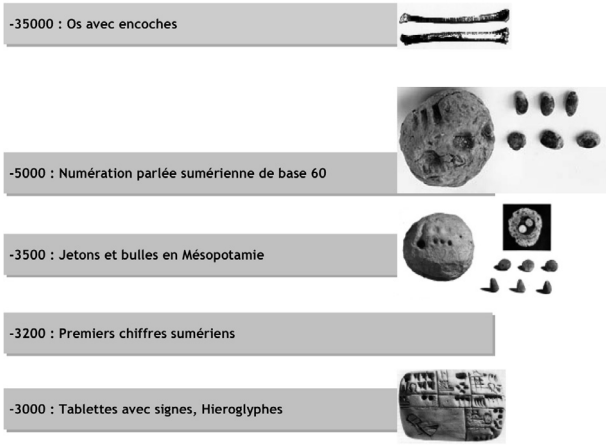

Figure 1 – Des cailloux aux signes.

Les besoins de comptabilité, de comptabiliser, font les débuts de l'écriture et l'écriture va ensuite connaître le succès que l'on sait. Elle va ensuite se diversifier. L'écriture permet d'écrire de la musique, d'écrire des romans, d'écrire de la science, d'écrire toutes sortes de choses. Elle s'est multimédiatisée en quelque sorte et l'informatique a suivi exactement le même chemin. Il y a donc un parallèle assez fort entre les origines, les besoins auxquels a répondu l'écriture et auxquels a répondu l'informatique, avec ce même parcours de diversification.

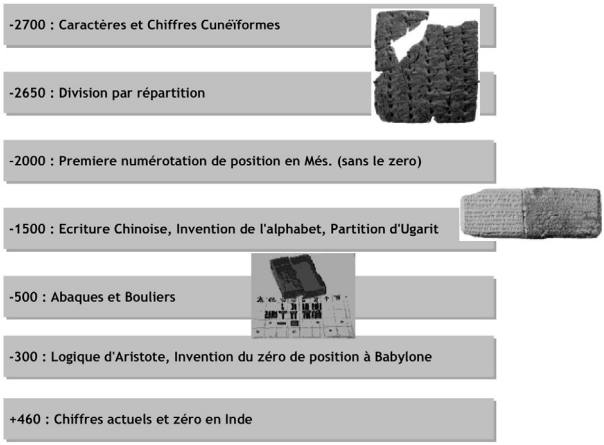

Figure 2 – Des signes aux nombres.

S'en suit tout un cheminement qui va amener à l'écriture des nombres tels qu'on les connaît. Il y a 4000 ans, l'invention de la numérotation de position est très importante, donc le fait, extraordinaire, qu'un signe puisse représenter des valeurs différentes suivant la position sur laquelle il est placé. Cela va conduire, il y a 1500 ans, à l'invention en Inde des chiffres indiens et du zéro, qui sont bien supérieurs aux chiffres romains dans la mesure où ils permettent de faire des tas d'opérations que ne permettaient pas les chiffres romains.

Figure 3 – L'introduction des chiffres arabes en Occident.

Nous pouvons noter, entre parenthèses, une chose très intéressante à savoir que la même chose représentée de deux manières différentes peut avoir un impact très différent. Les chiffres romains représentent des nombres, mais ils ne sont pas opératoires. On ne peut pas faire de calcul, on ne peut pas faire de multiplication, on ne peut pas faire d'addition avec, alors que les chiffres, appelons-les indo-arabes, permettent justement de faire des opérations. Donc un simple changement, non pas de sémantique mais de syntaxe, va avoir un effet dramatique sur les possibilités que l'on a de calculs. Cela nous permet de dire que

dans le domaine des langages de programmation, la syntaxe, la représentation des choses est très importante et va en quelque sorte conférer au langage de programmation une ergonomie qui va nous permettre de penser un certain nombre de choses, de phénomènes. Donc des choses équivalentes mais une représentation différente peut avoir un gros impact sur l'utilisation qu'on va en faire.

En Occident, on a été « assez mauvais » pendant très longtemps, puisqu'on est resté avec les chiffres romains. Il y avait donc des calculateurs, qui n'étaient pas des ordinateurs, ni des machines, mais des gens qui savaient calculer, qui utilisaient des abaques, des bouliers, etc. pour faire des calculs[2]. Il faut attendre l'abandon des chiffres romains, il y a environ 500 ans, pour qu'on adopte en occident les chiffres indo-arabes.

Dans l'histoire qui conduit à l'avènement de l'ordinateur, il y a en quelque sorte trois phases : la première phase correspond à l'introduction des chiffres et des méthodes de calcul, ensuite vient la mécanisation de ces méthodes de calcul puis c'est véritablement l'apparition de l'ordinateur, qui est différent du calculateur mécanique. C'est cette différence-là qui est tout à fait cruciale dans ce qui nous occupe aujourd'hui à savoir l'utilisation créative de l'informatique.

Nous pouvons citer un certain nombre d'étapes : la fameuse Pascaline de Blaise Pascal qui était capable de faire des additions, qui était une machine à calculer. Des machines à calculer ont ainsi existé bien avant les ordinateurs. Mais il y a une différence fondamentale entre une machine à calculer et un ordinateur, c'est que l'ordinateur est une machine *universelle*. Elle peut, par une simple reprogrammation en quelque sorte, faire toutes sortes de calculs. Donc les machines à calculer ont préexisté à l'ordinateur, mais quand on voulait faire des calculs différents, il fallait reconstruire physiquement une nouvelle machine. L'intérêt, l'in-

2. Montaigne ne savait pas compter « Or, je ne sçay compter ny a get ni a plume ». Voir Georg Ifrah, *Histoire universelle des chiffres*, éd. Robert Lafont, 1994.

vention de l'ordinateur, c'est qu'une seule construction physique est suffisante et après on se contente d'une reconstruction logique du calcul que l'on veut faire[3]. C'est ça l'invention fondamentale de l'ordinateur. Mais pour en arriver là, il a fallu non seulement faire des machines qui faisaient des calculs comme des additions mais en réalité, beaucoup de calculs sont une suite d'enchaînements d'opérations. Il a donc fallu inventer des machines programmables. Parmi les premières machines programmables, il y a eu des machines musicales que sont les orgues de Barbarie, ce qui nous permet de dire, entre parenthèses, que la préoccupation musicale est souvent très intéressante et qu'elle peut conduire à inventer des choses qui sont d'un usage beaucoup plus général.

Les premières machines programmables, dont on contrôle le fonctionnement, sont donc en fait des machines musicales, par exemple l'orgue de Barbarie. Très rapidement aussi apparaissent les automates, comme les automates de Vaucanson. Ainsi des techniques ont été développées pour permettre de contrôler le fonctionnement d'une machine par un programme. Cela va donner ensuite par exemple les cartes perforées et le métier à tisser. Cela conduit également jusqu'à l'ordinateur qui est une machine programmable dont on va contrôler le fonctionnement mais où en plus on va internaliser le programme, ce qui va permettre à l'ordinateur de calculer son propre programme. C'est là que, quand on va avoir réalisé toutes ces opérations, à savoir pouvoir mécaniser le calcul, automatiser l'enchaînement des calculs et internaliser l'enchaînement des calculs qu'on va véritablement avoir un ordinateur.

On va sauter un certain nombre d'étapes pour arriver à l'ordinateur.

3. À propos de la programmation Alan Turing parle d'un « travail de bureau », qui remplace la construction physique d'une machine spécialisée. Voir Andrew Hodges *Alan Turing ou l'énigme de l'intelligence*, éd. Payot, 2004.

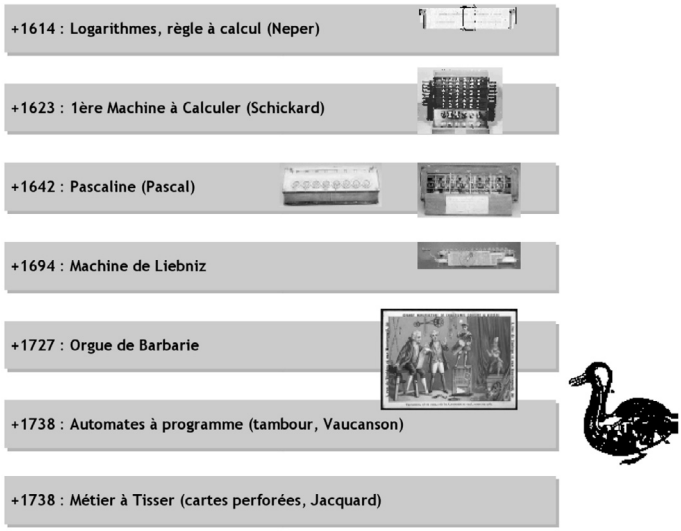

Figure 4 – Des premières machines à calculer aux premiers automates.

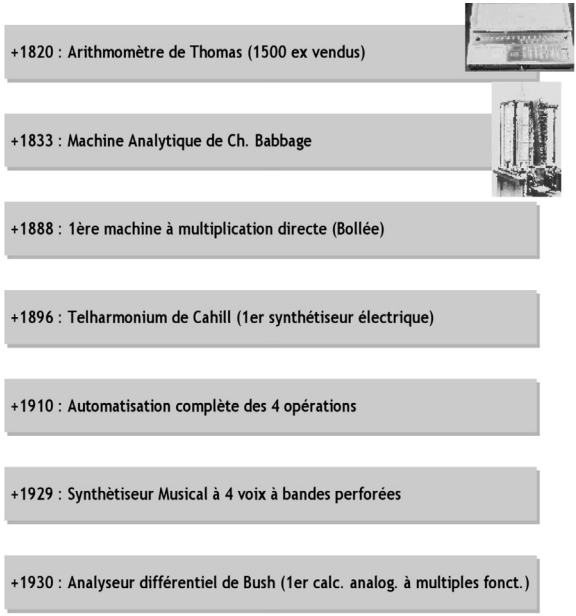

Figure 5 – Vers l'automatisation des calculs.

On cite souvent l'ENIAC comme étant le premier ordinateur. En vérité, ce n'était pas vraiment un ordinateur. C'est le premier calculateur électronique programmable, mais il était programmable par des systèmes de patchs en quelque sorte, un peu comme un synthétiseur analogique, c'est-à-dire qu'on devait câbler et décâbler des fils donc la programmation du calculateur n'était pas sous le contrôle du calculateur. Il n'y avait pas encore cette internalisation qui allait vraiment rendre la machine autocontrolable en quelque sorte. C'était un calculateur électronique qui allait mille fois plus vite que les autres, mais qui faisait quatre-vingts pour cent d'erreurs ![4]

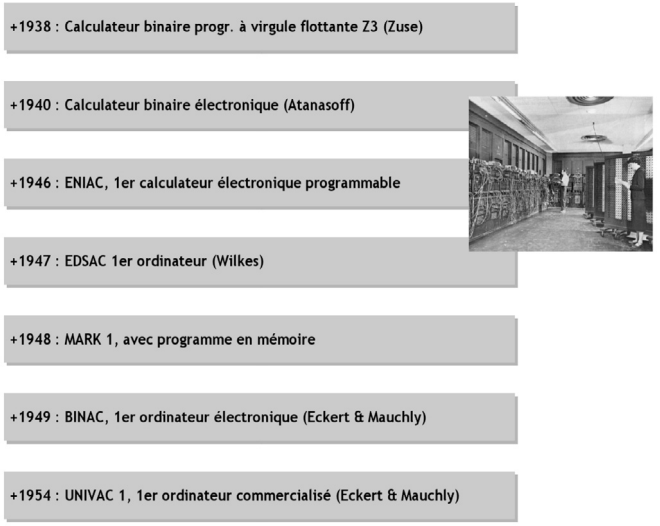

Figure 6 – L'apparition de l'ordinateur.

En réalité, le premier ordinateur commercialisé, en 1951, c'est l'UNIVAC et on est très proche des débuts de l'informatique musicale, en 1956. La musique a été la première forme d'art à véritablement s'intéresser aux possibilités de l'informatique, très rapi-

4. Les concepteurs disaient : ce n'est pas grave, de toute façon on refait les calculs, s'il y a concordance des résultats, on est à peu près tranquilles, on sait que c'est le bon résultat. Voir J. Peres, L. Brillouin et L. Couffigna, *Les Grandes Machines mathématiques : les machines américaines*, Paris, 1948.

dement. À l'époque, il fallait vraiment être visionnaire pour avoir l'idée d'utiliser ce genre de machine pour faire de la musique.

L'ordinateur : machine à calculer universelle

Un ordinateur n'est pas simplement une machine à calculer, un calculateur, c'est une machine à calculer universelle (en tout cas une approximation d'une machine à calculer universelle) c'est-à-dire qu'on va pouvoir l'instruire par le biais d'un logiciel des choses qu'on veut lui faire faire. Donc on va prendre une même structure matérielle et on va pouvoir la transformer en machine à écrire, en orgue, etc.

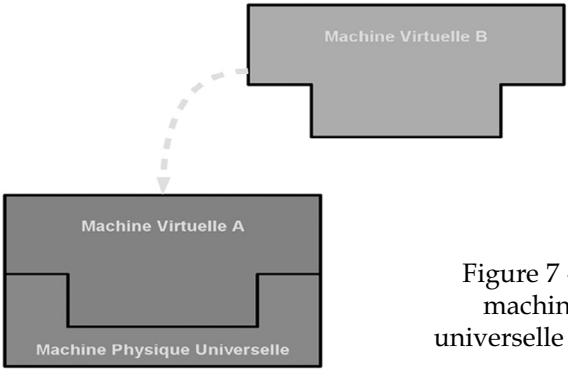

Figure 7 – Ordinateur, machine à calculer universelle programmable.

Par simple remplacement de la partie virtuelle on va pouvoir reconfigurer ainsi cette machine universelle. En réalité, dans un ordinateur, il y a un empilement de couches, c'est-à-dire qu'à l'intérieur de la machine universelle physique, on va mettre et empiler des machines universelles logicielles. Ce qu'on appelle un *système d'exploitation,* par exemple, a exactement ce rôle-là, à savoir de nous présenter en quelque sorte une machine universelle idéale, qu'on va retrouver d'un ordinateur sur l'autre, alors que physiquement ce ne sont pas exactement les mêmes ordinateurs. Tout ce qui se passe autour des machines virtuelles actuellement est exactement dans cette ligne-là. On a en quelque sorte un système de poupées russes, d'empilement de couches de logiciels qui donnent cette sorte de flexibilité.

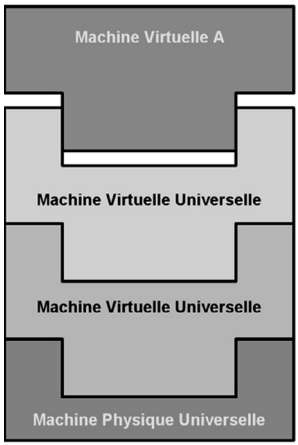

Figure 8 – L'ordinateur
et son système d'exploitation :
empilement de machines virtuelles.

Un ordinateur nous donne l'illusion d'utiliser plusieurs machines (logiciels) simultanément.

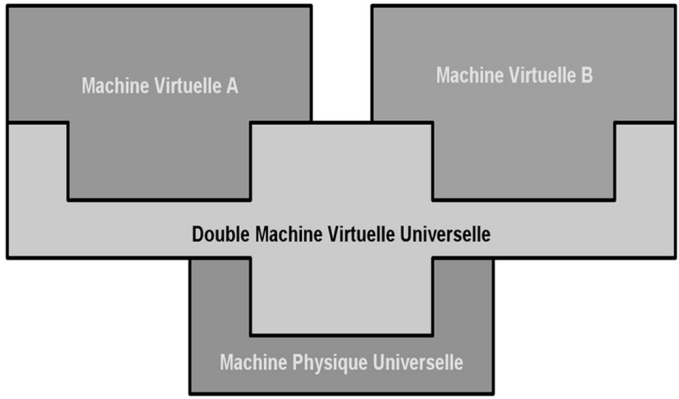

Figure 9 – Une machine virtuelle peut donner l'illusion
de multiples machines (système multi-tâches).

Ce qui se développe aussi, c'est par exemple ce qu'on appelle le *grid-computing*, le fait d'utiliser de nombreuses machines physiques séparées, via des réseaux et via un logiciel qui nous donne l'impression d'utiliser une seule et même machine, mais beaucoup plus puissante.

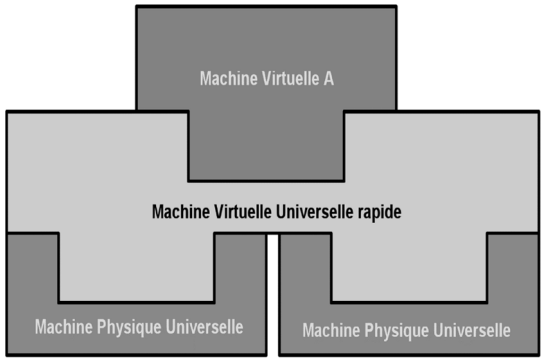

Figure 10 – De multiples machines réelles peuvent donner l'illusion d'une machine unique, mais plus puissante.

En fait, la tendance qui se dessine, c'est une sorte de déconnexion entre les machines physiques et les machines virtuelles. La révolution du parallélisme qui se déroule actuellement va nous conduire très rapidement à des machines qui comporteront des centaines, voir des milliers de coeurs de calcul, tout en donnant l'illusion d'une seule et même machine extrêmement puissante. Il va y avoir comme ça de plus en plus cette déconnexion entre le substrat physique sur lequel sont effectués les calculs et la représentation logique de ces calculs

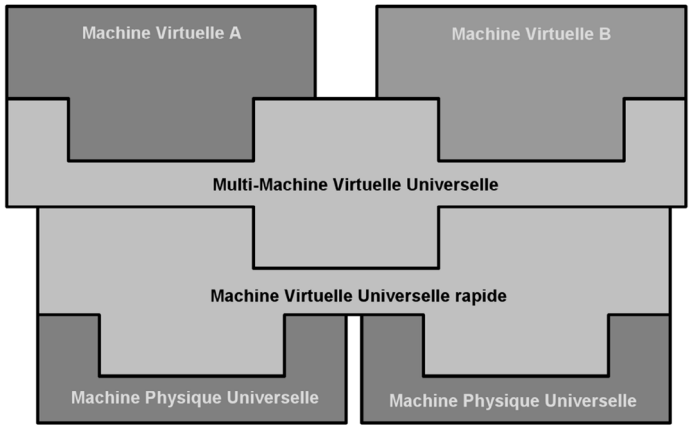

Figure 11 – Découplage entre machines physiques et machines virtuelles.

Programmation et Langages

L'évolution des langages de programmation a été un peu moins spectaculaire, mais tout aussi fondamentale. Un langage de programmation a une ergonomie qui fait qu'il est plus ou moins facile de « penser » un certain nombre de problèmes. À l'époque des premiers ordinateurs, le gros problème était de représenter les nombres qu'on voulait calculer. Les ordinateurs avaient un calcul binaire et la question était de savoir comment faire pour arriver à traduire des signes binaires de façon automatique. Petit à petit, nos besoins de traduction, de communication avec la machine ont évolué et il y a donc eu toute une série d'étapes qui conduisent du langage machine aux langages de programmation évolués (Open-Music par exemple est un langage graphique extrêmement évolué).

Nous pouvons rappeler quelques étapes. L'ordinateur apparaît véritablement en 1947. En 1840, Charles Babbages et Ada Byron avaient déjà en quelque sorte réalisé physiquement un ordinateur, qui n'avait pas vraiment marché, mais qui était en quelque sorte un prémisse d'ordinateur physique, mais surtout on considère qu'Ada Byron est la première programmeuse de l'histoire. Le premier informaticien est donc une femme[5]. Elle avait aussi envisagé, comme le rappelle Jean-Claude Risset[6] l'utilisation de l'ordinateur comme outil d'aide à la création dans la composition musicale.

Une étape décisive a été la formalisation du calcul par le mathématicien anglais Alan Turing, avec les fameuses machines de Turing[7], mais aussi par Alonso Church avec le Lambda-Calcul[8].

---

5. C'est en hommage à Ada Byron que le langage ADA qui existe depuis une vingtaine d'années porte ce nom.
6. Jean-Claude Risset, *Art, Science, Technologie (AST)*. Rapport de mission, mars 1998.
7. Alan Turing, « On computable numbers, with an application to the entscheidungsproblem », *Proceedings of the London Mathematical Society*, Series 2, 42, 1937.
8. Alonzo Church, *The Calculi of Lambda Conversion*, Princeton University Press, 1941.

+1951 : 1er Compilateur, A-0 (Grace Hopper, Univac I et II)

+1956 : IPL (Newell, Shaw, Simon) : IA, listes, récursivité, streams

+1957 : Fortran (Backus): sousroutines, branch. cond., boucles, variables

+1958 : Algol-58 Types et structures de données

+1959 : LISP (Mc Carthy) : Listes, récursivité, synthèse de programmes

+1960 : COBOL

+1960 : MUSIC III (Mathews) 1er lang. de synthèse du son

+1962 : Simula (Dahl, Nygaard) préfigure la prog. orientée objets

Figure 12 – L'apparition des premiers langages de programmation.

Figure 13 – Quelques langages musicaux.

Quand on parlait de calculateurs à l'époque, il s'agissait d'êtres humains, les calculateurs étaient des gens qui faisaient des opérations. On s'interrogeait pour savoir ce qu'ils pouvaient et ce qu'ils ne pouvaient pas faire. Il n'empêche que cette formalisation du calcul, c'est aussi la formalisation de la machine. D'une certaine manière, les hommes n'échappent pas aux limitations en terme de calcul, qui est le même que celui des machines.

Le premier compilateur, c'est-à-dire la première utilisation de l'ordinateur pour nous aider à programmer, date de 1951[9]. Le premier langage de programmation musicale, c'est Music III en 1960, l'utilisation de la programmation informatique pour programmer des sons.

On peut citer beaucoup d'exemples. La figure précédente présente une liste de quelques langages musicaux. Cette liste montre l'importance de l'enjeu qu'il y a de définir des langages de programmation, y compris des langages de programmation musicaux. C'est important et en même temps ce n'est pas satisfaisant. Le fait qu'il y en ait autant, cela veut dire que, d'une certaine manière, les nouveaux inventeurs de langages ne sont pas satisfaits de ce qui existait et veulent en quelque sorte poursuivre la chose parce que, encore une fois, un langage de programmation, c'est quelque chose qui a des dimensions ergonomiques extrêmement fortes. Alors les chercheurs qui s'intéressent un peu à la psychologie de la programmation, aux dimensions cognitives des langages de programmation, les catégorisent en six dimensions (voir figure 14).

Figure 14 – Dimensions cognitives des langages de programmation.

9. Grace Murray Hopper, *The education of a computer*, Remington Rand, 1952.

La capacité d'abstraction, c'est la capacité à généraliser. Abstraction (*abs-trahere* en latin), c'est séparer en quelque sorte l'essentiel du détail. Donc, dans les langages de programmation, on voit, si on étudie l'évolution des langages de programmation, cette capacité à abstraire de plus en plus. Par exemple entre le langage C++ et le langage C, il y a eu pas simplement la programmation objet mais il y a tout le système de *template* qui va permettre d'abstraire les types des paramètres et pas simplement les paramètres eux-mêmes. Il y a une marche comme cela vers une abstraction de plus en plus grande. L'abstraction, c'est la possibilité de généraliser.

La modularité, c'est la possibilité de décomposer un problème en sous-problèmes, pour ensuite recombiner les sous-solutions en solutions plus générales, donc la facilité, comme avec un lego, à combiner les choses entre elles pour faire l'objet qu'on veut.

La consistance, c'est aussi une dimension très importante. C'est le fait qu'il y ait des régularités dans le langage, qu'il n'y ait pas des exceptions, que si on a compris quelque chose, on puisse facilement le transvaser, en quelque sorte, l'adapter à autre chose, que des choses similaires aient des traitements similaires.

La lisibilité, c'est le fait que, non seulement on écrit des programmes, mais ensuite il faut les relire parce qu'on va vouloir les modifier. Il y a des programmes pour lesquels tout va bien quand vous les écrivez, mais si vous revenez trois jours après et que vous essayez de comprendre ce que vous avez fait, vous ne comprenez plus rien. La lisibilité est un élément très important dans le fait de pouvoir maintenir, modifier et améliorer les programmes.

L'expressivité correspond au coût pour exprimer quelque chose. Peut-on exprimer quelque chose de simple simplement ? C'est un peu le minimum qu'on peut demander. S'il faut exprimer des choses simples de manière compliquée, c'est qu'on a vraiment un très mauvais langage de programmation, mais est-ce que par hasard des choses un peu compliquées ne pourraient pas être ex-primées relativement simplement. Il y a donc toute une quête pour faire des langages extrêmement expressifs. Le

langage *Faust*[10] par exemple, à bien des égards est très expressif par rapport à d'autres langages. En général, un programme dans un langage expressif est plus court, plus compact que dans un langage moins expressif. Certains vont trouver que trop d'expressivité nuit à la lisibilité. Il y a des sortes de monnayages comme ça entre les différents aspects.

Enfin, la viscosité c'est la résistance au changement, le fait que, justement, entre l'expressivité et la lisibilité, on veuille pouvoir modifier des choses. Par exemple, des langages graphiques, du genre de *Max*[11], sont assez expressifs, assez faciles à programmer, mais par contre, quand on va vouloir faire des modifications si on a un patch très compliqué, cela peut être assez compliqué parce que justement il y a un manque de lisibilité, une sorte de difficulté à modifier les choses.

Avec ces langages, on va pouvoir exprimer des calculs musicaux. Nous allons essayer de faire une typologie des calculs musicaux qui correspondent à l'usage, même si c'est une typologie un peu arbitraire.

Typologie des calculs musicaux

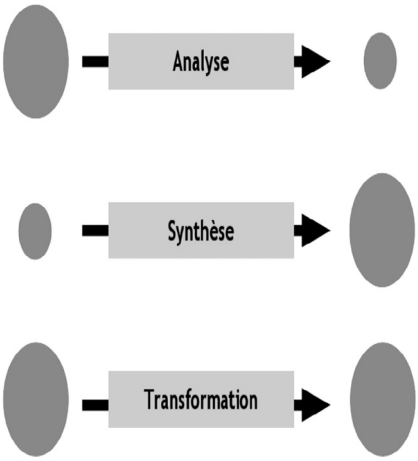

Figure 15 – Nature des calculs musicaux.

10. Éditeur GRAME.
11. Éditeur Cycling'74.

Un premier axe correspond à la nature des calculs. Typiquement, en musique, on va parler d'analyse, de synthèse et de transformation. Une façon de comprendre cette typologie, c'est de considérer le volume des données en entrée et en sortie de ces calculs.

L'analyse par exemple, c'est typiquement la situation où on a un son par exemple qu'on va essayer d'analyser, pour en reconnaître la hauteur. Donc on a beaucoup d'informations, des centaines de milliers d'échantillons en entrée et on va produire une seule valeur qui va être par exemple la hauteur du son. On est dans le cas de l'analyse. Mais cela peut s'appliquer aussi à autre chose que du son, comme par exemple à du geste ou se baser sur de l'analyse musicologique. On a beaucoup d'information et on essaye d'extraire en quelque sorte de l'information de ce tas d'informations.

La synthèse c'est en quelque sorte l'opération inverse. On a quelques paramètres, par exemple la hauteur d'un son ou le numéro de la touche, la vitesse avec laquelle on appuie sur la touche, on a un modèle de synthèse et on va produire un son qui va donc comporter des centaines de milliers d'échantillons.

La transformation correspond aux autres cas, c'est-à-dire qu'on a à peu près la même quantité d'informations en entrée et en sortie. Par exemple, on va prendre un son et lui appliquer un effet de spatialisation, une réverbération, etc. Il y a beaucoup de transformations qui sont des combinaisons d'analyse-resynthèse (comme on peut le voir dans les autres textes de cet ouvrage). On va donc prendre beaucoup d'informations, en extraire quelques informations, éventuellement transformer ces quelques informations, les injecter dans un modèle de synthèse pour produire beaucoup d'informations. Cela fait souvent des transformations très intéressantes parce qu'elles sont un peu moins immédiates que les transformations directes du signal.

Ensuite, on peut aussi s'intéresser à la nature de l'information que l'on traite. Là aussi on peut diviser ça, de façon un peu arbitraire, en trois catégories. Il y a l'information sonore, qui est très proche du phénomène sonore lui-même, le signal numérique en quelque sorte, l'information gestuelle ou performative qui est liée

par exemple à l'exécution de la pièce, l'interaction, les capteurs, le MIDI etc., c'est une sorte d'information intermédiaire, et puis une dernière catégorie, plus abstraite, qu'est l'information, disons, conceptuelle, liée à la composition de l'œuvre, comme la notation musicale par exemple, mais cela peut être aussi les programmes informatiques qui servent à la composition de l'œuvre. Il y a ainsi une sorte d'abstraction croissante.

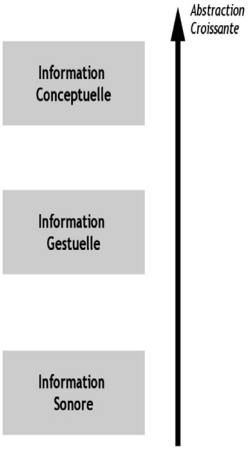

Figure 16 – Classement de l'information musicale par abstraction croissante.

Sur ces trois types d'informations vont s'appliquer les trois types de traitement que l'on a vus. Souvent, du reste, les traitements de type analytiques vont passer d'un stade d'information, par exemple l'information sonore, à l'information conceptuelle. À l'inverse, des informations, des traitements de type synthèse par exemple, vont passer de l'information conceptuelle à l'information gestuelle, si on veut modéliser, faire de la synthèse de la performance ou directement au phénomène sonore si on fait de la synthèse du son.

Enfin, il y a une dernière modalité qui est celle de la représentation et du traitement du temps. Quand on est dans le domaine de la musique, on traite d'informations évidemment de type temporelles. On va avoir deux grandes catégories, deux pôles en quelque sorte, qui sont le traitement différé et le traitement temps réel de cette information. Il y a souvent une mauvaise compré-

hension de ce qu'est le temps réel. Le temps réel n'est pas forcément le fait d'aller au plus vite, c'est le fait que le moment précis auquel on collecte l'information et le moment précis auquel on délivre l'information sont importants. Il ne faut être ni avant, ni après. C'est ça véritablement le temps réel. Ce n'est pas seulement une question de rapidité. Évidemment, si on est rapide, on a plus de chances de pouvoir réaliser les impératifs du temps réel.

Cette typologie des calculs musicaux donne une typologie de questions intéressantes, en tout cas qui nous intéressent très directement à Grame dans notre travail de recherche et en particulier cette passerelle qui est entre l'utilisateur et la machine. Puisqu'on parle de calculs musicaux, va se poser le problème de savoir comment on imagine ces calculs musicaux, comment nous inventons un calcul sonore, comment nous le communiquons à la machine, comment nous le mettons en œuvre au niveau de la machine. Et puis, il y a une nouvelle problématique, qui existe depuis très longtemps mais dont on commence seulement à prendre conscience et pour laquelle on n'a pas encore vraiment de solutions, qui est de savoir comment on conserve ces calculs indépendamment des logiciels qui les ont vus naître en quelque sorte, des machines qui les ont vus mettre en œuvre.

Un autre élément de réflexion concerne la pensée créatrice.

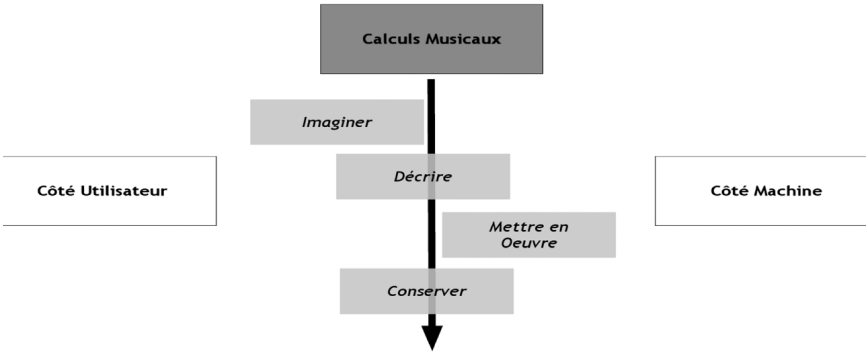

Figure 17 – Quelques questions intéressantes relatives aux calculs musicaux.

La Pensée Créatrice

Usuellement, on distingue deux choses dans la pensée créatrice, ce qui relève de l'invention et ce qui relève de la découverte. La découverte, c'est la découverte d'objets existants comme par exemple la découverte de l'Amérique, du virus du HIV etc., donc des choses que, si on ne les avait pas découvertes soi-même, d'autres auraient pu découvrir. L'invention, elle, est beaucoup plus personnelle, ce qui fait que si quelqu'un n'écrit pas telle pièce, il y a peu de chance que d'autres l'auraient écrite. Donc nous allons plutôt nous s'intéresser à l'aspect invention qui correspond plus à l'acte artistique.

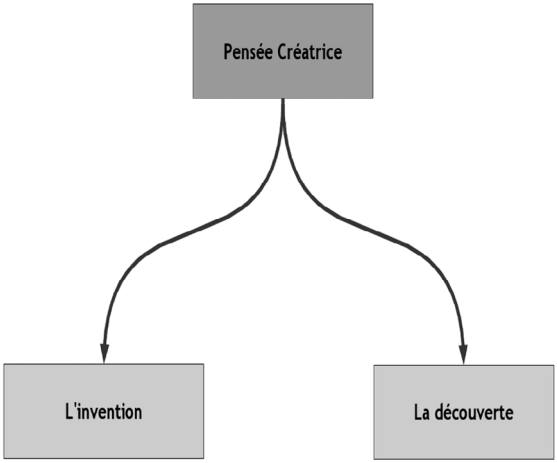

Figure 18 – Invention et découverte,
deux formes de la pensée créatrice.

Un certain nombre de théoriciens se sont intéressés à l'acte créatif, dont Todd Lubart. Ce dernier cite sept critères.

Identifier les problèmes intéressants, quand on fait de la recherche, il est évident que se poser les bonnes questions est un élément fondamental dans la démarche de recherche. C'est un élément fondamental du directeur de thèse que d'orienter son étudiant vers des questions potentiellement intéressantes. Ensuite Lubat cite le fait de **collecter toutes les informations per-**

tinentes, le fait de pouvoir établir des passerelles entre des mondes différents, de **faire des analogies**, de remarquer des similitudes, le fait de pouvoir prendre des bouts d'idées existantes et d'en fonder de nouvelles comme ça, en les **combinant**. Il parle aussi de **pensée divergente**, le fait de pouvoir, à partir d'une même idée, en générer toutes sortes de variantes. Un autre élément extrêmement important est **l'évaluation des idées**, la capacité à reconnaître les bonnes idées (point sur lequel nous allons revenir plus loin). Cela est tout à fait essentiel dans l'acte de création. Ensuite il parle de la **flexibilité**, la capacité à sortir du cadre, à ne pas rester enfermé en quelque sorte dans un trou, dans les idées de départ, de pouvoir en sortir.

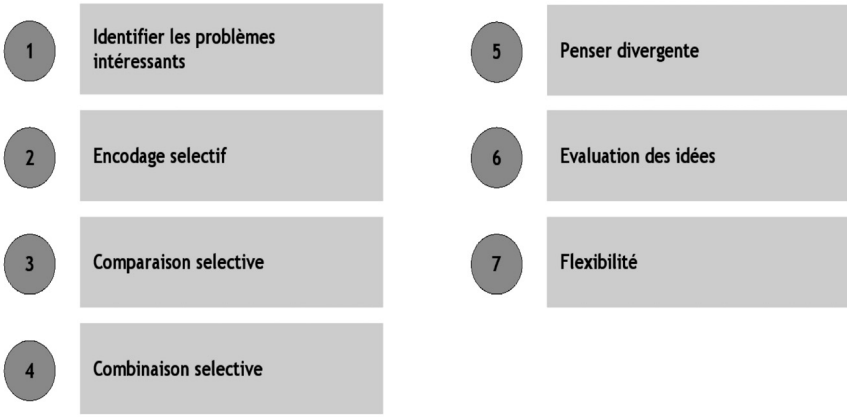

Figure 19 – Capacités essentielles à l'acte créatif.

Voilà pour les écrits récents sur l'acte créatif, mais à notre avis qui ne disent pas l'essentiel de la chose. Nous citerons un ouvrage remarquable, écrit par le mathématicien français Jacques Hadamard dont le titre est « Essai sur la psychologie de l'invention dans le domaine mathématique »[12]. Pour nous, ce fut

12. Jacques Hadamard, « Essai sur la psychologie de l'invention dans le domaine mathématique ». Texte suivi de: Henri Poincaré, « L'invention mathématique », éd. Jacques Gabay, 2007.

une sorte de révélation de lire ce livre et le texte de Poincaré, parce qu'il explique beaucoup de choses. Il explique par exemple pourquoi les mathématiciens sont si sensibles à la beauté des théories mathématiques et pourquoi il existe une certaine esthétique des mathématiques. Il explique aussi le phénomène de l'émotion liée à l'idée. Vous cherchez quelque chose et, tout d'un coup, une sorte d'émotion arrive, l'idée surgit etc. Bien sûr, il y a cette vision romantique de l'artiste qui, tout d'un coup, une nuit d'orage, est touché par la grâce et l'œuvre surgit. Cela correspond aussi à une certaine réalité. Cette théorie, ce modèle de l'acte créatif de Poincaré, Wallace et Hadamard y apportent des réponses. Nous allons l'expliquer en quelques mots.

Poincaré indique qu'il travaillait sur une équation (ou autre chose) à son bureau sans trouver de solution à son problème. Il part ensuite en vacances. En faisant un voyage en calèche (ou en bus), en montant sur la marche du véhicule, tout à coup, il a l'idée. Il rentre ensuite chez lui et vérifie l'idée.

Il y a quatre phases : **la phase de mobilisation**, le moment où on travaille sur un problème, quand par exemple on se dit « je vais composer une pièce, avec un certain effectif etc. ». La phase suivante est **la phase d'incubation** pendant laquelle on pense à autre chose, on travaille sur autre chose bien qu'en même temps l'inconscient travaille pour nous. Ensuite vient **la phase de révélation** où tout d'un coup il y a une sorte de jaillissement quelque part, il y a une émotion très forte que l'on ressent, une sorte d'excitation et *clac !* on a l'idée, elle arrive de manière soudaine. En réalité, elle n'est pas soudaine puisqu'elle cheminait, elle se fabriquait de façon inconsciente. L'idée n'est pas toujours la bonne. Il y a alors **une phase de vérification**. Si l'idée n'est pas bonne, on est déçu et on recommence.

La phase de mobilisation, pour Poincaré, est une phase pendant laquelle on va mobiliser au niveau inconscient des outils (Poincaré parle de chiens, en quelque sorte, qui vont suivre des pistes) qui vont chercher à notre place.

Dans la phase d'incubation, pendant laquelle l'inconscient va travailler, il faut quelque part un mécanisme qui permette à l'in-

conscient de signaler à la conscience quelque chose de potentiellement intéressant.

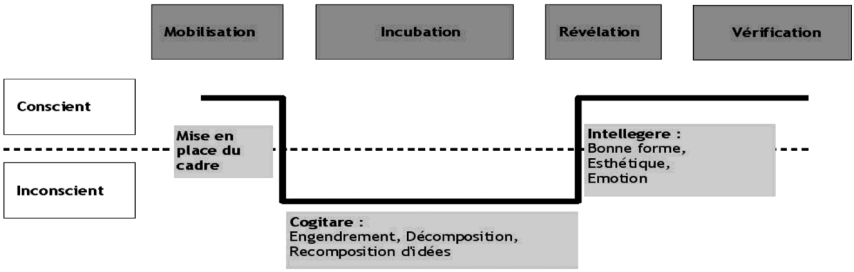

Figure 20 – Modèle classique de l'acte créatif (Poincaré, Wallas, Hadamard).

D'après les études psychologiques, l'inconscient ne calcule pas. C'est-à-dire qu'il est très rare de rêver de manière exacte d'une multiplication de nombres à cent chiffres et d'avoir effectivement le bon résultat. Par contre, l'inconscient est très fort pour faire des analogies, pour faire des recompositions, des recombinaisons. Cela veut dire que le moment où l'inconscient signale à la conscience qu'il a trouvé quelque chose de potentiellement intéressant, ce n'est pas basé sur la réalité de la chose, mais sur une esthétique de la chose, sur le sentiment qu'on a trouvé une bonne forme. C'est pour ça que l'esthétique est si importante, y compris en mathématiques, y compris en sciences. Il y a un goût, une capacité à reconnaître potentiellement les choses intéressantes.

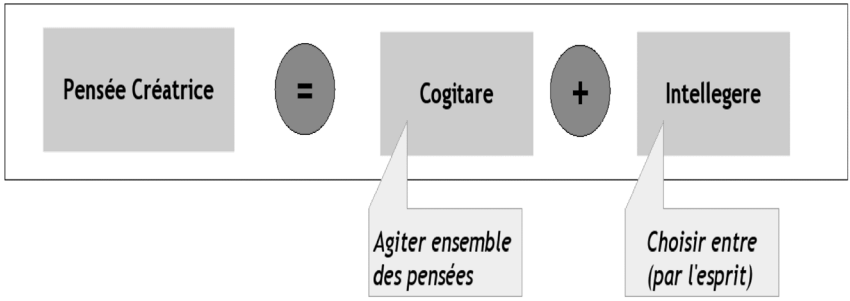

Figure 21 – Intellegere et cogitare.

Les gens créatifs sont des gens qui sont capables de faire deux choses à la fois : du ***cogitare*** au sens d'agiter ensemble des idées, de les décomposer de les recomposer afin d'en produire de nouvelles, ce que fait très bien l'inconscient, et de ***l'intellegere***, au sens de reconnaître les idées qui sont potentiellement intéressantes. Après, bien sûr, il y a une phase de vérification. Tout ce que nous voulons dire est basé sur ces deux choses-là : le *cogitare*, agiter ensemble des idées pour en produire des nouvelles, et l'*intellegere*, reconnaître les idées potentiellement intéressantes, avoir une esthétique des idées. On sent bien que quand on essaye d'être créatif, on essaye de se mettre ainsi dans des situations où on va pouvoir fabriquer des idées, se mettre dans une situation où on va pouvoir voir passer des choses intéressantes. La créativité marche sur ces deux jambes-là. Fabriquer des idées n'est pas suffisant. Être capable de reconnaître les idées intéressantes est tout à fait indispensable à l'acte de création.

Cette combinaison « *cogitare* + *intellegere* » se fait bien sûr dans notre tête, mais on fait ça aussi avec des tas d'outils. Ainsi, l'architecte Richard Mac Cormack dit « j'utilise le dessin comme processus de critique et de découverte », c'est-à-dire que « je vais me mettre devant une feuille de papier, je vais *cogitare* avec ma main, avec mon bras, en quelque sorte, je vais faire des gribouillis, je vais faire des esquisses et je vais *intellegere* avec mon œil, avec ma pensée et donc je vais créer une sorte de boucle créative » et cette boucle créative peut se faire dans notre tête, mais elle se fait aussi par exemple avec une feuille de papier quand nous gribouillons, elle se fait avec un piano quand nous improvisons au piano et que nous essayons de trouver des choses, elle se fait quand nous préparons une conférence et que nous écrivons ce que nous allons dire, que nous le lisons à haute voix en nous disant « ce n'est pas ça que nous voulons dire », etc.

Donc il y a besoin comme ça d'une phase de réflexion entre le fait de produire des idées, le fait de les voir produites, pour pouvoir ensuite les sélectionner. L'ordinateur va intervenir comme ça, comme une sorte de papier « superpuissant », qui va intervenir dans cette boucle entre le *cogitare* et *l'intellegere*.

"J'utilise le dessin comme processus de critique et de découverte"
(Richard MacCormac, architecte)

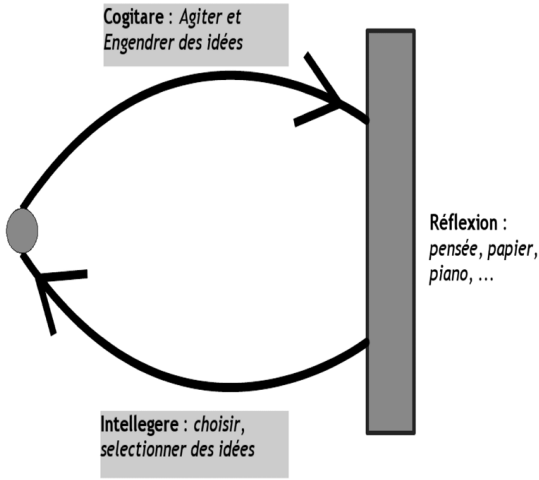

Figure 22 – Le modèle du croquis.

La Programmation Créative

Cela nous amène au concept de programmation créative, qui est, comme on peut le voir dans les autres textes de cet ouvrage, le fait d'utiliser la programmation informatique non pas comme moyen d'écrire des programmes mais comme moyen de conception et d'invention d'objets un petit peu complexes. La clé de cette chose-là, c'est la relation entre les descriptions en intention, ce qu'on appelle les modèles dans les autres textes de cet ouvrage par exemple, et la description en extension, c'est-à-dire la réalisation du modèle, la chose telle qu'on va pouvoir la percevoir. L'ordinateur est un super-moyen pour justement pouvoir, d'une part rentrer des descriptions d'intentions et d'autre part percevoir des descriptions en extension. Cela va être ça l'une des clés du processus d'invention avec l'ordinateur.

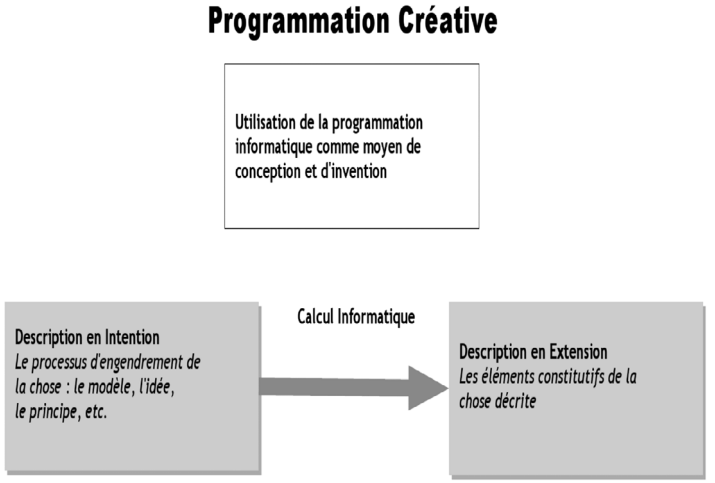

Figure 23 – La programmation créative, entre intention et extension.

Nous allons donner différents exemples. On voit, dans le schéma suivant, trois exemples de plantes artificielles.

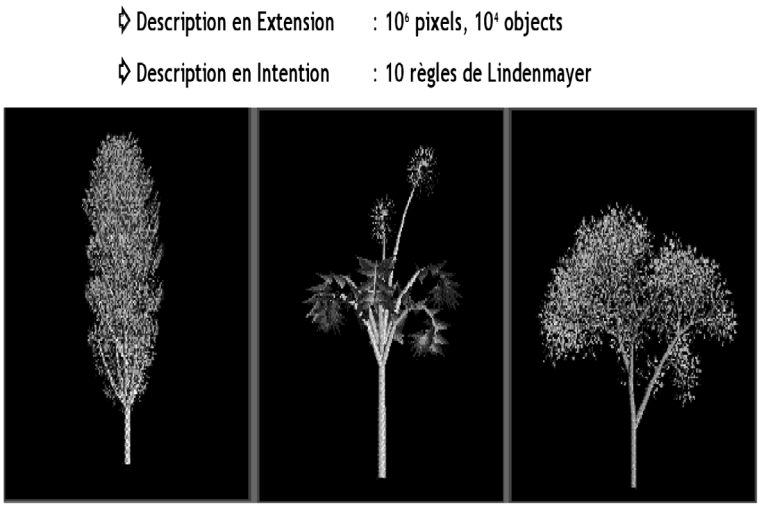

Figure 24 – Description de plantes artificielles.

Si on les regarde, on peut les reproduire en tant qu'image ou en tant qu'objet graphique 3D avec des sortes de primitives, des feuilles, des cylindres, des choses comme ça, mais ce sont des

objets complexes qui comportent plusieurs milliers d'objets primitifs et il est évident que si nous reproduisons cet objet-là en ajustant les objets primitifs, les feuilles et les troncs, nous restons en surface, sans capturer l'essence de la chose. En réalité, pour capturer l'essence de la chose, il nous faut faire une sorte de modèle. Il faut que nous décrivions non pas la chose mais la façon de produire la chose. Si nous sommes capables de décrire la façon de produire la chose de manière concise, condensée, et bien c'est beaucoup plus satisfaisant, nous avons capturé quelque chose de l'essence de la plante que nous cherchons à modéliser. Par exemple, toutes les plantes de la figure se modélisent très bien avec quelques règles d'un système qui est le système de Lindenmayer[13] une sorte de langage Logo amélioré qui permet de décrire des plantes (ou plus exactement leur processus de croissance).

On est donc au cœur du débat qui est l'utilisation du langage informatique, de la programmation pour décrire, non pas l'objet final, mais le processus d'engendrement de l'objet.

Ce qui est très intéressant c'est que, d'une part certains objets sont trop complexes pour qu'on puisse les décrire in extenso. On ne va pas s'amuser à décrire des milliers d'échantillons sonores, échantillons par échantillons. La seule façon que nous avons d'imaginer un nouveau son, c'est de décrire le processus d'engendrement de ce son. D'autre part, une fois que nous avons décrit le processus d'engendrement, nous pouvons faire de petites variations sur le processus d'engendrement et nous allons obtenir de grandes variations sur le résultat engendré. C'est là que va se situer la découverte, découverte qui ne doit rien au hasard, qui ne doit rien à la machine, si ce n'est l'effet d'amplification de la machine.

Autre exemple, pour un son, quelques lignes de codes (tirées du programme Faust) vont engendrer des millions d'échantillons sonores. Une petite modification dans les lignes de code va pro-

13. A. Lindenmayer, « Mathematical models for cellular interactions in development, Parts I and II », *Journal of Theoretical Biology*, vol. 18, 1968, p. 280-315.

duire des sons très différents, alors que si nous voulons, avec un éditeur, éditer les échantillons sonores, cela n'a absolument pas de sens et ne va rien produire d'intéressant.

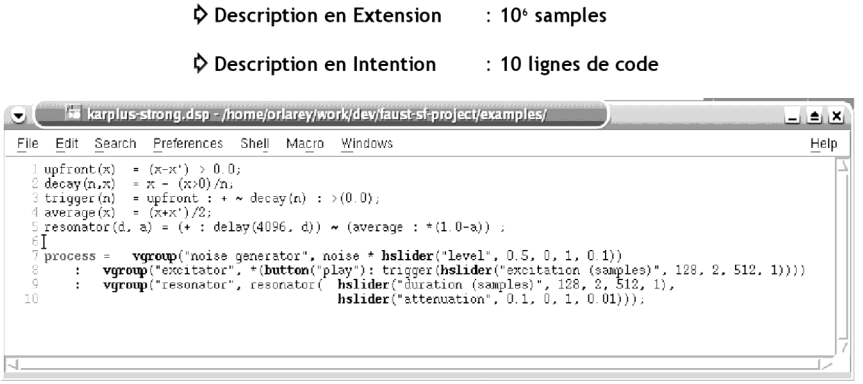

Figure 25 – Synthèse de sons.

L'exemple le plus probant, c'est évidemment la biologie, avec l'ADN et le processus d'engendrement des corps humains ou des corps biologiques. On sait très bien, avec l'exemple de la fameuse mouche drosophile qui est en quelque sorte le laboratoire des biologistes, que de petites modifications vont faire que la mouche aura deux fois plus de tête, trois fois moins de pattes, des ailes en plus, deux abdomens etc. De petites modifications dans le programme en quelque sorte, dans le processus d'engendrement vont conduire à de grosses modifications dans l'objet engendré.

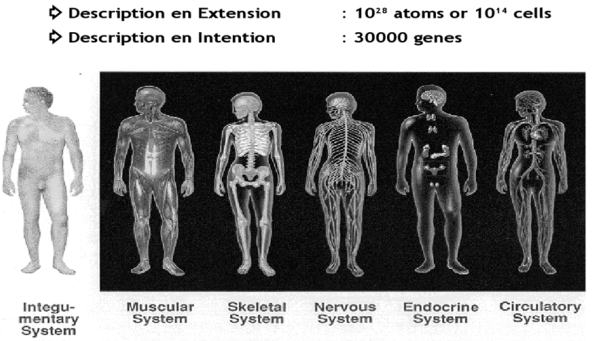

Figure 26 – « Description » du corps humain.

Un autre exemple dans un langage qui a été développé à Grame. On voit dans le schéma suivant un objet graphique extrêmement complexe, qui en réalité quand on le décrit, fait appel à soixante-treize éléments de départ.

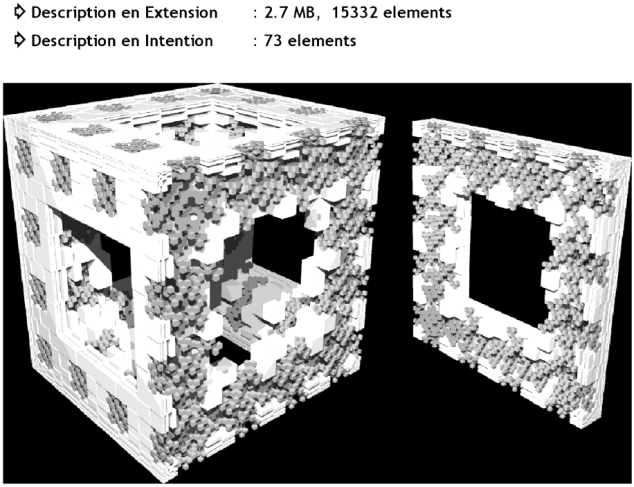

Figure 27 – Description d'objets 3-D.

Conclusion : que se passe-t-il entre calcul, programmation et création ?

L'ordinateur rentre dans cette boucle, entre le *cogitare*, l'agitation des idées, la production d'idées nouvelles, et *l'intellegere* le fait de choisir et de reconnaître les idées intéressantes. Cette boucle-là est médiatisée par le calcul de l'ordinateur. Il y a un certain nombre d'objets qui sont suffisamment complexes, un son par exemple, mais aussi une interaction dans le cadre d'une installation sonore pour qu'on ne puisse pas les décrire in extenso. On ne peut que décrire le processus qui va les engendrer. Le langage de programmation est extrêmement important dans cette description-là. Il y a une qualité, une ergonomie cognitive qui fait que c'est plus ou moins facile de penser les problèmes. Il nous donne une grille de lecture du monde d'une certaine manière, comme la notation musicale. Ce n'est pas un outil neutre. Si vous

voulez déclencher une bagarre entre deux informaticiens, lancez le débat sur les langages de programmation par exemple. S'ils ne sont pas d'accords, ils vont se taper dessus. Par exemple, nous sommes fan de la programmation fonctionnelle. Si vous nous parlez de programmation objet, nous allons vous dire que c'est nul et que seule compte la programmation fonctionnelle.

Donc les langages de programmation ont une dimension extrêmement importante. Dans la boucle qu'on voit sur la figure suivante, entre le fait d'exprimer des idées et de se servir de langages de programmation comme support, comme notation des idées, le langage de programmation, le programme devient une représentation de l'idée sous-jacente. Cela veut dire qu'on va avoir une première boucle de feedback, entre le *cogitare* et *l'intellegere,* sur les idées, sur les processus eux-mêmes. Cela veut dire que la capacité qu'a le langage à permettre justement d'agiter des idées, de prendre un bout de programme, de prendre un autre bout de programme, de les assembler et de faire en sorte que ce soit un programme qui marche est extrêmement importante.

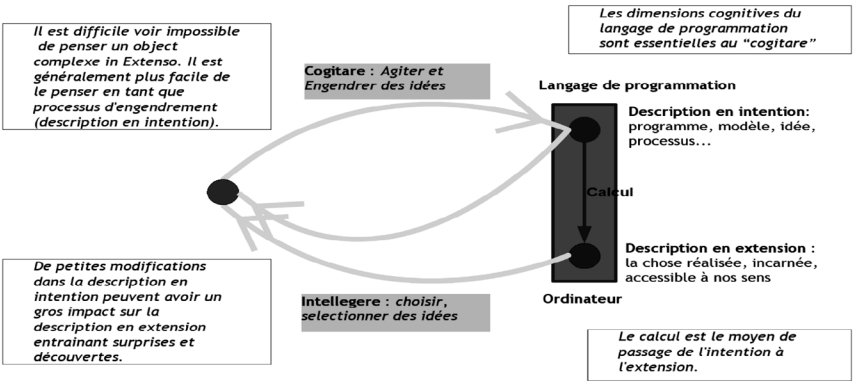

Figure 28 – Boucle créative entre programmation et calcul.

Actuellement, les langages de programmation nous forcent à formaliser les choses, mais nous pensons que dans le futur, les langages de programmation nous permettront d'avoir une

démarche beaucoup plus expérimentale sur les choses, c'est-à-dire qu'en gros, on pourra beaucoup plus programmer au hasard qu'on ne peut le faire actuellement. Il y a donc un enjeu, très spécifique à la création artistique, qui est l'utilisation des langages de programmation, non pas en ayant déjà en tête l'idée du programme que l'on veut faire, mais comme moyen, comme une sorte de pâte à modeler qui va nous permettre de créer de nouveaux programmes, sans forcément savoir dès le départ ce qu'on va vouloir faire. Cela nécessite une évolution de la technologie, mais nous pensons que ce sera une évolution de la technologie motivée par la musique ou l'art, extrêmement positive en termes de technologie pour la programmation.

Après cette première boucle de feedback, il y a une deuxième boucle de feedback, extrêmement puissante, qui est celle du calcul dans lequel va intervenir l'ordinateur. C'est-à-dire que si nous voulons dire : « tiens, je vais faire un programme de synthèse de sons, je vais prendre tel bout de tel programme de synthèse de son que j'ai déjà fait, tel autre bout, je vais les mettre ensemble, qu'est ce qui va se passer ? », nous sommes motivés par ce que nous savons de chacun de ces deux programmes, mais, en même temps, nous ne savons pas exactement quel va être le résultat de la chose. Le fait que l'ordinateur, par un calcul, nous transforme cette description en intention, en description en extension extrêmement rapidement, et bien cela va alimenter notre *intellegere*, nous rendre compte que, oui, c'est ça que nous voulons faire, ou c'est à peu près ça, et donc on va entrer dans une sorte de boucle de feedback extrêmement productive, où l'ordinateur ne se substitue pas à nous, mais amplifie en quelque sorte notre capacité créative. Nous croyons que c'est cela le cœur de l'ordinateur comme outil d'aide à la création. Nous ne disons pas qu'on ne puisse pas utiliser l'ordinateur dans d'autres usages, mais en tant qu'outil d'aide à la création c'est comme ça : rentrer dans cette boucle entre le cogitare et l'intellegere et introduire une sorte d'amplification énorme dans cette boucle de calcul.

Pour dire deux mots d'un certain nombre de travaux que nous avons faits à Grame.

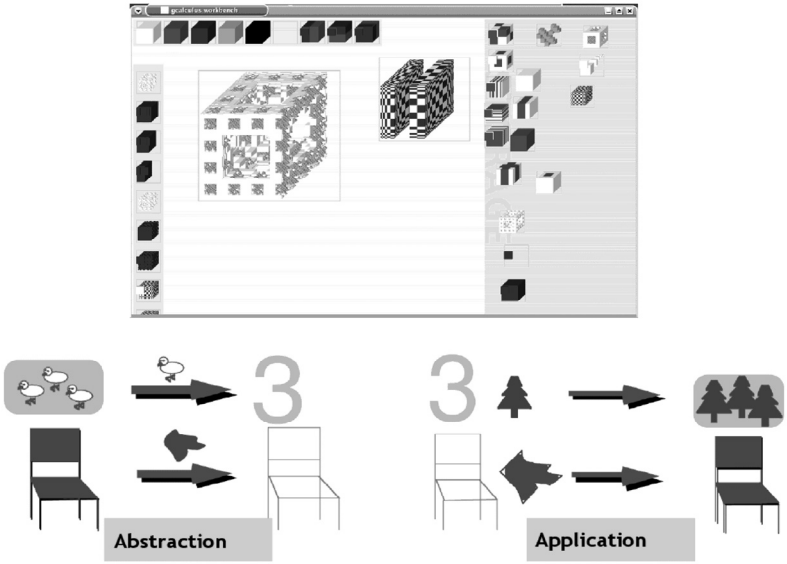

Figure 29 – Exemple de programmation créative
dans le domaine graphique, le G-Calcul.

Dans l'exemple de la figure, il s'agit d'un langage de programmation qui a la particularité de ne pas faire appel à un langage de programmation au sens usuel du terme. Il ne sert à rien d'autre qu'à faire des images. On a des cubes de couleur et on a trois façons de les assembler. On peut prendre deux cubes de couleur et les mettre gauche-droite, devant-derrière, haut-bas. C'est une matière un peu spéciale puisque quand on prend deux cubes de couleur, le résultat est toujours un cube, avec la moitié de la première couleur et la moitié de la deuxième couleur. On peut faire un certain nombre de choses avec ça, mais ce qui est intéressant, c'est l'introduction de la programmation. En 1936, concernant la programmation, il y avait les machines de Turing et le lambda calcul d'Alonso Church. Le lambda calcul c'est quelque chose comme la physique quantique de la programmation. Cela réduit la programmation à deux concepts : le concept d'abstraction et d'application et ça montre qu'on peut tout créer à partir de ces deux concepts-là. La logique booléenne, les nombres, tous les objets habituels de l'informatique n'ont pas besoin de préexister.

Simplement en introduisant le concept d'abstraction et d'application, et en les combinant on obtient tout ça. Ce qu'il y a dans ce G-calcul-là, c'est qu'on a des cubes et le concept d'abstraction et d'application du lambda-calcul. En gros, l'idée c'est de dire que si nous voulons capturer la forme d'un objet comme la chaise rouge par exemple, nous pouvons abstraire, au sens d'oublier la matière rouge de la chaise, pour capturer la forme de cette chaise et là nous avons créé un programme.

Ce programme, nous pouvons ensuite l'appliquer, par exemple à une autre matière, de la matière bleue pour créer une chaise bleue. Cela paraît extrêmement trivial, extrêmement simple mais à partir de ces deux concepts-là, on crée tous les autres concepts. Par exemple, si nous prenons une collection de trois poussins et que nous abstrayions le fait que ce sont des poussins, nous capturons l'idée de trois objets identiques. Ce concept de trois objets, si nous l'appliquons à un sapin par exemple, nous créons trois sapins.

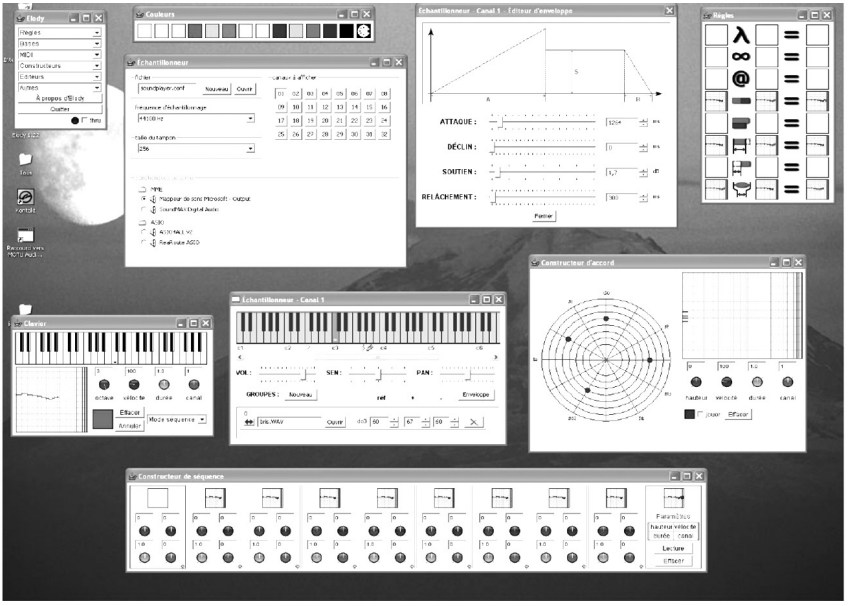

Figure 30 – Exemple de programmation créative dans le domaine musical, Elody.

Ce qui est intéressant c'est que ce principe réduit à deux concepts, abstraction et application, extrêmement puissant, nous pouvons l'introduire dans n'importe quel type d'objet, donc dans des cubes de couleur mais nous l'avons aussi introduit dans le système Elody qui est un langage de composition musicale où on retrouve l'idée de dire « je vais créer des objets musicaux, des structures mélodiques etc. » mais nous voulons pouvoir abstraire, généraliser une structure mélodique en rendant variables certains ingrédients que nous avons utilisés dans cette structure mélodique, pour capturer en quelque sorte le principe de composition.

Nous pouvons faire un canon par exemple à partir d'une séquence bien particulière que nous transposons, que nous décalons, que nous mixons puis si cette structure-là particulière nous plaît, nous pouvons nous en servir en tant qu'exemple d'un processus de composition et pour cela nous allons indiquer quelle est la partie variable. Nous allons dire voilà, c'est cette structure-là, il faut oublier la séquence de départ que nous avons utilisée, pour uniquement garder l'esprit en quelque sorte, la structure du processus. Avec ça, on fait des programmes, ... et de la création !