GNU LILYPOND : POUR UNE LINGUISTIQUE DU CODE SOURCE

Valentin VILLENAVE

Que l'on se place du point de vue des auteurs, des artistes ou de la société en général, la dématerialisation des supports culturels (et, partant, des processus créatifs) présente une ambigüité dialectique : elle peut être libératrice, ou aliénante. Libératrice en ce qu'elle ouvre des possibilités quasi inépuisables (dont la plus importante est sans doute la notion d'immédiateté que l'on peut aujourd'hui percevoir à tous les niveaux, de la création à la diffusion culturelle) ; aliénante parce qu'elle induit bien souvent une dépendance nouvelle à la technicité. Or l'ambition même d'une démarche artistique est précisément d'échapper à de telles contingences – ce qui définit une œuvre d'art n'est ni son utilité ni son idiosyncrasie, mais sa part d'universalité et de pérennité : à ce titre, il est d'ailleurs significatif que l'édition 2011 des Journées d'Informatique Musicale ait choisi de se concentrer sur la "pré-ser-va-tion des œuvres". Comment, en effet, assurer la pérennité et l'intelligibilité de créations intimement liées à des formats de données très spécifiques (souvent cryptés), à des logiciels figés, voire à une architecture matérielle en particulier ?

Ce problème, les logiciels dits "Libres" y répondent depuis les débuts de l'informatique (même si leur approche n'a été théorisée comme telle que depuis une trentaine d'années, en réaction à l'avènement d'acteurs industriels et commerciaux dont le modèle économique repose précisément sur l'assujettissement des données personnelles et des œuvres de l'esprit) en permettant de lire, d'adapter et de recompiler le code source des programmes au gré des besoins. Ce souci de transparence et de portabilité est au cœur de la "philosophie Unix", que Douglas McIlroy formulait ainsi dès les années 1960 : "n'écrivez que des programmes dont le format d'entrée est du texte pur, car c'est là une interface universelle". De fait, le texte pur, ou code source, demeure le format le plus ouvert, le plus accessible, le plus pérenne que l'informatique puisse garantir à ce jour : le réseau Internet nous en fournit nombre d'illustrations éclatantes.

Certains "langages" non-textuels, bien entendu, ne peuvent faire l'économie d'un "media", encodé voire compressé dans un format binaire ou hexadécimal : ainsi des captations sonores ou visuelles. D'autres formats sont, au contraire, orientés essentiellement vers le langage verbal, et peuvent dans bien des cas faire sens même sans formalisme spécifique : une page HTML peut être lue dans un navigateur en mode texte, et il est même possible de comprendre un document en LaTeX en lisant directement son code source. La notation musicale, en revanche, occupe une place plus subtile : c'est un langage non-textuel, mais tout aussi formalisé et conceptuel que n'importe quel langage verbal. Utiliser du code textuel pour représenter la musique est donc non seulement possible, mais nécessaire et évident (l'on m'objectera peut-être que les logiciels commerciaux d'édition musicale s'obstinent à stocker leurs fichiers dans des formats binaires cryptés : il conviendrait de se demander pour quelle raison, si ce n'est dans cette volonté d'assujettissement que j'évoquais plus haut).

Un langage ouvert de notation musicale, sous forme de code source textuel, peut s'orienter dans deux directions. Il peut favoriser la simplicité au détriment du formalisme : c'est le cas de ABC, ou dans une moindre mesure de GUIDO. Il peut, au contraire, rendre compte du formalisme de la musique (d'une façon souvent plus descriptive qu'analytique), au risque d'être impraticable pour l'utilisateur humain : c'est l'approche adoptée par tous les langages inspirés de la syntaxe XML, qu'il s'agisse de MusicXML, CanorusML ou MusE Score – notons qu'aucun de ces formats n'est destiné à être manipulé directement en tant que code source par l'utilisateur : ils nécessitent tous une interface graphique... et l'on retombe donc dans le problème de dépendance et de technicité évoqué plus haut. GNU LilyPond, enfin, propose une approche plus originale : son langage de représentation est à la fois simple et conceptualisé, expressif et extensible (il s'agit d'un langage "Turing-complet"). De cette plasticité incomparable naît un rapport nouveau entre l'auteur-codeur et la partition : l'écriture de la source (le "style" du code, comme disent les programmeurs) reflète la pensée musicale, mais peut également la structurer, et même la susciter. Avec Lilypond, une même pièce pourrait se coder de mille façons différentes – mais serait-ce alors vraiment la même pièce ?

Ce rapport nouveau entre le code et la partition n'est d'ailleurs pas sans rappeler celui, en miroir, qui existera ensuite entre la partition et l'interprète. Et à ce titre, le fait pour un compositeur de se servir d'un logiciel tel que LilyPond, avec son expressivité propre (la possibilité de commenter ou non le code, de l'organiser d'une façon bien précise plutôt qu'une autre, voire de documenter, au fil de l'écriture, les étapes du processus créatif, les repentirs et les subterfuges...), apporte une richesse sans précédent quant au regard que nous pouvons porter sur son œuvre : un don du ciel pour les historiens comme pour les musiciens !

De cette forme créative nouvelle que constitue le code source, il me semble aujourd'hui souhaitable de jeter les bases d'une étude linguistique. Je ne l'entends pas au sens (d'ailleurs quelque peu galvaudé) des "computational linguistics" chers à une certaine branche de l'informatique,

mais au sens beaucoup plus large qui a précédé à l'avènement de toutes les sciences du langage il y a tout juste un siècle. Je tâcherai pour cela de partir de quelques exemples concrets, notamment tirés de mon expérience personnelle, pour en dégager quelques hypothèses théoriques ; afin toutefois de ne pas me limiter à la théorie, je tenterai ensuite de les remettre en perspective autour de l'évolution actuelle de la société en général, et des pratiques culturelles et créatives en particulier.