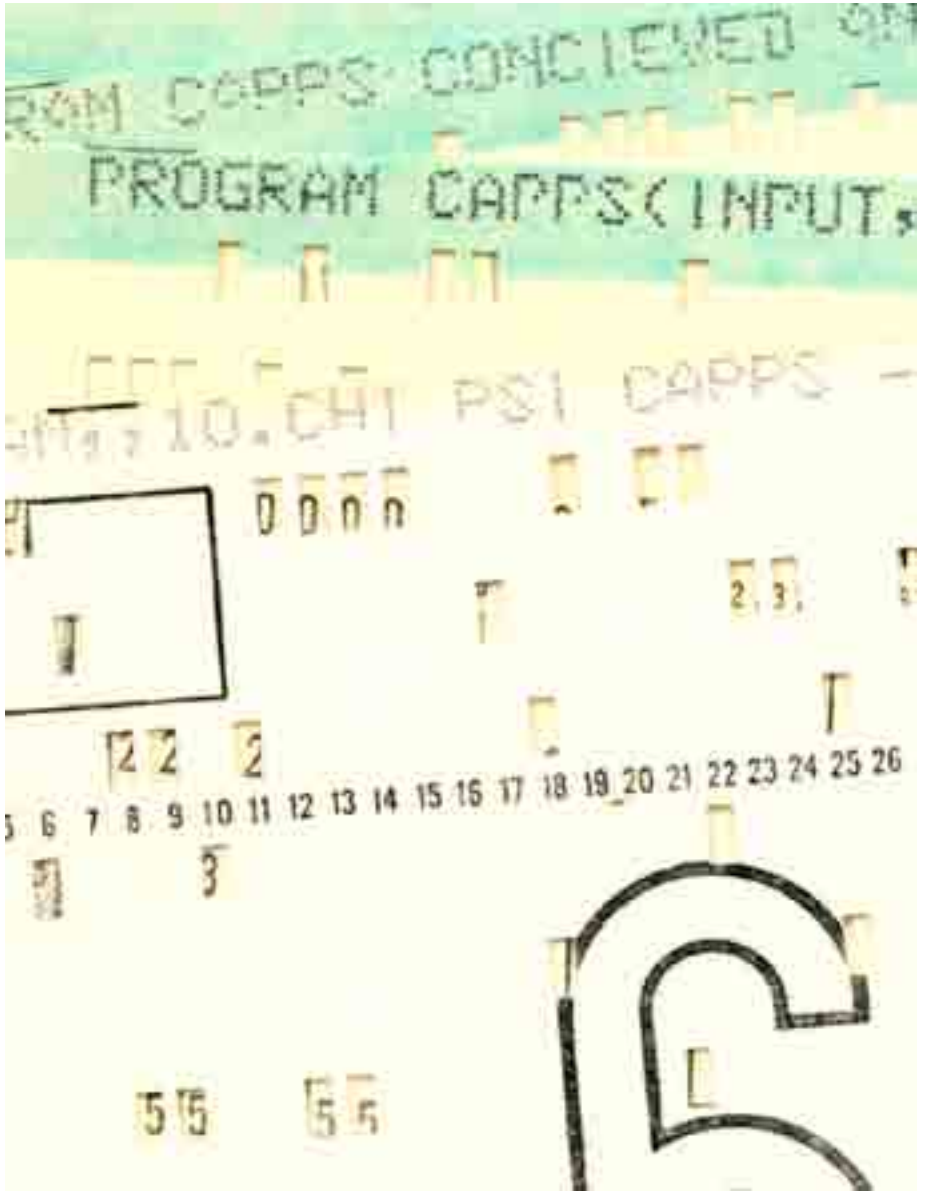


Carte perforée, 1970.

# Retour au code[1]

David-Olivier Lartigaud

Techniquement, la programmation est au fondement de toute production par ordinateur, qu'elle soit artistique ou non[2]. Toutefois, l'acte de programmer n'a pas toujours tenu la même fonction au sein des œuvres faisant appel à l'informatique.

Au cours des années 1960, à l'époque des premières recherches en art visuel sur ordinateur, les logiciels de graphisme (de type Photoshop) ou les langages « destinés aux artistes » (comme Processing) n'existaient évidemment pas[3]. La programmation à l'aide d'un langage « standard » était donc l'étape préalable à toute tentative artistique. Produit par l'artiste, par un ingénieur informaticien pour l'artiste, voire par un ingénieur-artiste, le code était un terrain d'expérimentation au cœur du processus créatif. Les œuvres utilisaient souvent des algorithmes simples et des programmes relativement brefs qui étaient parfois exposés en parallèle du rendu artistique obtenu.

Au cours des années 1970, le caractère programmé des œuvres devint moins primordial. La programmation restait centrale mais les problématiques se déplacèrent. L'animation par ordinateur et l'image de synthèse commencèrent à révéler leur potentiel attestant de la capacité des machines à être pourvoyeuses d'images en mouvement. Le graphisme sur ordinateur devint un domaine de recherche technique à part entière,

---

**1.** Code/coder/codeur ou programme/programmer/programmeur ? Un éclaircissement s'impose concernant l'emploi de ces mots. Si l'usage courant les a rendus synonymes en informatique, précisons que seul programme/programmer/programmeur sont corrects en français. L'utilisation du mot « code » pour « programme » n'est pas totalement impropre mais relève de l'approximation puisqu'il s'agit de l'abréviation du terme technique « code source » (les instructions écrites par le programmeur dans un langage de programmation particulier). Il est clair cependant que cette utilisation du mot « code » dans le contexte informatique éveille une nébuleuse de sens plus large que le strict « code source ». À commencer par le code binaire, au fondement de l'ordinateur, mais aussi, par exemple, le code ASCII (une norme de codage des caractères alphanumériques), et plus globalement au code comme « système rigoureux de correspondance entre ensembles de signes » qui est, comme le précise *Le grand Robert de la langue française*, hérité directement de la *Théorie de l'information* de Shannon et Weaver. Ainsi, l'utilisation du mot « code » à la place de « programme » n'est souvent pas anodine et signale le désir d'inscrire dans une sphère de compréhension plus large les problématiques liées à la programmation. En cohérence, « coder » prend le sens d'« écrire du code » ou, plus exactement, de « rédiger un programme », entretenant là aussi une ambiguïté par l'écart avec sa signification conventionnelle de « mettre en code »/« procéder au codage de… » (*Le grand Robert de la langue française*). « Codeur », moins fréquent et sans homonyme en français puisque la « mise en code » est réalisée par un *en*codeur, désigne en jargon informatique une personne qui programme. C'est un emprunt direct à l'anglais *coder* qui est également une formule familière pour parler d'un *programmer*.

**2.** Comme le rappel Norbert Ebel dans son article « Langages de programmation impératifs » pour l'*Encyclopédie de l'informatique et des systèmes d'information* (Vuibert, Paris, 2006, p. 1007.) : « La programmation est l'activité centrale de l'informaticien même si de très nombreux autres domaines constituent la technique de l'informatique. » Puis il donne, quelques lignes plus loin, des précisions qu'il est intéressant de rapporter ici car elles montrent combien le programme, son écriture et son exécution s'inscrivent dans un cadre de référence plus large que la seule sphère informatique : « Un programme n'exprime ni des idées ni des concepts, n'est pas un récit et ne véhicule aucun sentiment ; il donne des instructions *impératives* à un ordinateur non doté d'intelligence. Cependant on s'est rendu compte qu'un programme est fréquemment lu par d'autres humains et relu par l'auteur ; de ce fait, il doit donc aussi répondre à des critères littéraires ! On peut ainsi qualifier tel programme d'élégant, tel autre de lisible, de touffu, etc. On dit qu'un ordinateur exécute un programme sans doute par analogie avec les membres d'un orchestre qui exécutent une partition. L'analogie est cependant trompeuse puisqu'un ordinateur n'est ni doué d'intelligence ni de sensibilité ! » Rappelons que le mot « programme », du grec *programma* qui signifie « ce qui est écrit à l'avance », est usité dans de nombreux domaines (science, mathématiques, politique, monde du spectacle…) ; la définition qu'en donne l'*Encyclopédie philosophique universelle* (*Les notions philosophiques*, tome 2, Presses Universitaires de France, 1998) en tant qu'« Ensemble d'énoncés décrivant une action future […]/ Ensemble des opérations à effectuer pour obtenir un résultat », montre bien que le terme, malgré sa précision sémantique, a un large champ d'application.

**3.** Un des premiers recensés s'intitule *Artspeak*. Daté de 1972, il s'agissait essentiellement d'un langage dédié à la communication avec une table traçante.

qui alla en s'amplifiant durant les années suivantes avec le développement du multimédia et des applications « temps réel ». Pendant les années 1980, hormis quelques artistes restés attachés au travail sur le code[4], la plupart préférèrent utiliser des logiciels « pré-programmés » qui bénéficiaient généralement d'une interface visuelle facilitant leur manipulation[5]. « Programmer l'image » devint le pré carré de spécialistes, tel qu'en témoigne le développement de conventions, congrès et conférences autour du sujet. L'une des plus célèbres étant le SIGGRAPH[6], dont les rapports annuels relatent l'avancement des recherches en ce domaine.

Un changement s'opéra néanmoins au tournant des années 1990. L'apparition d'une « poésie en langage Perl » (*Perl poetry*), dès les premières années de la décennie, dénota l'envie de s'approprier différemment le programme, notamment en le considérant comme un écrit pouvant avoir une valeur propre[7]. Parallèlement, l'expansion du marché du CD-ROM et des logiciels multimédias associés à sa conception donna accès à des langages plus faciles et intuitifs, comme Hypertalk (1987) associé à Hypercard ou le Lingo (1988) dépendant du logiciel Director. Quelques années plus tard, le développement à grande échelle d'Internet familiarisa les utilisateurs avec le code par l'intermédiaire, entre autres, du HTML (1989) et du JavaScript (1995). Ce contexte donna naissance à une nouvelle génération d'artistes, souvent autodidactes en matière de programmation, qui s'approprièrent ces langages de script et de haut niveau de manière libre et (ré)créative. À la fin des années 1990, cette attitude s'installa pour devenir une forme d'opposition à l'usage de logiciels « pré-programmés » et s'affirma comme le symbole d'une liberté retrouvée face à l'ordinateur. Grâce à la programmation, certains artistes pouvaient redevenir « maîtres » de la machine et développer intégralement leurs idées. L'apparition de langages « orientés art » tel *Design by Numbers* (*DBN*) de John Maeda en 1999 puis Processing de Casey Reas et Ben Fry en 2001 fut très symptomatique de ce changement embrayé quelques années plus tôt[8]. Ce « retour au code », aux conséquences toujours perceptibles, changea l'approche de l'ordinateur en art.

Bien entendu, ce « retour » est à modérer : il ne concerne qu'une frange mince des

**4.** Comme par exemple Michel Bret, Yoichiro Kawaguchi, William Latham, Joseph Nechvatal…

**5.** Nous ne parlons pas ici de la micro-informatique domestique qui se développe à un rythme différent. La programmation « faite à la maison » est au contraire en plein essor dans les années 1980.

**6.** Le SIGGRAPH (Special Interest Group in Graphics) est une manifestation américaine sur l'infographie, durant laquelle de nombreux professionnels se retrouvent pour présenter leurs recherches et faire état des progrès dans le domaine de l'infographie et de la programmation graphique. Cette manifestation est organisée par l'association ACM SIGGRAPH (Association for Computing Machinery's Special Interest Group on Graphics), établie à New York et dont la fondation remonte à 1969. La première édition du SIGGRAPH s'est tenue en 1974.

**7.** Le langage Perl a été créé en 1987 par Larry Wall dans le but d'effectuer des opérations sur les fichiers textes avec plus d'efficacité et de facilité que ne le permettaient les langages existants. Cette origine permet de comprendre pourquoi Perl a donné naissance, plus que les autres langages, à une forme d'expression poétique. Dès 1991, Sharon Hopkins recensa lors d'une conférence intitulée « Camels and Needles: Computer Poetry Meets the Perl Programming Language » (http://www.cert.or.id/~budi/courses/el2001/plpaper.pdf) quelques-uns des « poèmes en Perl » les plus aboutis. Cependant, comme le rappelle Florian Cramer dans son essai *Words Made Flesh: Code, Culture, Imagination* (2005, http://www.netzliteratur.net/cramer/wordsmadefleshpdf.pdf), la « poésie du code » est apparue avant Perl et ne se limite pas à ce langage.

**8.** Et beaucoup plus tôt encore, comme en témoigne le texte de Miller Puckette concernant Pure Data, présent dans l'ouvrage p. 179-190.



artistes utilisant les « nouveaux médias », qui ne sont eux-mêmes qu'une frange mince de l'art actuel… Il ne faut pas non plus croire que ce « retour » est définitif car la dynamique de la pensée artistique, associée à la rapidité des évolutions techniques, ne peut laisser présager de l'avenir ! Cet élan, en dépit d'une certaine confidentialité, n'est pourtant pas à sous-estimer. Ce « retour » n'est ni une entreprise nostalgique, ni une simple affaire de *geeks*[9] : il relève d'un regain d'intérêt pour la puissance conceptuelle et créatrice d'une pratique (la programmation) avec des outils contemporains (les langages de haut niveau). Et même dans le cas de réutilisation d'anciennes machines[10] ou de langages de programmation obsolètes, le caractère *low tech*[11] de certaines œuvres sert un propos actuel.

Malgré des allures de bidouillage ironique[12] et un manque d'ampleur apparent – vu l'aspect « fait maison » de certaines productions très éloignées des installations spectaculaires que mobilisent, par exemple, les systèmes de réalité virtuelle –, ce « retour au code » est véritablement symptomatique d'un mûrissement des productions artistiques par ordinateur. De la poésie en langage Perl[13] aux *fork bombs*[14], des langages improbables[15] aux logiciels expérimentaux[16], des détournements humoristiques[17] aux interfaces graphiques « remixées »[18], la liste est maintenant longue de ces petits gestes programmés, de ces attaques en règle contre le monde standardisé de l'informatique… Mais le propos ne se réduit pas à ce seul domaine. Même si beaucoup de ces travaux procèdent d'une prise de conscience du « medium informatique », ils ne s'arrêtent pas tous à une simple affirmation de leur « essence numérique » : ils relèvent davantage d'une réflexion sur notre être contemporain en proie aux technologies numériques[19]. Bien sûr, toutes les œuvres ne sont pas porteuses d'une critique sociologique ou politique, certaines sont de l'ordre de l'exercice algorithmique (notamment parmi les productions dites « génératives ») ou de la pure recherche formelle et sonore, mais généralement elles abordent l'informatique avec distanciation et lucidité. L'ordinateur n'y fait pas figure d'accessoire pour émerveiller ou tromper le spectateur, il est passé au crible, mis à l'épreuve… La machine n'est plus seulement au service de l'idée de l'artiste, elle devient

**9.** Le terme *geek* désigne à l'origine une personne monomaniaque d'un domaine, le plus souvent d'ordinateurs, de science-fiction, de jeux vidéo… En français son emprunt est fréquemment limité au seul domaine informatique.

**10.** Comme par exemple *All Wrongs Reversed ©1982* de Jodi étudié par Gilles Rouffineau p. 163-168 de ce livre.

**11.** Low-tech, par opposition à *high-tech*, s'applique à des appareils constitués de montages électroniques simples ou dépassés. Dans le domaine artistique, *low-tech* a perdu son sens péjoratif pour devenir la revendication d'une certaine simplicité technologique.

**12.** *Super Mario Clouds* de Cory Arcangel, analysé dans « Bricodage » p. 317-329, en est un exemple.

**13.** Tel *London.pl* de Graham Harwood, en début d'ouvrage.

**14.** Comme *forkbomb.pl* d'Alex McLean (http://www.runme.org/project/+forkbomb/). Une *fork bomb* vise à saturer, jusqu'au plantage, la capacité de gestion de processus d'un ordinateur en multipliant rapidement une tâche à accomplir.

**15.** Tel *Brainfuck* de Urban Müller (à voir notamment sur http://esoteric.voxelperfect.net/wiki/Brainfuck).

**16.** Voir, par exemple, les productions de Ixi-software (http://www.ixi-software.net/).

**17.** Comme le fameux *Autoshop* d'Adrian Ward (http://www.signwave.co.uk/go/products/autoshop).

**18.** Tel *Wimp* de Victor Laskin et Alexei Shulgin (http://wimp.ru/about.php) ou encore *Scream* de Amy Alexander (http://scream.deprogramming.us/).

**19.** Telle que décrite dans le texte de Matthew Fuller, p. 303-315.

rivale, partenaire, victime ou inspiratrice. Elle est composante de l'œuvre, y compris sémantiquement.

Une partie de ces productions est maintenant regroupée sous l'intitulé « Software Art », une dénomination qui semble exclure[20] les œuvres qui n'intègrent pas de recherches sur le logiciel. En fait, cette appellation – et il en existe nombre d'autres – est essentiellement destinée à affirmer une singularité par rapport au qualificatif très général d'« art numérique ». Le Software Art, en particulier, est enfant indigne de l'art numérique, il en est le parricide dans sa radicalité. Il pose cette question : « Voulez-vous continuer à faire de l'art ASSISTÉ par ordinateur ou voulez-vous faire de l'art AVEC l'ordinateur ? ». Voire « malgré » l'ordinateur puisque « l'informatisation de la société »[21] n'est plus à démontrer… Mais accepter cette informatisation sans objections ne serait-il pas faire preuve d'un manque de discernement ? On peut donc comprendre le Software Art sous l'angle de cette lutte contre « l'ennemi » informatique en retournant vers lui ses propres armes. Mais ce n'est pas le seul. Dans le catalogue du festival Read_Me 2.3, on peut lire une remarque assez éclairante de Florian Cramer relative à une autre lecture du Software Art. Elle est extraite d'une discussion entre Pit Schultz[22], Alex McLean et Olga Goriunova :

> « Vous faites tous les deux [O. G. et A. McL.] référence au Software Art comme étant une pratique sociale spécifique, comme quelque chose qui aurait des affinités avec la culture du logiciel libre, avec la culture *hacker* ou leurs déclinaisons. Pour moi, ce n'est pas seulement ça. Je vois plutôt l'expression "Software Art" comme une perspective particulière d'où regarder l'art […] »[23]

Propos auxquels répond Alex McLean en ces termes :

> « Je n'essayais pas de dire que tout le "Software Art" devait être affaire de programmeurs, mais plutôt qu'il s'agissait du point de départ [de beaucoup d'œuvres]. Nous devrions prendre tous ces points de départ en considération et voir où ça nous mène. »[24]

Cet extrait nous montre bien que le Software Art n'est pas uniforme. Ses hérauts en ont eux-mêmes une définition variable que le site Runme.org tente de « cartographier ». La typologie qu'en propose Olga Goriunova, à partir des œuvres regroupées au sein de Runme.org, prouve d'ailleurs le caractère flou de cette notion[25].

Néanmoins, comme le rappelle l'artiste Christophe Bruno[26], faire de l'art à partir de

**20.** Ce qui n'est pas la réalité puisque le Software Art intègre des œuvres qui vont au-delà d'un simple jeu critique sur les logiciels. Voir les textes de Florian Cramer (p. 103-110), Inke Arns (p. 143-153) et Olga Goriunova (p. 113-140).

**21.** Simon Nora et Alain Minc, *L'Informatisation de la société*, Éditions du Seuil/La Documentation Française, coll. « Points Politique », Paris, 1978.

**22.** Pit Schultz est artiste et co-fondateur de http://bootlab.org/ et de la liste Nettime.

**23.** Amy Alexander, Florian Cramer, Olga Goriunova, Alex McLean, Antoine Schmitt, « Software Art – A Curatorial Fiction or an Artistic Tendency? », résumé de la table ronde tenue le 4 février 2003 à la Künstlerhaus Bethanien de Berlin, modérée par Inke Arns. Retranscription partielle publiée dans Gerrit Gohlke (Sous la dir. de), *Software Art: A Reportage about Source Code*, Künstlerhaus Bethanien, Berlin, 2003, p. 67, http://www.softwareart.net/. Retranscription intégrale : http://www.projects.v2.nl/~arns/Texts/Media/Software_Art_Panel.html

**24.** *Idem*.

**25.** Voir le texte de Olga Goriunova p. 113-140.

**26.** Voir l'entretien avec Christophe Bruno p. 227-238.



logiciels n'est pas forcément synonyme de Software Art. L'appellation « Software Art » s'apparente à une construction théorique, à une « fiction curatoriale »[27] dont la raison d'être est de pointer un supposé phénomène artistique. Bien que défendu avec brio par ses créateurs, le Software Art n'est pas à considérer comme une catégorie établie dans laquelle doivent s'insérer automatiquement ces « nouvelles » productions. C'est une grille de lecture possible de ces pratiques qui a l'avantage de mettre en avant ce regain d'intérêt pour la « programmation » et les principes logiques qui la régissent.

Car ce qui relie ces productions, ce qui les rend identifiables ou plutôt différenciables d'autres productions artistiques, c'est une référence marquée à cette logique programmatique, qui devient méthode de construction de l'œuvre et clef de sa compréhension. Qu'il s'agisse de productions affiliées au « Software Art », d'œuvres « génératives » sur ordinateur, de créations vues en installation, sur Internet ou sur CD-ROM[28], nombre de travaux récents soulèvent cette question du code à divers degrés : pour certains, l'expérience de la programmation prime, quoique peu d'artistes recherchent la virtuosité en ce domaine ; pour d'autres, c'est la sphère plus large du langage, logique, automatisé ou objet de spéculation, qui devient un champ d'investigation. La programmation est donc autant un moyen de production qu'une manière d'élaborer l'idée de l'œuvre.

Il reste à savoir à qui sont destinées ces pièces : à des spécialistes, à des « fans » d'ordinateur ou à un plus large public ?

Concernant le Software Art, il a été précédemment évoqué la position adoptée par Florian Cramer qui souhaite le désenclaver d'une image trop *geek*, parfois trop technique et hermétique. Il relie ainsi les œuvres qui en sont constitutives à l'histoire de l'art[29] avec l'intention de leur donner une portée allant au-delà des problématiques du logiciel libre et du *hacking.* C'est donc clairement une ouverture vers une audience plus diversifiée.

Pour l'ensemble des réalisations issues de ce « retour au code », un facteur générationnel, bien que peu souvent évoqué, entre indéniablement en compte dans l'élaboration mais aussi dans la réception des œuvres. On peut constater que la majorité des artistes et théoriciens à la base de ce « retour » sont pour la plupart trentenaires. Cela peut paraître sans importance, et pourtant, cela correspond à une fenêtre temporelle assez précise dans l'histoire de la micro-informatique. Ces personnes ont connu, dans l'enfance, les premiers micro-ordinateurs grand public (Amstrad, Apple, Commodore, Sinclair…) dont l'accès passait forcément par le code. Un témoignage extrait du célèbre livre de Sherry Turkle, *Les Enfants de l'ordinateur*, paru en 1984, illustre bien cette situation particulière :

**27.** En référence au titre de la conférence « Software Art – A Curatorial Fiction or an Artistic Tendency? », *op. cit.*, 2003.

**28.** Voir à ce sujet le texte de Gilles Rouffineau p. 155-177.

**29.** Voir David-Olivier Lartigaud, « Le Software Art en ses cloisons », dans *Synesthésie* n° 16, 2006, http://copresences.synesthesie.com/imprime.php?texte_id=109

« […] Jarish [12 ans] trouve des programmes de jeux dans des magazines spécialisés et il les programme sur son ordinateur en leur faisant subir quelques modifications, parfois par goût personnel, le plus souvent par accident quand il fait une erreur de frappe. Il éprouve toujours autant de satisfaction à jouer à un jeu de poursuite qu'il a modifié pour le rendre plus familier à son petit frère. […] Mais lorsqu'il fit cette modification, il se produisit quelque chose d'inattendu : "… Tout à coup, il y avait comme cinquante millions de petits points différents sur l'écran. C'était fantastique. […] D'habitude, dans ce jeu, la piste est constituée uniquement par des petits points, mais j'ai effacé cette ligne de programme si bien que maintenant, il y a des effets dingues." »[30]

Durant une décennie, un imaginaire lié à la programmation se développa à travers la presse, la publicité, les médias… Sans affirmer que ce « retour au code » ne produit que des œuvres destinées à ceux qui ont vécu cette période, il est évident que pour les générations à venir, cette approche ludique et intuitive du code sera totalement étrangère. Certaines réalisations perdront donc cette lecture possible, de l'ordre de l'intime, de l'amusement un peu enfantin, qui n'en constitue certainement pas la nature essentielle mais en fait indéniablement partie.

En bref, si des œuvres comme *.Walk* de Wilfried Hou Je Bek sont expérimentables par tous, s'il suffit de se laisser surprendre par les états d'âme du *Pixel blanc* ou la fluidité des algorithmes de Golan Levin, nombre de projets, d'un abord moins facile, sont probablement amenés à devenir encore plus obscurs avec le temps. La « sauvegarde » de ces œuvres est donc à prendre en considération et pas uniquement du point de vue technique ou matériel[31] : c'est leur mode de compréhension qu'il est nécessaire de perpétuer entre « culture numérique » et histoire de l'art.

*ART++* tente de relever en partie ce défi en offrant une sélection de textes-clefs sur le sujet. L'intention de ce livre est d'aider à comprendre non seulement le Software Art mais aussi l'ensemble des productions issues de ce « retour au code » et plus globalement, leur contexte d'apparition. L'ouvrage n'a pas été conçu comme une compilation de « textes de référence » mais comme le reflet d'une pensée vivante et évolutive, celle de la décennie 2000 ; il ne cherche pas à clore le débat mais convie à y participer. Composé de textes originaux ou inédits en français, ce livre est principalement issu des deux colloques Programmation orientée art, organisés dans le cadre de la ligne de recherche « Sens et usage de la programmation en art » soutenue par la Délégation aux Arts Plastiques. C'est paradoxalement le point fort et le point faible de l'ouvrage : point fort car ces deux rencontres ont permis de rassembler nombre d'artistes et théoriciens influant sur cette question, point faible car les absents étaient aussi nombreux. Le parti pris de ne publier que des textes émanant d'auteurs invités à ces colloques est radical mais

**30.** Sherry Turkle, *Les Enfants de l'ordinateur*, Denoël, Paris, 1986 (pour l'édition française), p. 64-65.

**31.** Voir à ce propos le texte d'Anne Laforet, p. 341-351.



peut se justifier par la richesse du contenu que constituent déjà ces interventions. L'ensemble des problématiques abordées est suffisamment conséquent pour donner un juste aperçu des enjeux. De plus, chaque texte de l'ouvrage appelle la lecture d'autres – repris dans une bibliographie générale en fin de livre – permettant ainsi de quadriller une part encore plus grande du champ de réflexion.

Pour mieux guider le lecteur au sein des divers sujets traités, un regroupement en quatre grandes thématiques a été préféré à une organisation chronologique[32]. Ces axes généraux intitulés « Une approche de la programmation informatique en art », « Logiciel et art », « Jeux de code » et « Une culture de la programmation » tentent de cerner les questionnements soulevés dans les textes mais sans la volonté de les rallier absolument à cette bannière. Notons qu'ils n'ont eu aucune influence sur les auteurs puisque constitués *a posteriori* ; il faut donc les considérer comme de simples points d'ancrage pour accéder aux diverses contributions.

« Une approche de la programmation informatique en art » rassemble des textes qui rappellent et approfondissent quelques notions fondamentales sur le sujet : Darko Fritz retrace un pan historique de l'art « programmé » par une présentation du mouvement artistique international « Nouvelles Tendances » et les expositions qui l'accompagnèrent, pionnières dans la présentation d'œuvres par ordinateur. Sylvie Tissot, par son expérience pratique, évoque quelques notions ayant trait à l'essence du travail de programmeur. Jean-Paul Fourmentraux, en croisant les témoignages de différents artistes, dépeint par touches le cadre général de réflexion de l'artiste confronté à la programmation. L'accent est ensuite mis sur les projets d'Antoine Schmitt et la manière dont il conçoit l'acte de coder. Cette partie s'achève sur un texte collaboratif rédigé par Geoff Cox, Alex McLean et Adrian Ward, réflexion esthétique sur la pratique de la programmation en forme de réexamen d'un de leurs écrits plus ancien, à savoir « L'esthétique du code génératif »[33].

« Logiciel et art » est évidemment axé sur le Software Art sans s'y limiter tout à fait. Andreas Broeckmann, en tant que directeur de Transmediale, fut acteur et témoin majeur de la création et de la diffusion du Software Art. Il propose donc une analyse du contexte de son développement qu'il inscrit dans une réflexion plus large sur la place du logiciel en art. Vient ensuite le « manifeste » du Software Art, rédigé par Florian Cramer, un texte jalon ayant fait l'objet de multiples débats comme l'implique généralement ce genre d'écrit. Olga Goriunova livre, pour sa part, un témoignage unique sur le Software Art en tant que membre fondatrice de Runme.org. L'histoire de ce site met à jour les nombreuses idées

**32.** La chronologie des textes est importante pour comprendre la progression des débats et les sujets qui, l'espace d'un moment, focalisent l'attention. En effet, pendant certaines périodes d'effervescence (Transmediale 01, Ars Electronica 2003…), les échanges, les réactions et les textes apparurent et circulèrent vite, notamment sur la question du Software Art. Néanmoins, comme le présent ouvrage répond à la logique thématique d'une ligne de recherche, la chronologie nous a paru moins essentielle que les questions étudiées par les auteurs, d'où l'organisation du livre en quatre grands axes de réflexions.

**33.** Geoff Cox, Adrian Ward, Alex McLean, « L'esthétique du code génératif », trad. de l'anglais par Damien Subotticki, 2009. La traduction de ce texte est consultable sur le site des Éditions HYX : www.editions-hyx.com.

et stratégies mises en place pour donner un cadre à ces pratiques. Suit le texte d'Inke Arns qui s'apparente à une analyse critique de la « générativité » en art ; bien qu'il soit à rapprocher directement de celui de Geoff Cox, Alex McLean et Adrian Ward, ce texte offre une stimulante différence de point de vue. Inke Arns approfondit notamment la notion de « performativité du code » et opère une distinction en art génératif et Software Art. Puis un regard plus extérieur nous est apporté par Gilles Rouffineau qui met en perspective la notion de Software Art en la confrontant à des œuvres antérieures pour une sorte d'archéologie du logiciel en tant qu'art. À distance du Software Art mais traitant directement du rapport art/logiciel, Miller Puckette, quant à lui, expose les questionnements auquel il est sujet en tant que concepteur d'un logiciel comme Pure Data.
« Jeux de code » débute par un texte de Nathalie Magnan sur l'« hacktivisme » qui, comme elle le précise, représente un bastion critique essentiel au sein de notre « société du code ». Suit un entretien avec Christophe Bruno qui montre que son travail ne se limite pas à débusquer les codes de notre société mais incite plutôt à réfléchir aux raisons sous-jacentes qui ont amené à les créer. Samuel Bianchini prend comme étude de cas une de ses œuvres pour passer au crible les notions d'adressage, de données, d'information ou de programme lorsqu'elles sont destinées à l'image interactive. Puis Karen O'Rourke, dans un texte et deux entretiens avec Jo Walsh et Wilfried Hou Je Bek, nous entraîne dans une déambulation entre psychogéographie, algorithmes, réseaux et GPS, où les cartes numériques deviennent beaucoup plus grandes que le territoire.
Le quatrième axe, « Une culture de la programmation » met en lien des réflexions touchant à notre monde fait d'algorithmes, de données, de statistiques et de réseaux. Ces textes, entre esthétique, critique et politique, appellent à s'interroger sur la manière de comprendre, notamment par le biais de l'art, notre quotidien « numérique ».
Matthew Fuller brosse le portrait des « monstres mathématiques » que nous sommes devenus. Son texte s'appuie notamment sur l'œuvre *London.pl* de Graham Harwood, traduite et reproduite en début d'ouvrage. La notion de « bricodage », évoquée ensuite, est une tentative de description d'une approche sensible du code par le bricolage et l'ouverture de « l'objet technique ». Puis Nicolas Thély nous invite à une traversée éclair de notre quotidien informatisé en nous montrant comment certains artistes l'assimilent et le restituent à l'aide de pinceaux, d'algorithmes ou de Patafix. Anne Laforet, dans une analyse qui aborde la pérennisation des codes autant que les difficultés d'émulation, nous confronte au problème de la conservation des œuvres programmées. Suit une étude de Wendy Hui Kyong Chun qui établit un parallèle entre idéologie et logiciel sous forme d'un véritable appel à la vigilance face à leur « transparence » et leur convivialité. Écho critique direct aux propos de Wendy Hui Kyong Chun, le texte d'Alexander Galloway revient sur le terme « idéologie », sur la notion de non « visible » en programmation et replace le logiciel au sein du politique, non pas comme porteur de messages, mais comme pièce constitutive du « réseau de relation » de notre société. Pour conclure,



le texte de Geert Lovink est une invitation à penser différemment le réseau, en pourfendant certains clichés, comme celui d'une représentation cartographique « automatisée », au profit d'une « esthétique qui engloberait la complexité sociale ».
*ART++*, on l'aura compris, embrasse un large éventail de thèmes et de questionnements amenés, pour la plupart, par des artistes et des théoriciens invités à réfléchir à la « programmation informatique en art ». On constatera donc sans équivoque que ce sujet ne s'arrête pas aux frontières de la technique, il se traite même essentiellement ailleurs.
Cela explique pour beaucoup l'éventuelle impression de « bric-à-brac » émanant des œuvres issues de ce « retour au code ». Déstabilisantes pour le critique et le spectateur, parfois hermétiques, souvent trop ludiques « pour être sérieuses », posant d'innombrables problèmes de conservation, elles font preuve d'une remarquable liberté d'expérimentation qui montre, et c'est heureux, bien peu de certitudes quant à ce que devrait être « l'art », qu'il soit ou non par ordinateur.