PPH: Le code est-il de l'art ?

Problématique

Quelles sont les possibilités de faire de l'art avec du code ?

Florian Rascoussier

INSA
INSTITUT NATIONAL
DES SCIENCES
APPLIQUÉES
LYON

Projet Personnel en Humanité

Institut National des Sciences Appliquées de Lyon (INSA Lyon)

Rapport de soutenance

Semestre d'été 2022

Problématique

Quelles sont les possibilités de faire de l'art avec du code ?

Autheur:

Florian Rascoussier 4IF - INSA Lyon

Tuteur:

Lionel Brunie, Professeur et Directeur du Département d'Informatique de l'INSA de Lyon (IF), chercheur au laboratoire LIRIS

Jury:

Lionel Brunie

Abstract

Ce rapport de projet présente un ensemble de recherches et de réflexions personnelles pour tenter d'explorer la pratique de la programmation et le code qui en résulte sous l'angle de la création artistique. L'objectif est de comprendre ce qu'est le code et comment il peut être concu et créé de manière artistique. Il s'agit donc de saisir en quoi le code est-il de l'art.

En outre, le code étant un outil à même de former des images, des mouvements ou des sons, on ne s'intéressera pas à la manière de créer des oeuvres d'art en utilisant du code. Il s'agit de se focaliser sur le code et la programmation elle-même, et non pas sur le produit ou résultat de ces derniers.

Remerciements

Remerciements aux encadrants de ce projet, à savoir:

- **Lionel Brunie** - <lionel.brunie@insa-lyon.fr>: Encadrant final du PPH.
- **Gonzalo Suarez Lopez** - <gonzalo.suarez-lopez@insa-lyon.fr>: Premier encadrant du PPH.

Une première ébauche de ce rapport à d'abord été rédigé dans les semaines précédentes à une séance de débat. Celle-ci a permise de réunir quelques personnes du Département Informatique de l'INSA Lyon ; afin de discuter de la problématique et prendre connaissances des recherches préalables avant de débattre sur le sujet et de proposer de nouvelles pistes de réflexion.

Remerciements particuliers aux participants de la séance de débat, qui à eu lieu le Jeudi 7 avril 2022 (10-12h):

- **Eric Guérin** - <eric.guerin@insa-lyon.fr>: Professeur, chercheur au laboratoire LIRIS.
- **Anicet Nougaret** - <anicet.nougaret@insa-lyon.fr>: Etudiant IF 4ème année.
- **Ithan Velarde Requena** - <ithan.velarde-requena@insa-lyon.fr>: Etudiant IF 3ème année.

Avant propos

Ce projet de PPH est titré par la question «Le code est-il de l'art ?». Ce titre décrit bien l'objectif et le cadre général de ce PPH mais couvre un vaste champs de réflexion qui n'est pas réellement représentatif du point de vue choisie. En effet, le code étant le produit d'une pratique, il semble assez direct de dire qui puisse être l'object d'une création ou d'une pratique artistique. Ainsi, la problématique réelle est donc «Quelles sont les possibilités de faire de l'art avec du code ?». Cette question est donc ciblée sur les pratiques, les formes et les moyens de création artistique ayant le code pour object.

La réponse à la problématique ne pouvant passer que par la compréhension du sujet global, il va donc falloir consacrer les 2 premières parties du rapport à discuter de ce qu'est le code, et de la nature de l'art, avant de réellement pouvoir aborder les différents axes de création artistiques, et les pratiques connexes que l'on peut relier au code et à la programmation. En effet, art comme code sont des concepts flous qui cachent une multitude des réalités et concepts différents. Bien qu'il ne sera pas possible d'en donner une définition précise et complète, discuter de ces concepts est néanmoins nécessaire pour se doter des outils de compréhension et de réflexion utiles pour la suite du rapport.

Enfin, il semble important de rappeler que ce rapport est issue d'un projet de PPH, c'est à dire d'un travail personnel de $4^{\text{ème}}$ année, sans créneau horaire réservé dans la formation, et valorisé au minimum d'un unique crédit ECTS. Il est donc important de considérer que ce rapport n'a en aucune manière l'ambition de faire un tour exhaustif de la question auquel il tente d'apporter un éclairage.

Contents

1 Introduction

Nous verrons d'abord comment la pratique de la programmation et le code qui en résulte ont évolué afin de comprendre en détail qu'est ce que le code. Nous nous intéresserons ensuite à l'art, et la relation entre le code et ce dernier. Enfin, nous exploreront les diverses méthodes, pratiques et techniques qui font du code un art à part entière.

2 Qu'est-ce que le code ?

Comprendre le code est la première étape indispensable afin de pouvoir considérer le problème du rapport entre celui-ci et l'art. Le terme n'est pas nécessairement bien connoté dans la sphère de l'informatique de même que l'on ne parle pas de rond mais de cercle en géométrie cartésienne. Pourtant, le terme est intéressant en cela qu'il à le mérite d'être en apparence simple et donc à même de parler à chacun. Il faut cependant tenter d'éclaircir le terme et d'en apporter une définition satisfaisante.

2.1 Programmation, code et langage

La programmation est un terme générique. Composé du suffixe *-ation*, du latin *-atio*, utilisé pour signifier une action, et du nom *programme* lui-même issue via le latin *programma* du grec ancien πρόγραμμα. qui peut désigner divers concepts selon le domaine, par exemple le terme *programmation* désigne, dans le milieu du cinéma, l'action de déterminer les programmes ou films d'une salle donnée. Bien sûr, le terme est ici considéré sous l'angle des sciences informatiques. «Pour éviter cette confusion polysémique, des termes alternatifs sont parfois utilisés, comme le code ou le codage, mais la programmation informatique ne peut pas se réduire au code» [3]. D'après le Larousse, la programmation est donc: «

- l'action de programmer, c'est-à-dire fournir à un ordinateur les données et les instructions conscernant un problème à résoudre, une tâche à effectuer, etc.
- l'établissement d'un programme, c'est-à-dire une séquence d'instructions et de données enregistrée sur un support et susceptible d'être traitée par un ordinateur.

» [2]. Cette définition n'est pas parfaite mais pose les éléments essentiels. Elle montre que la programmation est une action, un acte qui suppose des considérations techniques (quelle machine, quelle format, quels instructions), physique ou structurelles (comment l'exécution est effectuée, comment fournir les instructions à la machine, quel support), et un but (quelle tâche, pour quoi faire). La programmation est donc une activité créatrice et créative. On comprends aussi que le code est l'object de cette création, ce qu'elle produit.

Le code est donc un object issue d'un acte de production. En cela, la programmation relève donc de la technique tandis que le code est de l'ordre de la production. Le mot *code* est cependant lui aussi fortement polysémique. En effet le terme est issue du latin *codex*, variante de *caudex* signifiant «tronc». Le terme est cependant fortement associé aux livres et particulièrement aux ouvrages juridiques. Aujourd'hui, le terme peut prendre divers sens selon le domaine considéré, du droit à la linguistique en passant par l'informatique ou la cryptographie. Le Larousse distingue ainsi de nombreux sens différents pour le mot, dont les plus intéressantes du point de vue du sujet sont les suivantes: «

- Ensemble de règles qu'il convient de respecter.
- Système de symbole permettant d'interpréter, de transmettre, un message, de représenter une information, des donnée.
- Système conventionnel, rigoureusement structuré, de symboles ou de signes et de règles combinatoires intégrées dans le processus de la communication.
- Ensemble d'instructions écrites dans un langage lisible par l'homme et devant être traduites en langage machine pour être exécuté par un ordinateur.

» [2]. Toutes ces définitions sont intéressantes, notamment en considérant le point de vue d'un non informaticien. On comprend que le code désigne donc un moyen de transmettre de l'information via un système de règles, mais aussi les règles elle-mêmes.

On considèrera tout de même quelques définitions du Wikitionnaire: «

- Le code source est un texte qui représente les instructions de programme telles qu'elles ont été écrites par un programmeur.
- Le langage machine, ou code machine, est la suite de bits qui est interprétée par le processeur d'un ordinateur exécutant un programme informatique.

- Système de symboles permettant de représenter une information dans un domaine technique (code binaire, alphanumérique, morse).

» [4]. Ces définitions viennent compléter la définition du code. Du point de vue de l'informatique, le code est donc un ensemble d'instructions et les instructions elles-mêmes qui sont alors dépendantes d'un langage, système et niveau d'abstraction considéré. En effet, tout code ne peut être écrit qu'en respectant des normes et des règles afin d'être finalement exécutable par une machine. Il est le fruit d'un acte de création, d'un travail d'écriture et de conception. Le code est donc fortement lié aux règles qui le définissent.

Cet étallage de définitions permet donc de saisir la nature profondément polysémique du terme et son lien étroit avec la notion de programmation. On peut donc catégoriser les différents sens qui seront utiles dans la suite de ce rapport:

- Séquence instructions et de données qui constituent un programme et qui ont vocation à être exécuté par un ordinateur. Il s'agit d'un sens général qui englobe une variétés de représentation, du *code source* aux Application-specific integrated circuit (ASIC).

- Système de règles, de symboles et de conventions permettant de représenter une information. Il s'agit d'une considération générale de ce que peut désigner le terme. On pensera notamment aux langages informatiques et langages de programmation qui sont des éléments essentiel à la pratique de la programmation moderne et qui déterminent fortement la forme que prendra le code source.

- On désigne par *code source*, l'ensemble des documents qui forment un programme avant toute forme de traitement. Ce code est alors réparti dans un ou plusieurs textes et documents écrits qui peuvent être rédigés et lus par un humain. Ils servent de base à divers procédés permettant de passer de ces documents à un ensemble de données et d'instructions exécutés par un ordinateur.

- Le *code machine* désigne les instructions, généralement après traitement, qui sont donc directement exécutées par le processeur d'un ordinateur. Elle n'ont pas vocations à être écrites ou lues par un humain, et sont dites *de bas niveau*. Elles sont également fortement dépendantes du jeu d'instruction, c'est-à-dire des instructions machines éxécutables par le processeur considéré. Cette élément explique entre autre le besoin de disposer de systèmes de représentations de plus hauts niveaux, indépendants de la notion de jeu

d'instruction. On notera que l'*assembly*, c'est à dire du code machine écrit sous une forme lisible par un humain n'en est pas stricto sensu une forme alternative puisqu'il représente la même chose, mais nécessite quand même une conversion.

- Enfin, on introduira la notion de *code-idée*, c'est à dire l'algoritme, l'objectif derrière le code source ou machine, l'idée indépendante de tout langage. Cette idée traverse les différents niveaux de code et nait en premier lieu dans l'esprit du programmeur lorsqu'il écrit ou lit un code source. Cette idée peut s'exprimer en une multitude de formes selon le choix du langage utilisé. D'une certaine manière, le choix du langage va cependant impacter la forme prise par l'idée et il y a donc une distinction entre l'idée, la forme et l'exécuion effective.

Pour résumer, on peut considérer que le code est un moyen de transmettre des informations, mais aussi de les représenter. Ces informations constituent un programme, fruit du travail du programmeur et conçu avec un objectif praticulier. Pour cela, il est nécessaire de respecter des règles généralement sous la forme d'un langage de programmation. Afin de mieux comprendre l'évolution de ces concepts et leurs différentes formes, il convient de s'intéresser à l'histoire.

2.2 Petite Histoire du code

Remonter dans l'histoire du code et de la programmation est important pour comprendre ses origines, ses différentes formes, et ses évolutions. La programmation est une technique qui permet définir des règles et des comportements sans avoir à changer le système physique qui l'utilise. Selon la définition que l'on se donne, et les exemples que l'on choisi de considérer, on peut remonter jusque dans l'antiquité avec les automates d'Héron d'Alexandrie [20]. L'idée originale étant de pouvoir définir un système capable de modifier son comportement sans modifier ses composants physiques dans le but évident de multiplier les comportements sans avoir à changer le système.

La première invention majeure remonte cependant au début du XIX$^{\text{ème}}$ siècle, avec l'invention du métier à tisser Jacquard, en 1801 à Lyon [17]. Son invention, inspirée des orgues de barbarie, permettait de lever automatiquement les fils de soie nécessaire à la réalisation de motifs. Motifs pouvant être modifiés grâce à un système modulaire de cartes perforées et d'un mécanisme en carré mobile [18]. Ce système est donc l'ancêtre direct des cartes perforées encore en usage

au XX$^{\text{ème}}$siècle, et successivement amélioré par les équipes de Sir Thomas Watson et sa société IBM. Passant par étape de 22 à 80 colonnes et de 8 à 10 lignes, la fameuse *IBM card* a permis à l'entreprise de se développer fortement dans la première moitié du XX$^{\text{ème}}$ siècle, en restant pendant près de 40 ans le moyen par excellence pour stocker, transmettre et traiter des données [19].

Entre temps, il est important de regarder les developpement de Charles Baddage et ses travaux sur les machines calculatoires et analytiques qui font qu'il est souvent considéré, non sans raison, comme le père de l'ordinateur [21]. Inspiré par le métier Jaquard, ce professeur de l'université de Cambridge a d'abord créé le *Difference Engine*, une machine entièrement analogique, composée de pièces mécaniques et capable de calculer automatiquement des tables de calculs pour certaines fonctions mathématiques comme le logarithme, ou encore pour le calcul automatique des marées. Bien qu'il n'acheva jamais sa machine, il proposa une nouvelle idée d'une machine généraliste qui aurait eu sa propre unité centrale de calcul et sa mémoire.

Dans les notes accompagnants la traduction de notes d'un séminaire donné par Baddage en 1840, Augusta Ada King, comptesse de Lovelace, ou simplement Ada Lovelace proposa une méthode de calcul automatique de la suite des nombres de Bernouilli sur la machine analytique, sous forme d'un diagramme et de notes. Cette méthode décrite dans la note G, qui se distingue du pur calcul scientifique jusque là envisagé, notamment par Baddage, et est souvent considéré comme le premier véritable programme informatique [22]. Ada Lovelace est ainsi considéré comme la première programmeuse de l'histoire.

Il faut cependant attendre le milieu du XXème siècle, et notamment les travaux du mathématicien anglais Alan Turing et son article fondateur de la science informatique, pour que cette technique se développe réellement "On computable numbers, with an application to the entscheidungsproblem". La programmation allant de paire avec le développement et la monté en puissance des ordinateurs, on passe progressivement des programmes simples réalisés physiquement sur circuit électroniques, aux programmes sur cartes perforées, avant que ne se développent les premiers langages de programmations. Ces langages ont largement impacté la pratique de la programmation du fait du cadre et des outils qu'ils donnent aux programmeurs.

2.3 Le Langage

Le terme *langage* est souvent associé en premier lieu à une faculté intrinsèque de l'homme. En informatique, il désigne la façon dont instructions et données sont

codées et de les manipuler. Le langage a enfin un sens différent du point de vue de l'artiste. Aborder le sujet du langage et du code requiert ainsi de bien comprendre les notions couvertes par le terme en question.

Pour le dictionnaire Larousse, le langage à donc de multiples définitions: «

- Faculté propre à l'homme d'exprimer et de communiquer sa pensée au moyen d'un système de signe vocaux ou graphiques; et ce système.
- Système structuré de signes non verbaux remplissant une fonction de communication.
- Ensemble des procédés utilisés par un artiste dans l'expression de ses sentiments et de sa conception du monde.
- Mode d'expression propre à un sentiment, à une attitude.
- Ensemble de caractères, de symboles et de règles permettant de les assembler, utilisé pour donner des instructions à un ordinateur.
- (machine) Langague directement exécutable par l'unité centrale d'un ordinateur, dans lequel les instructions sont exprimées en code binaire.

» [2]. On remarque que ces définitions recoupent en parties celles que l'on a pu voir dans les parties précédentes au sujet du code. Code et langage sont ainsi très lié. Du point de vue du programmeur, le langage est en effet la langue qui définit comment organiser les instructions et les données afin de produire un programme [6].

Un langage de programmation est un outil pour le programmeur. En effet, tout programme étant par définiton une suite d'instructions machines c'est-à-dire de 0 et de 1, il n'est pas aisé pour un humain d'écrire ses programmes directement dans ce langage bas niveau. Cependant, il est possible d'écrire des programmes dans des langages compréhensible par un humain mais qui puisse être tout de même être transformé en langage machine pour être exécuté par un ordinateur. De mêmes que les langues parlées, il existe une multitude de langages de programmation. Comme le dit Amade, «Il existe des pratiques de programmation très différentes, il existe aussi des langages de programmation qui proposent des modes d'approche très différentes» [6]. La variété des premières expliquant naturellement la multiplicité des seconds, en plus d'autres facteurs comme les évolutions de la recherches, l'histoire jusque même aux goûts, habitudes et usages des programmeurs.

Le choix du ou des langages que l'on souhaite utiliser pour écrire un programme est donc un aspect important. Comme l'écrit Dijkstra dans son livre *A Discipline of Programming* en 1976: «[...] one is immediately faced with the question: Which programming language am I going to use ?, and this is not a mere question of presentation! A most important, but also most elusive, aspect of any tool is its influence on the habits of those who train themselves in its use. If the tool is a programming language, this influence is, -whether we like it or not- an influence on our thinking habits.» [5].

La création d'un langage de programmation est en elle-même un tâche complexe. En effet de très nombreux éléments rentrent en comptent dans la création d'un langage. Comment les rappellent les auteurs de *Introduction à la théorie des langages de programmation*, «Nous sommes encore très loin d'avoir trouvé un langage de programmation définitif. Presque chaque jour, de nouveaux langages sont créés et de nouvelles fonctionnalités sont ajoutées aux langages anciens. [...] La première chose que l'on doit décrire, quand on définit un langage de programmation, est sa syntaxe. Faut-il écrire x := 1 ou x = 1? Faut-il mettre des parenthèses après un if ou non ? Plus généralement, quelles sont les suites de symboles qui forment un programme? On dispose pour cela d'un outil efficace : la notion de grammaire formelle. À l'aide d'une grammaire, on peut décrire la syntaxe d'un langage de manière qui ne laisse pas de place à l'interprétation et qui rende possible l'écriture d'un programme qui reconnaît les programmes syntaxiquement corrects. Mais savoir ce qu'est un programme syntaxiquement correct ne suffit pas pour savoir ce qui se passe quand on exécute un tel programme. Quand on définit un langage de programmation, on doit don également décrire sa sémantique, c'est-à-dire ce qui se passe quand on exécute un programme. Deux langages peuvent avoir la même syntaxe mais des sémantiques différentes.» [8]. Un langage de programmation est ainsi composé plusieurs éléments qui permettent à une machine de comprendre et d'excuter le programme sans ambiguité.

En plus ces considérations, les langages se définissent par la ou les grands approches qu'ils choisissent de considérer ou favoriser. Ce sont les paradigmes. Dans la conférence intitulée *L'importance des langages en informatique* en Nov. 4, 2015 à l'INRIA de Rennes, Berry reviens sur le processus créatif derrière la création des nombreux langages. Au départ, on peut considérer deux langages A et B différents avec des approches fondamentalent uniques l'un par rapport à l'autre. Chacun forme sa propre communauté d'utilisateur. Puis arrive un nouveau langage C qui ne propose pas un nouveau point de vue mais reprend les concepts des deux langages précédents pour tenter tant bien que mal de les associéer. Ce dernier viens grapiller des utilisateurs aux deux premiers. Les langages A et B originaux se mettent donc à incorporer des concepts de l'autre afin de récupérer des utilisa-

teurs et venir proposer de nouvelles améliorations à ses utilisateurs.«C'est-à-dire que vous avez des tas de gens qui ont des idées complètement baroques. Vous avez des tas de groupes d'utilisateurs avec des traditions totalement différentes dans des pays qui n'ont rien à voir [...] et la paysage deviens très joyeusement incompréhensible. Et c'est là vie, parce que c'est un espace de création. Difficile.» [7]. Berry propose ainsi une vision caricaturale mais néanmoins descriptive derrière l'apparition et l'évolution des langages informatiques.

Le langage, support essentiel de la programmation moderne, est donc en lui-même un «espace de création» [7] qui n'est pas sans influence sur le code en lui-même [5]. Au contraire, il représente un élément essentiel du code, de par la structure, la forme et finalement la philosophie qu'il impose nécessairement de considérer. Nous devrons donc explorer ce que cela implique sur le plan artistique. Cependant, il faut d'abord nous intéresser à la question de l'art lui-même.

3 Le code est-il de l'art ?

Une fois que la difficile question de la définition du code a été discutée, il reste encore la non-moins difficile question de la définition de l'art. La partie suivante tenter de monter toute la complexité de l'entreprise en essayant néanmoins d'établir un cadre utile pour les réflexions futures.

3.1 Un problème de définition

Bien difficile est le problème de la définition de l'art. Faut-il tenter d'en donner une version précise, chercher à décrire ce qu'il n'est pas ou encore tenter d'apporter une certaine idée de la pensée qui l'inspire et le fait naître ? C'est que le terme peut facilement changer de sens d'une personne, d'un groupe, d'un pays ou d'une culture à l'autre. Ainsi, il n'y a pas de définition précise de l'art [15]. Plutôt que chercher à en donner une définition, il vaut donc mieux se concentrer à sa compréhension et l'évolution de ses transformations.

Pour Davies dans *Definitions of Art*, l'art est une notion à la perception évolutive en fonction des époques et des auteurs: «In the past, art has been variously defined as imitation or representation(Plato 1955), as a medium for the transmission of feelings (Tolstoy 1995), as intuitive expression (Croce 1920) and as significant form (Bell 1914). Judged as essential definitions, these are unsatisfactory.» [15].

La définition de l'art et la distinction entre art et non art a historiquement été controllée par les instances de pouvoir, politiques et religieuses mais aussi par les inovations des artistes eux-mêmes. Par exemple, l'artiste Marcel Duchamp, bien connu pour ses *ready-mades* et notamment sa *fontaine* qui n'est qu'un simple urinoir, se fait connaître en montrant que c'est l'idée et la démarche qui vont compter. En d'autres termes, ce qui va définir qu'un objet est artistique, n'est pas l'objet lui même mais l'idée que celui-ci représente une forme d'art. Dès lors que l'artiste le décide, et que le spectateur le regarde comme tel, alors l'objet sera considéré comme artistique comme l'explique Arthur Danto dans *La transfiguration du banal : une philosophie de l'art* [23].

Malgré de ses évolutions, l'art reste un moyen d'expression, un langage à part entière avec des élaborations formelle et une force expressive signifiante. Il sert donc à communiquer comme illustré par le travail et les recherches de nombreux artistes. Par exemple, Christopher Pillault, artiste contemporain atteint d'autisme, qui se sert de l'art pour communiquer à défaut de pouvoir s'exprimer oralement. L'art peut permettre de transmettre des informations claires et ordonnées. L'art est un langage métaphorique, composé de connotations et de symboles, et qui s'appuie sur des analogies.

La question de la définition de l'art pose en retour de nombreuses interrogations connexes. Parmi celles-ci se trouve celle de l'oeuvre. Une oeuvre se défini comme étant une chose matérielle ou immatérielle fabriquée par l'homme mais n'ayant aucune utilité fonctionnelle d'après le philosophe contemporain Heidegger, expert de la pensée rationnelle et de la mécanique phylosophique. On peut citer l'exemple du tableau de René Magritte *La trahison des images (Ceci n'est pas une pipe)* qui cherche à représenter l'objet mais pas à l'utiliser, comme l'énonce la maxime «Ceci n'est pas une pipe» du tableau[14]. Le tableau en lui-même étonne, en jouant sur les règles implicites de la pensée et du rapport entretenu entre les choses, concepts et représentations. Pour le *Définitions : oeuvre - Dictionnaire de français Larousse*, une oeuvre au sens relatif à l'art est une «production de l'esprit, du talent ; écrit, tableau, morceau de musique, etc., ou ensemble des productions d'un écrivain, d'un artiste : Les œuvres de Bach. Une œuvre d'art. Une thèse sur l'œuvre de Rimbaud.» [26]. Cette définition est plus générale et ne se rattache pas directement à la notion d'art. Il s'agit plutôt d'une production, d'une création issue d'une pensée d'un auteur, d'un créateur ou d'un artiste.

On le comprend, l'histoire de l'art est pleine de chamboulements et de transformations de la compréhension de ce qu'il est censé être. On peut alors penser que l'art ne peut alors être défini que par sa constance inhérente au changement et à la transformation. C'est le cas de Marcus Steinweg dans "Nine Theses on Art"

qui explique que «art asserts a consistency owed to its opening to inconsistency» (l'art affirme une consistance due à son ouverture à l'inconséquence). En abordant dans son essai que toute définition considérée, chacune est arbitraire puisqu'elle s'attache à ne prendre en compte qu'un ensemble restreint d'inconsistence dans une certaine perception de l'art, il montre que toute définition qui n'aurait pas pour sujet cet inconsistence même de l'art est nécessairement lacunaire donc arbitraire. Cette définition de l'art semble alors n'être pas réfutable en se rapprochant à sa manière du *conatus* de René Descartes, de l'idée que toute chose est vouée au changement. Mais cette définition est-elle réellement satisfaisante en dépit d'être difficilement réfutable ?

Définir une relation entre art et oeuvre est donc important mais là encore difficile. L'art étant par essence un langage mystérieux, influencé à la fois par la pensée de l'artiste et celle du spectateur, celui-ci est donc source de questionnement. Il invite au dialogue intérieur, politique ou social de par la variété des interrogations, des ressentis et des réactions qu'il suscite. Du fait qu'il convoque les sentiments, l'art et la perception de l'art est avant tout personnelle. Sans tomber dans l'écueil du relativisme, il faut pourtant reconnaître que la définition de l'art est donc flexible et surtout dépendante de l'individu dont la subjectivité offre en elle même une certaine légitimité dans la compréhension, l'apréciation et la création d'art.

3.2 Définir Art et Œuvre

Définir l'art, de même que définir la relation entre art et oeuvre est difficile. La partie précédente, bien qu'elle ait tentée à sa manière de présenter l'étendue des possibles et la complexité des réponses n'a pas permise de dégager un concensus. Afin d'éviter de tomber dans le relativisme, il va donc falloir néanmoins se donner des règles et des définitions un minimum satisfaisante.

En reconnaissant l'impossibilité de définir consensuellement les concepts d'art et d'oeuvre, il semble cependant correct de choisir un ensemble de définitions personnelles.

C'est ainsi que l'on définira pour la suite le concept d'*art* de plusieurs manières. On peut d'abord s'inspirer des définitions courantes:

- «Méthode pour faire un ouvrage, pour exécuter ou opérer quelque chose selon certaines règles.» [25]
- «Reproduction par la main de l'homme ou la représentation de ce qui est

dans la nature ; par opposition à naturel.» [25]

- «Ensemble des procédés, des connaissances et des règles intéressant l'exercice d'une activité ou d'une action quelconque.» [27]
- «Manière de faire qui manifeste du goût, un sens esthétique poussé» [27]
- «Création d'objets ou de mises en scène spécifiques destinées à produire chez l'homme un état particulier de sensibilité, plus ou moins lié au plaisir esthétique» [27]

Ces définitions sont intéressantes de par leur simplicité et leur rapport avec la réalité concrète.

En plus de ces définitions de surface, on utilisera nos propres définitions. Avant cela, on se donnera l'expression *auteur-créateur*, toute personne humaine physique qui utilise l'ensemble de ses connaissances et facultés dans un processus de création d'une chose. Ainsi, l'*art* est donc:

- Etat qui naît de la volonté d'un auteur-créateur du fait des caractéristiques qu'il donne à l'objet de sa création. Il naît d'une maîtrise technique, d'une volontée particulière et d'un parti pris.
- Méthodes, cadre utilisés par un auteur-créateur pour créer un objet ou une mise en scène.
- Elément considéré comme ayant des caractéristiques particulières qui font naître des sentiments chez un spectateur. Il suppose un dialogue entre oeuvre et spectateur et des réactions chez ce dernier.
- Pratique qui nécessite de faire des choix qui aboutissent à l'établissement une création immatérielle ou réelle, éphémère ou pérenne dont la finalité peut être utile ou non mais qui peut être percue par un spectateur.

La notion d'art suppose en effet un dialogue. Dialogue d'abord entre un artiste et lui-même. Puis entre ce dernier et un public ou un spectateur imaginé. Et enfin entre l'oeuvre elle même et le spectateur.

Ainsi, dans notre cadre, une *œuvre* est donc simplement:

- Une création considérée comme artistique, c'est à dire qui tendent à considéré que la création est de l'art pour le spectateur.

- Un object qui peut être apprécié pour ses caractéristiques et qualités du fait d'un processus de création particulier.

On remarquera que les sens d'oeuvre et d'art se recouvent en partie puisque une oeuvre peut être elle même de l'art.

Il est important de rappeler qu'aucune de ces définitions n'est satisfaisante par elle-même. En outre, la somme de celles-ci ne l'est pas non plus. Elle est cependant utile dans le cadre précis de la compréhension du rapport entre programmation et code.

3.3 Programmation et code

Une fois que l'on s'est fixé un cadre suffisant pour comprendre les notions de programmation, code, art et oeuvre, on peut alors dire simplement que **le code est à la programmation ce que l'oeuvre est à l'art**. Cette phrase simple condense les parties précédentes en une maxime qui ne doit pas faire oublier les réserves et les problèmes mis au jour plus tôt.

On peut alors dire que **le code peut être de l'art dans la modalité que le code peut être oeuvre et qu'oeuvre peut être art**. Cette deuxième maxime est à comprendre dans les réflexion de la sous-partie précédente et notamment sur la relation sémantique entre les termes *art* et *oeuvre*.

On peut comprendre ces deux maximes de différentes manières en fonction des définitions considérées: Le code en tant que système de règles ou en temps que langage est donc de l'ordre de la méthode. En ce sens, dire que *le code est de l'art*, c'est dire que un ensemble de contraintes et de règles est en soi un art puisque art peut être cadre et méthode. C'est aussi impliquer qu'un système de l'ordre d'un langage de programmation, du fait des choix de conceptions et d'écriture qu'il impose est en lui-même de l'art, que la création de langages de programmation est aussi un art et que le cadre offert par le langage lui-même peut être considéré comme artistique. Il convient tout de même de rappeler les précautions qui s'impose: cela ne veut pas dire que tout langage de programmation est art. Il s'agit plutôt de réaliser que rien ne empêche et qu'il est possible de considérer qu'un langage est artistique.

La programmation étant une pratique créatrice qui impose des choix et nécessite souvent une prise de position et un point de vue particulier, il est naturel de la considérer comme une pratique artistique c'est-à-dire de l'art. De même, le code

en tant qu'objet, c'est à dire un ensemble d'information par exemple sous forme de texte peut être une oeuvre puisqu'il est issue d'une création artistique.

On remarque alors que la critique traditionnelement utilisée pour discréditer le code comme oeuvre est en fait caduque. En effet, dire qu'une chose ne peut être de l'art au seul motif qu'elle peut avoir une finalité pratique n'a pas lieu d'être. D'une part il existe du code qui n'existe sans aucune intention d'usage pratique ou de finalité utile réel. Tout code naît d'une idée et donc supose un but mais ce but n'est nullement imposé d'être utile. D'autre part, le rapport entre art et utilité n'est qu'une conception de l'art dont les contre-exemples sont légions et qui n'entre même plus en considération dans les définitions modernes de l'art. En outre, l'art est souvent aux coeurs d'intérêts économiques importants du ménénat au expositions en passant par de nouvelles variations comme les NFT.

Au regard de ces réflexions, il apparait que le code peut être de l'art. La notion de code doit alors s'entrendre dans la complexité de ce qu'elle implique de même que la notion d'art. Un parallèle intéressant consiste à comparer le code et la musique.

La musique se conçoit en effet sur différent support. Elle désigne l'art d'arranger un ensemble de sons entre eux dans le temps. Par extension, elle désigne aussi cet arrangement de son. C'est une pratique définie par un ensemble de règles physiques et conventions. La musique est donc un art de l'information. Ces informations peuvent être accédées directement par un humain en écoutant les sons lorsqui sont produits par un chanteur, un instrument ou tout autre dispositif pouvant émettre des sons. Ces informations peuvent être stockées sur un support physique, par exemple les sillons d'un disque vynil ou les micro-polarités d'un disque dur par exemple. Ces informations peuvent aussi être retranscrites sous forme de texte. Texte qui peut lui-même être stocké sous différentes formes ou être lu directement par un humain disposant des connaissances et capacités adéquates.

Ainsi, la musique peut désigner à la fois l'art et l'oeuvre sous différentes formes de représentations. On pourrait se simplifier la tâche en dissociant les concepts derrières 2 mots: la *musique-pratique* et la *musique-oeuvre*. On peut alors les comparer à la programmation et au code. En effet, de même que la musique-pratique, la programmation dispose nécessairement de règles et contraintes. Puisqu'il s'agit de programmer des machines électriques, la programmation se voit contrainte par les principes de la physique. L'immense majorité des ordinateurs étant basés sur des calculs physiques en binaire, la programmation se doit donc de considérer d'emblée les contraintes physiques des machines en question. C'est ainsi que le programmaur doit souvent faire face à ces contraintes, par exemple en terme de vitesse

d'écriture de disque, de temps de réaction ou de calcul, on encore de l'espace mémoire disponible à différent niveau (cache, RAM, disque ou cloud...). La musique, elle doit faire avec les perceptions humaines et l'acoustique et la physique des ondes sonores.

Afin d'encadre la pratique de la musique, de nombreux cadres et règles ont étés développées. En informatique et donc en programmation, on trouve aussi de nombreux concepts et paradigmes qui offrent un cadre et un ensemble d'outils d'analyse et de construction aux programmeurs. L'objectif final est exclusivement la production d'un code dans le but d'être exécuté. Code qui se compose d'instruction à plus ou moins haut niveau en fonction des niveaux d'abstractions qui séparent le travail du programmeur de celui de la machine physique finale qui exécute les instructions. La programmation est donc aussi art de l'information. C'est aussi un certain art de la traduction. Il faut passer d'une idée humaine, exprimée en langage oral complexe et qui est souvent très imprécise et contextuelle, à un langage très simple, basés sur des règles mathématique et physiques.

De même que le musicien doit s'entraîner pendant des années afin de parvenir à la maîtrise de son intrument afin de transmettre au mieux ses idées et sentiments par la musique, le programmeur doit lui aussi pratiquer pendant des années et surmonter les difficultés intellectuelles et techniques afin de mener à bien ses projets. Et de même en musique qu'un musicien particulièrement talentueux pourra très bien surpasser de nombreux autres musiciens, un programmeur chevronné est tout à fait à même de surpasser une équipe complète. Ceci ne doit pas faire oublier que bien souvent, que ce soit dans la sphère de l'Open Source ou dans le cadre professionnel ou même amical, la programmation se pratique rarement seule mais plutôt en équipe. Et de même que le musicien peut faire de la musique son métier, la programmation peut tout aussi bien relever du gagne-paim que du hobby.

Enfin, la programmation produit un code, une fois que l'idée initiale à réussie à être exprimée d'une manière ou d'une autre en code qui peut être aisément converti en code-machine directement exécutable. Le code intermédiaire, exprimé d'une façon lisible pour un humain est souvent réalisé dans un langage de programmation mais aurait vraisemblablement une forme différente dans un autre langage de programmation. De nombreux langages ayant des propriétés similaires, le choix d'un langage plutôt qu'un autre laisse une certaine marge à la subjectivité et finalement constitue un premier choix important dans la pratique de la programmation. S'ensuit de nombreux autres choix tout au long du processus de développement, encadré par des principes, des goûts personnels, des paradigmes et des pratiques considérés comme plus ou moins bonnes. Le résultat final est donc le produit d'un processus complexe et créatif qui dispose de toutes les caractéris-

tiques d'une oeuvre. Si tous les codes ne sont pas nécessairement artistiques, un beau code, un bel algorithme ou un beau programme est parfaitement capable de susciter des réactions sur un spectateur averti ou non. De par l'aspect complexe et souvent cryptique qu'il semble avoir, fais l'object d'une certaine crainte mêlée de fascination sur le grand publique, ce dont se sont emparées de nombreuses oeuvres à l'exemple des films de science-fiction tels que The Matrix ou TRON.

4 Faire de l'art avec du code ?

Cette partie s'éloigne d'un travail de compilation et de discussion purement journalistique ou philosophique. L'art touchant à la dimension personnelle et profonde de l'individu, la section suivante prend clairement le partie d'être plus subjective parce qu'elle tente de parler d'art au sens personnel. Il est de plus difficile d'étayer les arguments par des sources pertinentes. Cette section tentera de présenter les considérations et les moyens de l'art du code.

4.1 La programmation en tant qu'art

Maintenant qu'on a pu établir pourquoi le code peut être de l'art, on va pouvoir discuter du comment. Il ne s'agit donc pas de s'intéresser littéralement à *faire de l'art avec du code*, mais plutôt de *faire du code avec de l'art.* En effet, on a montré précédemment que le rapport entre programmation et code se rapportait à celui qui existe entre art et oeuvre et c'est donc plutôt dans le sens de la seconde proposition qu'il faut l'entendre.

La programmation à donc pour principal object l'écriture et la conception de code. Le rapport entre programmation et code est lui-même sujet à débat en fonction de l'interlocuteur et de la situation. Selon que celui-ci considère plus ou moins la pratique ou la théorie, il n'aura alors pas la même approche de la pratique. En effet, si on se place plus d'un point de vue théorie, par exemple en considérant une approche formelle de l'algorithmique et de la programmation, alors les règles que l'on va tendre à considérer sont d'ordre logique. On pourra par exemple se placer dans un certain domaine théorique, par exemple le lambda-calcul. Les considérations pratique telles que le choix de tel ou tel langage devient alors secondaire. A l'autre bout du spectre, un programmeur de systèmes embarqué se doit de connaître au mieux les caractéristiques technique du système en question afin de pouvoir le programmer au mieux. Dans ce cas, le choix de l'outil comme

le langage, ou le processus et les outils de tests et de débogages deviennent des considérations de premiers plan et le résultat primera plus le le respect stricte des règles théoriques.

La nature du projet et de l'idée initiale est bien souvent déterminante dans l'approche retenue. En outre, des considérations éloignées peuvent néanmoins amener des solutions et des approches similaires puisqu'il n'existe pas réellement de meilleure façon de faire, et que celles considérées comme potentiellement bonnes peuvenr se recouper. L'ensemble des choix guidant le développement seront responsables du résultat final, du code-source produit. Il n'existe malheureusement aucune garantie de succès et un projet informatique, qu'il soit personnel ou le fruit d'années de labeurs pour des centaines de personnes peut se solder par un échec et un abandon pur et simple de celui-ci. Dès lors, lorsque le code fonctionne bien, il n'est pas difficile d'éprouver des sentiments positifs et artistiques. La programmation est donc l'art de maîtrise la complexité des possibles et des contraintes afin de produire un code qui répond à des objectifs et un point de vue particulier.

A la manière de l'écrivain qui commence inmanquablement sont périble par la feuille blanche et qui peut perdre courage face à l'immensité des possibles, le programmeur moderne commence par un fichier vide. Au départ, il y a souvent une idée, un concept ou un objectif. Dans un cadre plus professionnel, le travail commence généralement bien en amont. Il s'agit d'abord de formaliser les attentes des clients, les besoins, le cas d'utilisations. Puisqu'il s'agit de dialoguer in-fine avec la machine, rien ne doit être laissé au hasard si bien que l'étape de conception initiale peut représenter à elle seule la majeure partie du travail. Le code n'arrivant qu'en fin de cycle une fois que tous les concepts ont été formalisés.

Malgré tout, toutes ces étapes viennent simplifier la programmation, ou tout du moins lui fournir les bases nécessaires pour la faciliter. C'est que le programmeur doit jongler avec de nombreuses règles et concepts: modélisation, langages et outils, paradigmes, élégance, simplicité, efficacité, maintenabilité entre autres. ces critères permettent de comprendre ce qu'est le *clean-code*, c'est-à-dire le beau code. Le code beau aura pour effet de rendre la vie du programmeur et de ses successeurs plus agréable à défaut d'être plus facile. A l'inverse, un code sera dit sale s'il ne respecte pas ces règles de beauté, ce qui aura souvent des conséquences négatives pour l'avenir du projet. Il y a donc souvent un intérêt réel à programmer élégemment. Il pourrait alors s'agir de critères purement à but fonctionnel mais ce n'est pas la cas. En effet, ces règles de beautés reposent sur des parties pris, des réflexions et des goûts qui, s'ils peuvent reposer sur des réflexions réelles, n'en demeurent pas moins sujettent à interprétation et à évaluation. Chacun ayant plus ou moins ses propres règles et considérations de premiers plans.

4.2 Approche esthétique

Le code se définissant le plus souvant comme étant un texte, il y a alors un parallèle naturel à faire entre texte litéraire ou poétique et texte de code informatique [9].

4.3 Conotation et perception

L'essor de l'informatique et notamment la démocratisation des ordinateurs personnels, puis Internet ont dès leurs origines baignés dans la culture des sphères d'initiés à ces nouvelles technologies. Très tôt, ces communautés ont développés leur propres esthétiques et leur influence n'a fait que se renforcer avec l'essor des réseaux sociaux de masses.

5 Conclusion

6 Appendices

6.1 Obscurcissement du code

Références

[1] Alan Turing. "On computable numbers, with an application to the entscheidungsproblem". In: *Proc. Lond. Math. Soc. (3)* s2-42.1 (1937), pp. 230–265.

[2] Claude Nimmo and Larousse (Firm). *Le petit Larousse illustré*. 21, rue de Montparnasse 75283 Paris Cedex 06: Larousse, 2017.

[3] M Romero, B Lille, and A Patiño. *Usages créatifs du numérique pour l'apprentissage au XXIe siècle*. Presses de l'Université du Québec, 2017.

[4] *code*. fr. `https://fr.wiktionary.org/wiki/code`. Consulté le 19-3-2022.

[5] Edsger W. Dijkstra. *A Discipline of Programming*. Prentice-Hall, 1976, pp. vii–30.

[6] Bernard Amade. *Introduction à la programmation*. Paris: MA Editions, 2019.

[7] *L'importance des langages en informatique*. INRIA de Rennes, Nov. 4, 2015.

[8] Gilles. Dowek and Jean-Jacques Levy. *Introduction à la théorie des langages de programmation*. Ellipses, 2006.

[9] Florian Cramer. "Digital code and literary text". In: *Beehive Hypertext/Hypermedia Literary Journal* (Sept. 27, 2001).

[14] René Magritte. *La trahison des images (Ceci n'est pas une pipe)*. Paintings: Oil on canvas 23 3/4 × 31 15/16 × 1 in. Currently on public view: Broad Contemporary Art Museum, floor 3. 1929.

[15] Simon Davies. *Definitions of Art*. Cornell University Press, 1991. ISBN: 978-0-8014-9794-0.

[17] *Les inventions qui ont changé le monde*. 1st Ed. Edition Sélecion du Reader's Digest, 1982. ISBN: 2-7098-0101-9.

[18] *Jacquard - Les Rues de Lyon*. fr. Accessed: 2022-3-21. July 2009.

[19] *The IBM punched card*. en. Accessed: 2022-3-21. Mar. 2012.

[20] View all of Hansel's posts. *The history of coding and computer programming*. en. Accessed: 2022-3-21. Aug. 2018.

[21] B Jack Copeland. "The modern history of computing". In: *The Stanford Encyclopedia of Philosophy*. Ed. by Edward N Zalta. Winter 2020. Metaphysics Research Lab, Stanford University, 2020.

[22] Joasia Krysa. "Ada Lovelace: There Never Was a Note G". In: (). Ed. by Carolyn Christov-Bakargiev. introduction to a notebook 55 Ada Lovelace, author Joasia Krysa, published in dOCUMENTA (13) series 100 Notes – 100 Thoughts. Also published as a chapter in The Book of Books.

[23] Arthur Danto. *La transfiguration du banal : une philosophie de l'art*. Edition du Seuil, 1989. ISBN: 978-2-02-010463-0.

[24] Marcus Steinweg. "Nine Theses on Art". In: *Art U+0026 Research* 3.1 (2009). ISSN: 1752-6388.

[25] *art / Wiktionnaire*. fr. Accessed: 2022-5-12.

[26] Éditions Larousse. *Définitions : oeuvre - Dictionnaire de français Larousse*. fr. Accessed: 2022-5-12.

[27] Editions Larousse. *Définitions : art, arts - Dictionnaire de français Larousse*. fr. Accessed: 2022-5-12.

Bibliographie complémentaire

[10] Andy Oram and Grew Wilson. *L'art du beau code*. 1ère Ed. Paris: Edition O'REILLY, 2008. 621 pp. ISBN: 978-2-84177-423-4.

[11] Martin Fowler. *Refactoring*. 2nd Ed. Malakoff: Dunod, 2019. 419 pp. ISBN: 978-2-10-080116-9.

[12] Robert C. Martin. *Clean Code: a handbook of agile software craftsmanship*. Montreuil: Pearson, 2009. 457 pp. ISBN: 978-2-3260-0227-2.

[13] Dustin Boswell and Trevor Foucher. *The art of readable code*. 1st Ed. O'Reilly Media, Inc, 2011. 204 pp. ISBN: 978-0596802295.

[16] Donald E. Knuth. *The art of computer programming*. 3rd Ed. Vol. 1. Addison-Wesley Longman, 1997. 664 pp. ISBN: 0-201-89683-4.