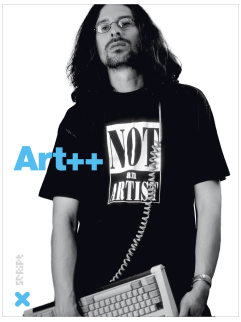


« *The Aesthetics of Generative Code* » (2001) a été traduit dans le cadre de l'ouvrage *ART ++* publié aux Éditions HYX, en juin 2011, sous la direction de David-Olivier Lartigaud. Il est uniquement disponible en ligne. Une biographie des auteurs figure après les notes. Les Éditions HYX tiennent à les remercier pour leur contribution.
http://www.editionshyx.com/site/web/fr/librairie/horscollection/horscollection21.html

---

http://www.generative.net

L'esthétique du code génératif

Geoff Cox, Alex McLean, Adrian Ward[1]

si (

Résumé

L'esthétique, dans son acception générale, met l'accent sur une perception sensorielle subjective associée au vaste domaine de l'art et de la créativité humaine. Cet article, s'appuyant notamment sur l'ouvrage *I See a Voice: A Philosophical History of Language, Deafness and the Senses*[2] de Jonathan Rée, entend démontrer l'utilité d'un réexamen de la relation tumultueuse qui unit l'art à l'esthétique, en vue d'un débat sur la valeur du code génératif. Nous soutenons qu'à l'instar de la poésie, la valeur esthétique du code réside dans son exécution, et pas seulement dans sa forme écrite. Or pour apprécier pleinement le code génératif, il nous faut le « ressentir » afin d'appréhender ce à quoi nous sommes confrontés et afin de parvenir à une certaine compréhension des actions du code.

Toute séparation entre le code et les actions qui en résultent ne ferait que limiter l'expérience esthétique, et en définitive restreindre l'étude de ces formes – en tant qu'objet critique – ainsi que ce qu'il faudrait plutôt appeler dans ce contexte une « poétique » du code génératif.

) {

Esthétique

« La saveur d'une pomme […] réside dans le contact du fruit avec le palais, pas dans le fruit proprement dit. De même, la poésie réside dans la rencontre entre le poème et le lecteur, pas dans les symboles imprimés sur les pages d'un livre. L'essentiel c'est l'acte esthétique […] »[3]

La notion d'esthétique (du grec *aisthesis*), généralement considérée comme s'appliquant aux éléments matériels perceptibles par les sens, est définie de façon plus précise dans *Esthétique* (1750) d'Alexander Gottlieb Baumgarten, qui décrit la beauté comme une « perfection phénoménale »[4] en ce qu'elle est perçue par nos sens, là où l'esthétique « se rapport[e] à la critique du beau ou à la théorie du goût »[5]. Par la suite l'usage général resta centré sur la perception sensorielle subjective, en renvoyant cependant plus particulièrement aux idées d'esthétique et de beauté telles qu'on les associe généralement au vaste domaine de l'art et de la créativité humaine – et ce malgré la tentative de Kant de distinguer la beauté comme étant exclusivement un phénomène du monde sensible et l'esthétique comme une science plus vaste traitant des conditions de la perception sensorielle[6]. Pour les besoins de notre problématique, nous retiendrons cette conception élargie de la notion d'« esthétique », étant entendu qu'elle renferme une idéologie relativement confidentielle et difficilement perceptible sur le plan critique. Bien que cet aspect idéologique sorte quelque peu du champ du présent article, il convient de prendre en compte la description évocatrice qu'en donne Slavoj Žižek – ce qu'il appelle la « matrice générative »[7] –, qui de façon similaire exprime le code génératif sous-jacent à l'action. L'idée, dans la lignée du présent article, est qu'une certaine transparence serait de mise si l'on souhaitait soumettre cette idéologie à la critique. Ce que nous espérons, c'est que le réexamen de la question des limites de l'expérience esthétique permettra de résoudre certains des aspects sur lesquels s'opposent théorie et pratique, ainsi que le problème de la répartition du travail entre l'intellectuel et le physique inhérente à toute production d'art génératif. On a facilement tendance à oublier ces considérations devant une surconcentration sur des productions esthétiques qui se réduisent bien souvent à un jugement et à des critères de goût subjectifs.

Limites

Dans le discours esthétique, l'héritage philosophique prédominant veut que toute théorie sur « la division des arts particuliers […] s'appuie sur les sens auxquels ils s'adressent »[8]. C'est un fait communément admis aujourd'hui, on ne peut tout simplement pas se contenter de la perception sensorielle, à moins qu'elle ne s'inscrive dans le contexte du monde des idées[9]. De la même façon, il est bien trop facile de ramener le monde du multimédia à une expérience multisensorielle (basée sur une combinaison d'images fixes et animées, de son, d'interaction, etc.[10]) se déroulant sans

compréhension préalable du système intégré (le corps-machine) et du code sur lequel il se fonde – qui comprendrait des structures sociales et discursives.

La théorie esthétique avait pour habitude de réduire l'expérience à ce que l'on perçoit au moyen de nos cinq sens, tout en privilégiant la vue et l'ouïe au détriment du toucher et du goût et en plaçant l'odorat « tout en bas de l'échelle » (ce qui n'est pas sans rappeler Dominique Laporte et son *Histoire de la merde*[11]). Par la suite il a été reconnu que cette division de l'expérience sensible était inadéquate, et qu'il était nécessaire d'adopter une approche plus systématique reconnaissant le corps dans son ensemble comme un système intégré. Pourtant, l'héritage de cette approche globale réductrice est bien présent dans le domaine des arts, où les cinq sens se reflètent dans les classifications elles-mêmes. C'est au cours des années 1750, dans *L'Encyclopédie* de Diderot, que les cinq « beaux-arts » – architecture, sculpture, peinture, musique et poésie – furent érigés parallèlement aux sens. La question est : où se situerait le multimédia dans ce schéma ?

On pourrait faire preuve de plus de bon sens, et adopter une approche suggérant que le multimédia a pour rôle de rassembler d'un côté les autres formes d'art et de l'autre les sens. On reconnaît depuis longtemps qu'une forme de mécanisme organisationnel est à l'œuvre dans ce qu'Aristote a appelé le « sens commun », un mécanisme qui se retrouve d'une manière ou d'une autre distribué entre les cinq autres sens – non pas un sixième sens à proprement parler, mais quelque chose qui s'apparenterait peut-être davantage à un système d'exploitation. En philosophie, une approche visant à faire accepter ce dogme consistait à conclure que l'appareil sensoriel convergeait vers le cerveau, et de surcroît que les « idées » mentales englobaient la totalité de l'expérience (c'était l'opinion de Descartes quand il avança dans les années 1630 le fameux « Je pense, donc je suis. »). Cependant cette approche, à l'image de la plupart des pratiques et des théorisations du multimédia, ne permet pas d'apporter des explications satisfaisantes sur les sens, sur les instruments intellectuels ou opératoires. Or il serait tout aussi réducteur d'offrir une synthèse de la perception sensorielle et de la fonction organisationnelle sous l'angle de l'informatique – émanant de ce même héritage marqué par une confiance excessive dans les codes auditifs et visuels. Si c'est la direction que semble prendre cette argumentation, il devient nécessaire d'apporter des précisions sur le contexte.

Dans *I See a Voice*, Jonathan Rée explique que la « philosophie critique » de Kant a permis de résoudre certaines des dissensions établies entre une approche « rationaliste » (par exemple celle de Platon ou de Leibniz) qui défendait généralement l'idée d'un savoir émanant de l'intellect et prévalant donc sur l'expérience sensorielle, et une approche « empiriste » (celle d'Aristote et de Locke) qui prétendait que le savoir était le produit des sens, remettant ainsi en question la fiabilité des vérités universelles (c'est d'ailleurs ce sur quoi se fondent les mathématiques et l'informatique). Kant s'était fixé comme objectif de résoudre ce dilemme, de la manière suivante : « L'entendement ne peut rien intuitionner, ni les sens rien penser. De leur union seule peut sortir la connaissance. »[12] Une telle affirmation ne suggère pas l'existence d'un compromis relativiste, mais sert à souligner que c'est l'entendement qui structure ces processus. Ou pour être encore plus catégorique, on pourrait citer Hegel : « il n'y a rien dans le sens, qui n'ait été dans la pensée »[13]. Si l'on devait s'en servir pour faire une analogie avec les systèmes génératifs, peut-être cela permettrait-il également de mettre en avant les procédures de programmation dissimulées derrière le code brut, lesquelles ne peuvent en elles-mêmes rien percevoir ni penser.

Poésie

Dans la tradition de cette ligne de pensée, Hegel éleva l'« art du son » au domaine du spirituel, concluant que l'« art de la parole » était l'« art total » – « le vrai art absolu de l'esprit »[14]. Malgré des critiques ultérieures à l'égard de ce « phonocentrisme » en tant que source de tout sens et de toute autorité et garant de leur légitimité (de la part de Derrida, notamment), les limites de l'esthétique traditionnelle sont mises en lumière dans la poésie et le problème de sa définition. La poésie, parce qu'elle peut être à la fois lue et entendue, ou encore écrite et récitée/jouée, remet en cause les distinctions à la base de cette classification selon les sens. Un paradoxe que Hegel propose de résoudre au moyen d'un raisonnement dialectique : la parole ne pouvant être entendue par une simple écoute, il devient nécessaire de se la représenter sous forme écrite pour en saisir l'essence. Ainsi, la poésie ne peut être réduite ni à des signes audibles (le temps de l'oreille) ni à des signes visibles (l'espace de l'œil), mais se compose du langage lui-même. Cette synthèse suggère que les formes écrite et parlée œuvrent de concert à la formation d'un langage que nous apprécions en tant que poésie.

Pourtant, peut-on considérer que le code fonctionne de la même façon ? Cette analogie est-elle utile ?

À notre grande déception, cela semble ne pas être le cas avec la « poésie en Perl ». Prenons par exemple *Best of Show*, de Angie Winterbottom, présenté lors du « Perl Poetry Contest », et comparons-le au texte original fourni en parallèle[15] :

```
if ((light eq dark) && (dark eq light)
   && ($blaze_of_night{moon} == black_hole)
   && ($ravens_wing{bright} == $tin{bright})){
 my $love = $you = $sin{darkness} + 1;
};
```

If light were dark and dark were light
The moon a black hole in the blaze of night
A raven's wing as bright as tin
Then you, my love, would be darker than sin.

La seule chose que l'on a prouvée ici, c'est un acte de traduction par lequel on ne fait que « passer » un texte existant de la poésie au langage Perl. Le résultat est une poésie tout ce qu'il y a de plus conventionnelle, certes ingénieuse dans ses quelques changements syntaxiques et grammaticaux, mais qui ne fait pas grand-chose pour articuler le langage Perl en lui-même. Lorsqu'on exécute de la poésie en Perl de cette façon, elle se répète simplement sans reconnaître sa propre exécution. Or c'est cette fonction opérative qui constitue une part essentielle de l'expérience poétique.

Au moment de son exécution (par la lecture et l'écoute), la poésie dispose d'innombrables moyens de produire du sens, et peut être déclamée avec d'infinies variations d'accentuation, de prononciation, de tempo et de style. Bien conscients de ce fait, les surréalistes et les dadaïstes utilisaient des formes arbitraires, du bruit rythmique et des arrangements purement aléatoires de mots et de sons – notamment dans le cadre de poèmes bruitistes et simultanés qui consistaient à lire en même temps des textes dans des langues différentes, et dans d'autres expérimentations automatiques ou génératives. Ils rejetaient ainsi les conventions esthétiques de perfection et d'ordre, d'harmonie et de beauté, et montraient par là même leur refus du goût bourgeois et de toutes les valeurs qui y sont associées. Tristan Tzara, dans le manifeste Dada de 1918, disait : « Je suis contre les systèmes, le plus acceptable des systèmes est celui de n'en avoir par principe aucun »[16]. Comme chacun sait, Tzara

recommandait aux aspirants poètes de découper tous les mots d'un article de journal, de les mélanger dans un sac et de les tirer au hasard pour en faire un poème – un acte révélant les possibilités cachées du langage tout en portant clairement atteinte aux notions de créativité, de génie et d'autorité. Hugo Ball, pour sa part, expliquait : « Dans ces poèmes phonétiques, nous renonçons totalement au langage que le journalisme a corrompu par ses abus »[17]. Ainsi l'universalité de la poésie est-elle remise en question, à l'instar des notions de logique, de raison et d'esthétique. Bien que le texte automatique ait permis de relativiser l'importance du poète en tant que transcripteur ou découvreur du texte plutôt que producteur ou inventeur, il convient de souligner qu'il y a un côté plus résolu dans les arrangements de code auxquels se livre le programmeur.

```
# Extract from walk1/start.pl

my $walk1_beat=0;
my $foo;
sub on_clock {
  return if($foo++ % 4);
  my $beat = $walk1_beat + 1;

  if (($beat-1)%4 eq 0) {
    playnote(7,47+$pitches[$bassctr]-(int($beat/4)*12)) # on-beat
  }
  if (($beat-1)%3 eq 0) {
    playnote(7,35+$pitches[$bassctr]-(int($beat/6)*12)) # syncopate!
  }

  for (0..$#pitches) {
    if (abs($beats[$_]) eq $beat) {
      playnote($_+1,59+$pitches[$_]);
    }
  }

  $bassctr=($bassctr+1)%$#pitches;

  if (rand(50)<25) { $beats[rand(@beats)]++ }
  else { $beats[rand(@beats)]-- }

  if (rand(50)<25) { $pitches[rand(@pitches)]+=$pitches[rand(@pitches)] }
  else { $pitches[rand(@pitches)]-=$pitches[rand(@pitches)] }

  for (0..$#beats)   {   $beats[$_]=wraparound(  $beats[$_],16) }
  for (0..$#pitches) { $pitches[$_]=wraparound($pitches[$_],12) }

  $walk1_beat = ++$walk1_beat % 16;

}
```

Il est primordial, lorsqu'on rencontre le code sous sa forme écrite ou dans le contexte de son exécution, de se concentrer sur les détails plutôt que de se fier à des arrangements aléatoires. En effet, d'importantes portions de code peuvent par exemple correspondre à des « conditions » qui dictent à quel moment devront être exécutées

les parties indentées à venir. Sur le plan de la forme, toute indentation ou autre motif visuel constitue une technique permettant de visualiser le flux logique – bien qu'un code identique puisse être exprimé sous n'importe quelle forme ou arrangement et produire toutefois le même résultat. Certaines conditions sont évaluées dans le cadre d'autres conditions, contribuant ainsi à produire des réponses infiniment complexes – en programmation, le recours à des techniques d'indentation permet de visualiser la logique booléenne qui forme le noyau dur du code. L'utilisation du langage est hautement contrôlée et joue sur des nuances subtiles.

Voyons l'exemple suivant :

```
$walk1_beat = ++$walk1_beat % 16;
```

On pourrait éventuellement ajouter une parenthèse pour clarifier – ou compliquer – un peu le tout.

```
$walk1_beat++;
if ($walk1_beat eq 16) { $walk1_beat=0 }
```

Lors de l'exécution le résultat est presque identique à celui de la première ligne de code, mais l'opération qui permet d'y arriver n'est pas la même. De plus, une connaissance spécialisée du langage Perl est nécessaire pour comprendre que « `eq` » est un opérateur qui compare des chaînes de caractères, et non un opérateur numérique. L'équivalence entre « `eq` » et « `==` » est un subtil jeu de langage.

Les médias génératifs reposent sur le principe fondamental d'une modification des données à mesure que le code s'exécute. Dans l'exemple plus haut, les symboles « `++` » et « `--` » servent à incrémenter et décrémenter des nombres – ce qui, associé à l'opérateur mathématique « % » appelé modulo, montre bien la variation constante des nombres. Bien que ceux-ci puissent être calculés à la main et tracés sur ce qui ressemblerait à une partition musicale, grâce au pouvoir du code cette opération s'effectue en « temps réel », et ses effets sont pratiquement inconnus jusqu'à l'exécution. Le code pourrait être exécuté à l'infini, il n'en continuerait pas moins à produire de nouveaux arrangements.

Le fonctionnement du code est manifestement identique à celui de la poésie dans la mesure où il joue avec les structures du langage lui-même, et avec les différentes perceptions que nous en avons. En ce sens, toute poésie doit être considérée comme générative en ce qu'elle est en perpétuel devenir. Même pour le surréaliste Paul Valéry, un poème « entraîne une liaison continuée entre la voix qui est et la voix qui vient et qui doit venir »[18]. Il est génératif dans le sens où il se déploie en temps réel.

```
# Extract from nuane/start.pl

sub on_clock {
  return if ($foo++ % 4);
  return if (++$beats < $aTime);
  $beats = 0;
  $client->ctrl_send('note', "$aNote, 1, 0") if $aNote;
  $aNote=47+$notes[$ptr];
  $aTime=$times[$ptr];
  $ptr=($ptr+1)%8;
  $client->ctrl_send('note', "$aNote, 1, " . (80 + rand(40)));
}
```

Les commandes peuvent être exécutées de diverses façons. Les deux premières lignes de la fonction « `on_clock` » sont des déclarations « `return` » qui empêchent le reste du code de s'exécuter si la condition fournie devient vraie.

```
return if (++$beats < $aTime);
```

est d'un point de vue fonctionnel similaire à

```
if (!(++$beat < $aTime)) {
 # …
}
```

Dans cet exemple, l'ordre de mots qui a été choisi est « alternatif ». Ici le parallèle avec la poésie est évident, l'ordre des mots pouvant en effet aider à exprimer ce qui est le plus important dans une instruction donnée : la condition ou l'action.

Par analogie, le code génératif a des qualités poétiques, puisqu'il n'agit pas dans un contexte spatio-temporel bien défini mais plutôt comme une série d'« actions » consécutives pouvant être répétées, et dont le résultat doit être imaginé dans différents contextes. Le code est la notation d'une structure interne que l'ordinateur exécute, exprimant des idées, une logique et des décisions qui agissent comme un

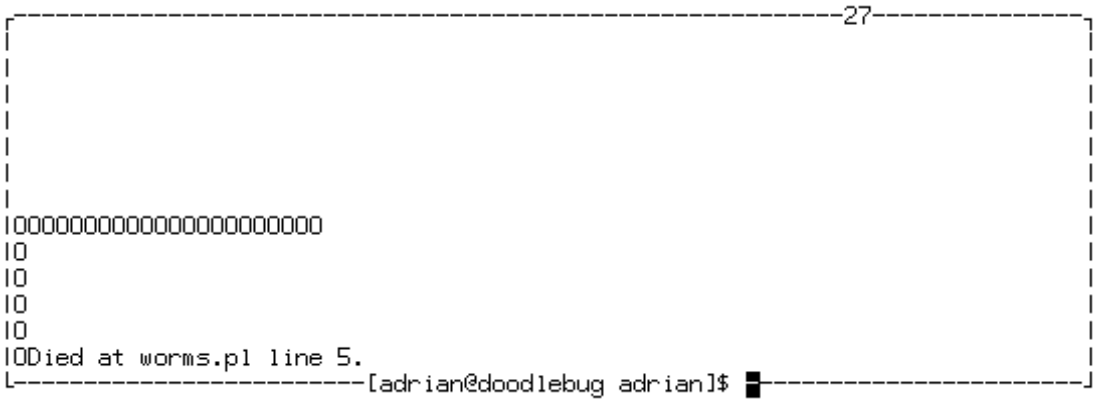
prolongement des intentions de l'auteur. La forme écrite n'est que la notation d'une logique lisible par un ordinateur, et constitue la représentation de ce processus. Pourtant le code écrit ne correspond pas exactement à ce que la machine exécute, puisqu'il existe de nombreux niveaux d'interprétation, de compilation et de liaison. Le code n'est vraiment compréhensible que dans le contexte où l'on considère sa structure d'ensemble – c'est précisément ce qui le rend comparable à un langage (qu'il se présente sous forme de code source, de code machine, ou même de *bytes* bruts). Il peut être difficile de comprendre le code lorsqu'on ne l'a pas écrit, mais après tout, l'ordinateur est polyglotte. Ainsi, comprendre le code écrit par un autre programmeur revient presque à écouter de la poésie dans une langue étrangère – l'apprécier demande bien plus qu'une simple compréhension de la syntaxe ou de la forme du langage utilisé, et à ce titre la traduction est notoirement problématique. La forme et la fonction ne devraient pas être abusivement séparées.

Poétique

Il est clair que le code en lui-même n'est pas de la poésie, mais il en adopte en partie le rythme et la forme métrique. Il est finement travaillé, et les façons de l'exprimer sont tout aussi innombrables que singulières. À l'instar de la poésie, la valeur esthétique du code réside dans son exécution, et pas seulement dans sa forme écrite. Pour apprécier pleinement ce fait, il nous faut « percevoir » le code afin d'appréhender ce à quoi nous sommes confrontés et afin de parvenir à une certaine compréhension de ses actions[19].

```
#!/usr/bin/perl
use Curses;keypad initscr;nodelay 1;box qw{| -};($l,$d,$k,@f)=(1..3,[10,10]);&
n;while(){refresh;@f=([$f[0][0]+$d%2-($d==1)*2,$f[0][1]+$d%2-1+($d==2)*2],@f);
select$f,$f,$f,.06;($c=getch)+1and$d=4-($c%2?2:0)-($c<260);addch@{pop@f},' 'if
@f>$l;$l+=$_=inch@{$f[0]};if(!/ /){/\d/||die;addstr 0, 60,$l;&n}addch@{$f[0]},
'O'}sub n{while(){@v=(rand 24,rand 80);inch(@v)eq' '&&last}addch@v,''.rand 10}
```

La portion de code ci-dessus est extrêmement dense et difficile à interpréter. Certains mots-clés émergent, mais surtout le code est soigneusement justifié en cinq lignes de même longueur. Les structures conditionnelles sont toujours présentes ici (notez l'utilisation de « { » et « } »), mais elles sont arrangées et condensées en vue d'un plus grand impact visuel. Pour apprécier pleinement le code, il faut soit le déconstruire et l'utiliser, soit y jouer dans la mesure où c'est en réalité un jeu[20].



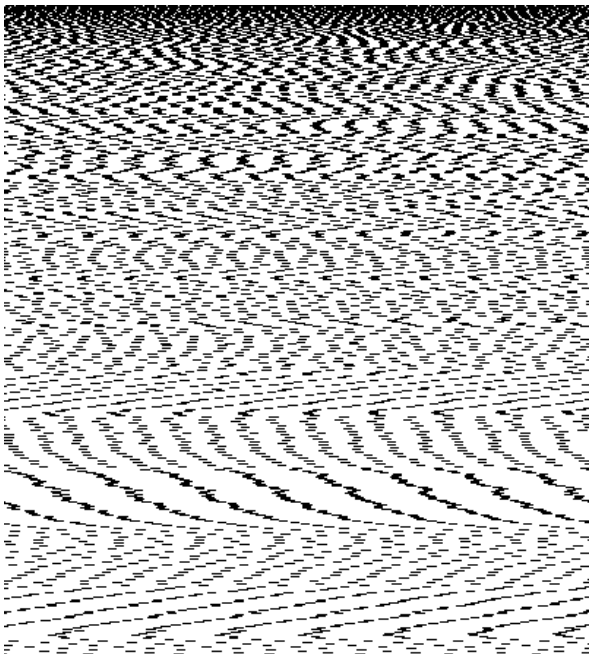
L'obfuscation est un effet secondaire de sa densité, l'intention principale ayant été de réduire le code au plus petit nombre de caractères possible. En termes généraux, il s'agit là d'une tentative de combiner la forme et la fonction.

Assurément, cela ne signifie pas qu'il faille privilégier le code (comme l'insinue Adorno lorsqu'il parle de la musique en tant que sous-produit de la partition), mais plutôt que le code et son exécution doivent faire l'objet d'une expérimentation parallèle – ce qui est à la fois nécessaire et impossible pour des systèmes génératifs ou autonomes. Toute notion d'une autonomie du code dépend de sa place au sein de sa structure opérationnelle. De cette façon le code reflète l'activité de l'homme, qui est elle-même codée dans des cadres sociaux et discursifs – ainsi la paternité se définit-elle en termes de responsabilité (sociale) vis-à-vis du système d'exploitation et des structures de langage[21]. Il ne fait aucun doute que les médias génératifs opèrent eux aussi de cette manière et semblent « encapsuler » le paradoxe de l'autonomie. L'art génératif doit prendre conscience des conditions de sa propre réalisation – de sa *poesis* (du grec *poiesis*, « art poétique » ou « créativité », lui-même issu de *poiein*, qui signifie « faire »).

Dans l'esprit du processus ouvert et de l'*open source*, une certaine transparence s'impose sur ce point.

```
#!/usr/bin/

$power = 8;
sub fission {
  fork or $child = 1;
  --$power if $child;
  if ($child) {
    exit unless --$power
}
  return $child;
}
while (not &fission) {
  print 0;
  bomb:
  while (&fission) {
    print 1
  }
}
goto 'bomb';
```



Dans cet exemple[22], le programme se sépare en deux à chaque itération. Le code paraît relativement long, dans la mesure où l'instruction de base pourrait être réduite à une courte ligne :

```
fork while 1;
```

L'instruction ordonne simplement de « séparer ce processus en deux parties à l'infini » – après la première itération on obtient donc deux processus, après la seconde on en obtient quatre, puis huit, et ainsi de suite pour une durée indéterminée. Pourtant, le résultat du premier exemple est significatif en ce qu'il constitue une visualisation performative et plus complexe de l'exécution du processus. Sur le plan technique, l'ordinateur est soumis à une telle charge qu'il ne peut satisfaire à ces instructions – au bout d'un certain temps les appels « fork » ne parviennent plus à séparer le processus en deux, et la façon dont l'ordonnanceur gère les différentes requêtes est de moins en moins ordonnée à mesure que la charge augmente. Le résultat est donc une visualisation des performances de l'ordinateur au cours de l'exécution du programme. Le rendu visuel serait très différent d'un ordinateur à un autre, et ferait ainsi office de « filigrane » du processeur et du système d'exploitation. Le code et les actions qui en résultent sont intimement liés en un dialogue poétique.

Toute séparation entre le code et les actions qui en résultent ne ferait que limiter l'expérience esthétique, et en définitive restreindre l'étude de ces formes – en tant qu'objet critique – ainsi que ce que l'on pourrait nommer dans ce contexte une « poétique ». Le code génératif résume bien ces questions :

« Le résultat serait […] en quelque sorte [comme] une poésie correctement définie ; un langage si bien choisi et si judicieusement arrangé qu'il resterait vivant, entraînant et "plaisant à l'oreille", et ce même lorsqu'il aborderait des sujets ennuyeux ou déplaisants. »[23]

Ce que nous proposons, c'est une production de code génératif mise en œuvre avec autant de réflexion critique et de panache.

}

1. Tous les scripts Perl présents dans ce texte sont d'Alex McLean et Adrian Ward du groupe Slub, http://www/slub.org

2. Jonathan Rée, *I See a Voice: A Philosophical History of Language, Deafness and the Senses*, Flamingo, Londres, 1999.

3. Jorge Luis Borges, Avant-propos à *Obra Poética* (cet extrait n'apparaît pas dans sa traduction française publiée sous le titre *Œuvre poétique, 1925-1965*), cité dans Juhani Pallasmaa, *The Eyes of the Skin: Architecture and the Senses*, Academy Editions, Londres, 1996, p. 6.

4. Cette célèbre formulation d'Alexander Gottlieb Baumgarten, à l'origine tirée du paragraphe 662 de *Metaphysica* et reprise plus tard dans *Aesthetica*, n'apparaît pas dans la version française partielle *La Métaphysique*, dans *Esthétique*, trad. du latin par Jean-Yves Pranchère, L'Herne, Paris, 1988.

5. Terry F. Hoad, *The Concise Oxford Dictionary of English Etymology*, article « aesthetic », Oxford University Press, Oxford, 1986, p. 7.

6. Raymond Williams, *Keywords: A Vocabulary of Culture and Society*, Fontana, Londres, 1988, p. 31.

7. Slavoj Žižek (Sous la dir. de), *Mapping Ideology*, Verso, Londres, 1997.

8. Georg Wilhelm Friedrich Hegel*, Esthétique Volume 2*, trad. de l'allemand par Charles Bénard (revue et complétée par Benoît Timmermans et Paolo Zaccaria), Livre de Poche, coll. « Classiques de Poche », Paris, 1997, p. 16.

9. Pour en savoir plus sur les limites de l'esthétique, voir Andrew Benjamin, Peter Osborne (Sous la dir. de), *Thinking Art: Beyond Traditional Aesthetics*, ICA, Londres, 1991.

10. Sean Cubitt, *Digital Aesthetics*, Sage, Londres, 1998, compte parmi les tentatives de considérer les systèmes numériques au-delà de simples questions de design. Un ouvrage dont le titre est tout à fait approprié.

11. Dominique Laporte, *Histoire de la merde : prologue*, Christian Bourgois, coll. « Choix-essais », Paris, 1993.

12. Immanuel Kant, *Critique de la raison pure*, trad. de l'allemand par André Tremesaygues et Bernard Pacaud, Presses Universitaires de France, coll. « Quadrige », Paris, 1944, p. 77, cité dans Jonathan Rée, *op. cit.*, 1999, p. 330.

13. « Nihil est in sensu, quod non fuerit in intellectu », Georg Wilhelm Friedrich Hegel, *Encyclopédie des sciences philosophiques*, tome 1 : *La Science de la logique*, trad. de l'allemand par Bernard Bourgeois, J. Vrin, Paris, 2000, p. 173 (cité dans Jonathan Rée, *op. cit.*, 1999, p. 342). Cela ne signifie pas que les sens ne jouent pas un rôle déterminant, dans la mesure où ils structurent nos interprétations dans l'espace et le temps (ce que Kant a dissocié comme étant l'expérience « extérieure » et « intérieure » – respectivement spatiale et temporelle). Rée pour sa part en retrace l'histoire en soulignant l'importance de la phénoménologie de Husserl. Pour en savoir plus, voir dans Jonathan Rée, *op. cit*., 1999, le chapitre intitulé « The Five Senses and the History of Philosophy », p. 329-345.

14. Georg Wilhelm Friedrich Hegel, *Esthétique de la peinture figurative*, trad. de l'allemand par Samuel Jankélévitch, Hermann, Paris, 1964, p. 178, cité dans Jonathan Rée, *op. cit.*, 1999, p. 356.

15. Kevin Meltzer, « The Perl Poetry Contest », dans *The Perl Journal*, Vol. 4, n° 4, 2000, http://www.foo.be/docs/tpj/issues/vol5_1/tpj0501-0012.html. Meltzer explique :
« Cette courte contribution, que l'on doit à Angie Winterbottom, fut la plus intéressante. Le style est frais et unique, et la façon dont elle utilise les représentations visuelles dans le texte est ingénieuse. Voyez l'extrait suivant :
($blaze_of_night{moon} == black_hole)
"The moon, a black hole in the blaze of night."
Merveilleux ! Angie nous dit que cette contribution est tirée de la chanson de Jim Steinman intitulée "The Invocation", sur l'album *Original Sin* de Pandora's Box. »

16. Tristan Tzara, *Manifeste Dada 1918*, dans *Sept Manifestes Dada*, Pauvert, Paris, 1967, p. 26, cité dans Charles Harrison, Paul Wood, *Art en théorie, 1900-1990 : une anthologie*, trad. de l'anglais par Annick Baudoin, Hazan, Paris, 1997, p. 287.

17. Hugo Ball, cité dans Henri Béhar, Catherine Dufour, *Dada circuit total*, L'Âge d'homme, Lausanne, 2005, p. 148.

18. Paul Valéry, *Introduction à la poétique*, Gallimard, Paris, 1938, p. 41, cité dans Jonathan Rée, *op. cit.*, 1999, p. 361.

19. Le développement de certains logiciels montre bien qu'il est possible d'adopter ce principe. C'est le cas notamment de ceux correspondant à la norme MPEG-4 Structured Audio, « qui distingue le son non pas comme des données échantillonnées, mais comme un programme informatique qui génère un signal audio lorsqu'il est exécuté. Les informaticiens appellent cette approche le codage de Kolmogorov.
MP4-SA combine un langage puissant pour le traitement audio (SAOL, prononcé « sail ») et un langage d'édition de partitions de musique (SASL, prononcé « sassil ») avec un support hérité pour le format MIDI. MP4-SA définit également un encodage efficace de ces éléments en un format de fichier binaire […] adapté à la transmission et au stockage. »
John Lazzaro, John Wawrzynek, « MPEG-4 Structured Audio: Developer Tools », http://www.cs.berkeley.edu/~lazzaro/sa/

20. Il s'agit en l'occurrence d'une version complète d'un jeu auquel on devient très vite accro, appelé *Worm*. Les utilisateurs de téléphones Nokia redécouvrent aujourd'hui ce classique des jeux d'arcade sous le nom de *Snake*.

21. Pour davantage d'informations sur la question, voir notre article précédent intitulé « The Authorship of Generative Art », GA 1999, http://www.generative.net/

22. Graphisme et source tirés de « invalidObject Series (while) » de Pimmon, http://www.fallt.com/invalidObject/while/index.html
23. Jonathan Rée, *op. cit.*, 1999, p. 349, paraphrasant Alexander Gottlieb Baumgarten, *Méditations philosophiques sur quelques sujets se rapportant à l'essence du poème*, dans Alexander Gottlieb Baumgarten, *op. cit.*, 1988.

Les auteurs

Geoff Cox

Artiste, auteur et commissaire pour divers évènements, Geoff Cox enseigne actuellement à l'Université de Plymouth (Royaume-Uni) où il est responsable du Master arts numériques et technologie. Membre de la faculté de recherche de l'institut Transart, ses travaux portent notamment sur le Software Art, auquel il a consacré sa thèse intitulée *Antithesis: The Dialectics of Software Art* (2006). Co-éditeur de *Economising Culture* (2004), *Engineering Culture* (2005) et *Creating Insecurity* (2009) dans le cadre des publications de DATA browser, il est aussi membre du conseil de la Kahve-Society et de la galerie Spacex, ainsi que trésorier du UK Museum of Ordure. Il travaille actuellement sur plusieurs projets d'expositions avec le centre d'art contemporain Arnolfini de Bristol.

Alex McLean

Alex McLean est un musicien-programmeur vivant et travaillant à Londres. Préparant actuellement une thèse en arts et technologie informatique au sein du Goldsmiths College de l'Université de Londres, ses recherches portent sur la synthèse de vocables, avec pour objectif de faire de la musique à partir de texte. Ses compositions, résultant de l'utilisation combinée des langages de programmation Perl et Haskell, sont présentées lors de performances live au cours desquelles la musique évolue au gré de l'écriture et de la manipulation du code. Membre actif de la branche londonienne des Dorkbots, il est également l'un des fondateurs de Runme.org. Il forme avec Adrian Ward et Dave Griffiths le groupe Slub, qui depuis 2000 développe des logiciels composant des « musiques à boire de la bière ».

Adrian Ward

Adrian Ward est un programmeur, musicien et artiste dont les oeuvres ont été exposées internationalement. Créateur de la société Signwave, il met à profit son savoir-faire en matière d'applications génératives dans le monde des affaires et les sphères culturelles. Il a également intégré les rangs de Clay Interactive Ltd, pour laquelle il conçoit des expositions interactives. Il est l'auteur des logiciels parodiques *Autoshop* (1999) et *Auto-Illustrator* (2001), ce dernier ayant été lauréat de Transmediale 2001 et récompensé de la mention honorable du Prix Ars Electronica 2001. Membre du conseil d'administration du UK Museum of Ordure, il forme avec Alex McLean et Dave Griffiths le groupe Slub, qui depuis 2000 développe des logiciels composant des « musiques à boire de la bière ».