grande popularité du logiciel dans les années 1990 et 2000. Director permet de créer une grande palette de projets interactifs, en deux et trois dimensions et sur de nombreux supports (CD-ROM, DVD-ROM, Internet) ainsi que pour des installations. Une partie de la notoriété de Director provient de la facilité avec laquelle ses programmes peuvent être mise en ligne : ceux-ci sont lisibles dans un navigateur grâce au module externe (ou plug-in ; voir glossaire p. 601) Shockwave (pour les systèmes d'exploitation Mac et Windows). Le langage de script de Director, le Lingo, créé par John Henry Thompson, a une syntaxe simple et facile à lire pour les anglophones, même s'il a évolué depuis ses origines pour se rapprocher d'autres langages de programmation. Depuis Adobe Director MX 2004, Director accepte, en plus du Lingo, le langage ECMAScript, version standardisée de JavaScript, un autre langage orienté objet.

En ce qui concerne la pérennisation des œuvres conçues grâce à Adobe Director, l'approche développée par ce logiciel, depuis sa création en 1994, n'a pas d'équivalent en termes de logiciels propriétaires ou libres (voir glossaire p. 601). À moins que Director ne connaisse un changement de mode de développement et de distribution, comme cela a été le cas pour Blender (un logiciel propriétaire, en partie payant, racheté par son concepteur et ses utilisateurs et transformé en logiciel libre), il faut trouver une alternative pour reprogrammer une œuvre comme *still living*, c'est-à-dire un logiciel qui permette d'implémenter les algorithmes créés par l'artiste et d'animer les graphiques.

La migration (voir glossaire p. 601) ne peut être une stratégie de conservation viable, ou seulement à très court terme, si le développement de Adobe Director ne s'arrête pas et si certains éléments visuels doivent être mis à jour (par exemple dans le cas d'une augmentation significative de la résolution d'écran sur les ordinateurs contemporains, ou dans celui d'une impossibilité d'exécution correcte des programmes encapsulés). Pour l'artiste, il est essentiel que l'œuvre soit exécutée en temps réel, que l'on ressente que le mouvement est généré au moment même où on le voit. L'affichage de l'heure et de la date sous le titre de l'œuvre (seul paramètre pour lequel l'œuvre requiert une information externe au programme, en récupérant l'heure de l'ordinateur) participe de cette sensation d'ici et maintenant.

L'émulation (voir glossaire p. 600) permet de préserver l'existant, c'est-à-dire le code source de l'œuvre dans son environnement. Un émulateur permet de faire tourner le fichier exécutable dans de nouvelles plateformes. La popularité des plateformes sur lesquelles fonctionnent *still living*, Mac OS X, Mac 10.4 et 10.5 et Windows 98, XP et Vista, implique que de nombreux émulateurs de qualité sont et seront disponibles à court et moyen terme, encore faut-il qu'ils permettent de conserver les éléments que l'artiste juge importants. Lors d'un entretien avec Antoine Schmitt [6], la question de la ligne de démarcation entre les altérations acceptables et les dommages inacceptables a été discutée.

6 Cf. l'extrait de l'interview avec Antoine Schmitt, p. 484-487.

Ainsi, il n'est pas acceptable pour l'artiste que l'œuvre soit montrée sur un écran en noir et blanc ou présentant des couleurs dégradées, ni sur un ordinateur trop lent. Il n'est pas non plus acceptable que la composition des graphiques sur l'écran soit abîmée par une résolution trop faible ou par un format ne respectant plus la proportion entre les différents types de graphiques, tels que l'artiste les a dessinés. Antoine Schmitt est favorable à l'émulation, à condition que la qualité du mouvement ne soit pas altérée par l'interprétation du code par l'émulateur. Il n'a pas de préférences entre l'émulation et la reprogrammation, pourvu que l'œuvre soit respectée et montrée sans altération dommageable. Il souhaite par ailleurs être tenu informé, voire consulté à ce sujet et à propos des interventions de conservation sur son œuvre.

Le portage (voir glossaire p. 601) dans un environnement différent d'Adobe Director apparaît donc comme l'une des principales stratégies de pérennisation de cette œuvre. Une dimension technique et une dimension stratégique sont impliquées dans ce choix. Techniquement, il s'agit de trouver un logiciel permettant de créer des programmes autonomes et de gérer aussi bien le dessin que le mouvement des graphiques. Stratégiquement, un logiciel libre semble plus approprié, si possible, en cas de reprogrammation, dans la mesure où la publication et le partage des données, la collaboration directe entre développeurs et utilisateurs dans le développement du logiciel rendent possibles l'appropriation et la maintenance d'un environnement de programmation. Cela permet d'éviter le risque d'arrêt de développement en cas de changement de politique commerciale, à la condition toutefois que cet environnement libre soit soutenu par une grande communauté de programmeurs, autant développeurs qu'utilisateurs. Un logiciel libre peu utilisé n'est pas plus assuré de pérennité qu'un logiciel propriétaire. Antoine Schmitt utilise d'ailleurs maintenant Processing dans son travail, après avoir utilisé pendant des années Adobe Director, car il estime que ce logiciel libre - conçu pour un usage artistique par Casey Reas et Ben Fry [7] - peut être utilisé de manière polyvalente. L'algorithme de mouvement conçu par l'artiste est au cœur de l'œuvre et lui donne sa qualité propre. Les algorithmes qui permettent le mouvement et les graphiques animés peuvent être dissociés lors d'un portage de l'œuvre dans un autre environnement. Plus que le code source (qui inclut les éléments graphiques), il s'agit de conserver l'algorithme du mouvement. L'algorithme central pourrait, si nécessaire, être transposé dans un autre langage, à condition de garder la même qualité et dynamique de mouvement qu'à l'origine. Quant à l'aspect visuel très travaillé, ce formalisme capitaliste, le code source de cette deuxième catégorie peut aussi être remplacé, à condition que le rendu visuel soit le plus proche possible des spécifications de l'œuvre définies par l'artiste. Celui-ci a défini un certain nombre de paramètres : composition (disposition dans l'espace des différents éléments graphiques), couleurs

7 Cf. : http://processing.org/ (consulté le 15/7/2013).

franches, typographie, cartel (titre de l'œuvre, heure de l'ordinateur), netteté du pixel, grain de l'œuvre, vitesse du déroulement qu'il a fixée dans le code, etc.

La résolution de l'écran et le format de celui-ci peuvent être des facteurs importants de reprogrammation. En ce qui concerne la résolution, si celle d'un ordinateur à venir affiche la composition de manière plus petite sur l'écran, il faut alors simuler une résolution d'écran plus basse, tout en gardant le grain de l'œuvre. De même, le format d'un écran affecte la composition visuelle de l'artiste, les proportions entre les différents éléments. Ainsi, entre le moment de la création de l'œuvre et le moment de son acquisition, le format 16:9 est devenu populaire, s'ajoutant au format 4:3. L'artiste a pu facilement remédier à ce changement en modifiant son code afin que le format soit repéré et que la composition appropriée à ce format soit lancée.

still living comporte des éléments contextuels : les codes graphiques utilisés (camemberts, courbes...) pour représenter le mouvement. À l'heure actuelle, ces éléments sont transparents, relativement universels, liés à la culture visuelle d'entreprise, et semblent ne pas être en voie d'obsolescence rapide. Cependant, ils vont être amenés à évoluer à court ou moyen terme. Lors des discussions avec Antoine Schmitt, la possibilité de les transformer pour suivre l'esthétique de l'époque à laquelle l'œuvre serait montrée a été évoquée. Pour l'artiste, le but de cette œuvre n'est pas d'évoluer au fur et à mesure ; ce faisant, elle porte intrinsèquement les éléments propres à l'époque de sa création. Cependant, il ne s'oppose pas à cette transformation, en particulier dans le cas d'une reprogrammation complète.

La mise à disposition de l'œuvre pour le public est également associée à cette forme de communication institutionnelle qui va évoluer tout autant que les graphiques. Les préconisations de l'artiste en matière de monstration sont plus ouvertes à l'interprétation que ne l'est l'œuvre elle-même, et donc plus susceptibles d'évoluer avec le temps.

Les trois étudiants-ingénieurs qui ont étudié la pérennisation de *still living H, I, J* avant le début du projet *digital art conservation*, ont proposé d'utiliser le langage Java pour reprogrammer l'œuvre. Ils n'ont pu achever la reprogrammation dans le temps qui leur était imparti pour cette recherche et ne sont pas parvenus à un résultat visuel qui soit compatible avec l'œuvre. Cependant, cela n'exclut pas la pertinence de leur proposition. Pour justifier ce choix, ils ont mis l'accent sur plusieurs avantages de Java. Java est un langage universel, dans la mesure où ses programmes sont exécutables dans toutes les plateformes qui disposent d'une machine virtuelle Java. Il est portable sur plusieurs plateformes, grâce au Java Runtime Environment (JRE) qui interprète le code écrit en Java et le convertit en mode natif, puis l'exécute. Cela rend le code indépendant de la plateforme. De plus, il est possible d'accéder à certains éléments de la machine cible grâce à des bibliothèques standards. L'orientation objet de Java le rend compatible avec les besoins d'une œuvre comme *still living*. C'est un langage populaire, proche de C++ (voir glossaire p. 600) mais

beaucoup plus simplifié que ce dernier qui permet de faire de la programmation modulaire, rendant possible la réutilisation des objets séparément. De plus, c'est un langage largement documenté. Le code écrit avec Java peut être intégrable dans des pages Web, grâce à des *applets*, ces applications affichables et interprétables par les navigateurs. Java présente également l'avantage d'une compatibilité ascendante (ou rétrocompatibilité), c'est-à-dire qu'un programme exécuté dans un JRE ancien pourra être exécuté dans un JRE plus récent, ou qu'un programme conçu dans un environnement de développement Java Development Kit (JDK) plus ancien pourra être modifié dans un JDK récent. Le code source de Java est publié sous la licence libre GNU General Public License (GPL) sous le nom d'OpenJDK. C'est devenu un standard. L'ouverture de la plateforme Java a permis un perfectionnement et un développement de bibliothèques par une communauté de développeurs et d'utilisateurs. Java est doté de bibliothèques graphiques de base performantes et très complètes, ainsi que d'autres dédiées au dessin permettant la génération de graphiques de type courbe, barre, etc. pour représenter des données chiffrées dans une application ou dans une image, comme la bibliothèque JFreeChart [8] que les étudiants ont utilisée. Java permet aussi de gérer de nombreux autres éléments multimédia (2D, 3D, animation, vidéo, interactivité, etc.).
Les étudiants ont cependant mis en lumière les inconvénients de Java pour cette œuvre. Java consomme beaucoup de ressources informatiques, ce qui implique une lenteur dans l'exécution du code. L'interprétation du code par la machine virtuelle rend l'exécution moins rapide que si le code est compilé directement en langage machine (voir glossaire p. 600-601). Cela peut être le cas avec certains compilateurs (voir glossaire p. 599), ce qui permet un temps d'exécution plus rapide mais porte atteinte à la portabilité du code, le code étant alors exécuté dans une seule architecture.

Le choix entre les différentes stratégies se fait selon les possibilités matérielles et logicielles, au moment où l'œuvre doit faire l'objet d'une intervention spécifique. La situation sociotechnique, les moyens de l'institution, amènent à un choix qui permet au public de faire l'expérience d'une œuvre non altérée, où la qualité de mouvement et la précision des graphiques sont respectées.

Conclusion

Les installations *still living H, I, J* sont des natures mortes mais vivantes (d'où leurs noms) grâce au mouvement qui les sous-tend et aux forces internes programmées qui animent les graphiques. Pour Antoine Schmitt, *still living* tente de retirer le pouvoir de l'image, de passer à travers elle vers ce qui en est la cause, le mouvement. Cependant, cette qualité de mouvement implique que le code source de l'œuvre soit exécuté en temps réel, dans une durée infinie, ce

8 JfreeChart est une interface de programmation (API) Java qui permet de créer des graphiques et diagrammes. Cf. : www.jfree.org/jfreechart/ (consulté le 15/7/2013).

qui guide le choix de la stratégie de conservation. Un enregistrement (qu'il s'agisse d'une captation de l'image de l'œuvre ou des données des mouvements qui pourraient être rejouées) ne peut nullement être envisagé pour présenter l'œuvre. Une captation n'aurait qu'une valeur documentaire et ne pourrait permettre le ressenti face au mouvement que l'artiste souhaite créer chez le spectateur. Cette volonté de garder l'algorithme au centre de l'œuvre se retrouve chez l'artiste dans sa stratégie personnelle de pérennisation de ses œuvres, en particulier dans le cas d'une acquisition. L'ajout du code source, à fin de maintenance, aux fichiers exécutables et aux préconisations d'installation et de monstration, permet aux institutions et aux collectionneurs d'appréhender plus sereinement la longévité d'une œuvre, même si cela ne constitue pas une solution globale pour toute création numérique. Le code source permet d'enrichir la connaissance de l'œuvre et, associé aux préconisations générales de l'artiste, d'envisager un portage adapté aux technologies de l'époque à laquelle l'œuvre sera montrée.

still living
Entretien avec Antoine Schmitt

Anne Laforet

Extrait de l'interview du 15 juin 2011,
Espace multimédia gantner, Bourogne (Territoire de Belfort).

Vous avez parlé du logiciel Adobe Director (MX 2004) pour le développement de *still living* [2006]. C'est un logiciel dont il semble qu'il y aura peu de versions supérieures, contrairement à ce qui a pu se passer avant. Quel est votre sentiment concernant ce logiciel particulier par rapport à la question de la conservation et de la pérennisation de votre œuvre ?

Antoine Schmitt : C'est une question plus large. Ce qui m'obligerait à porter *still living* sur un autre logiciel serait qu'elle ne fonctionne plus sur les ordinateurs actuels. Ça c'est de l'ordre de l'obligation. Par contre, pour ce qui est d'une stratégie de conservation, je réfléchis à l'idée de porter *still living* sur un autre environnement, en me disant que ça faciliterait sa conservation dans le futur. Je pourrais le faire moi-même ou bien, d'ailleurs, le faire faire, soit pour la conservation elle-même, soit pour rassurer les acheteurs potentiels sur le fait que l'œuvre est conservable et maintenable plus facilement. L'équation clé dans l'histoire étant que – en supposant que l'œuvre ne fonctionne plus un jour telle quelle – la manière la plus simple de la reprogrammer – pas la plus simple, peut-être, mais la seule possible – serait de la reprogrammer dans un environnement qui existe à ce moment-là. Or un environnement comme Director a de grandes chances de cesser de fonctionner. Il serait donc impossible, dans le futur, de reprogrammer dans cet environnement-là ; un autre serait donc requis. Dans ce cas, autant se baser sur un environnement open-source (voir glossaire p. 601), qui a plus de chances d'exister à l'avenir, car maintenu sur le long terme par une communauté de programmeurs, contrairement à un logiciel propriétaire à la merci d'une politique commerciale qui peut enterrer un environnement, du jour au lendemain.

Quelles seraient selon vous les caractéristiques principales nécessaires à votre œuvre dans un tel environnement ?

Il y a deux dimensions : la dimension technique et la dimension stratégique. Au niveau technique, il y a assez peu de contraintes finalement. Il faut juste que ce soit un environnement permettant de fabriquer des programmes autonomes qui fonctionnent, et non pas des films, par exemple, par opposition. Un environnement qui permette de faire tourner des programmes et qui permette de dessiner sur un écran, de faire quelque chose de visuel, de gérer le dessin de ces graphiques. Ça c'est pour côté technique. Au niveau stratégique, je reviens encore une fois à cette idée d'open-source : pour moi, il est stratégiquement beaucoup plus intelligent de travailler dans un environnement open-source que dans un environnement propriétaire, pour des raisons de maintenance future.

Est-ce que pour vous, l'utilisation du code source original, même si cela pourrait impliquer un changement perceptible, est importante pour *still living* ?

Pour cette œuvre-là, je pense que le plus important est effectivement de préserver non pas le code source en tant que code mais l'algorithme du mouvement, sachant qu'évidemment dans cette œuvre-là il y a l'algorithme du mouvement lui-même, donc le petit programme qui génère du mouvement en tant que tel. Il y a aussi toute la dimension de représentation de ce mouvement de manière graphique, qui induit aussi des bouts de programmes, qui dessine les courbes. Ces choses sont complètement secondaires et peuvent être entièrement transformées, remplacées par d'autres. Là vraiment, c'est sans importance. Par contre, ce qui est très important, c'est l'algorithme central. On peut très bien imaginer d'utiliser le même code source, évidemment, mais on peut également imaginer de le transposer dans un autre langage, à partir du moment où il préserve l'algorithme et sa dynamique de fonctionnement.

En ce qui concerne la composition de l'image, la représentation, quels sont les éléments qui constituent une version intègre de votre pièce ?

Dans l'image, ce qui est important ce sont les couleurs, avec une certaine latitude, parce que ce sont des couleurs franches, assez basiques : du rouge pur, du bleu pur, etc. ; les couleurs, donc, ainsi que la composition générale, au sens traditionnel du mot composition, comme l'emplacement des différents éléments dans l'espace. Ce qui est également important et dont je n'ai pas encore parlé c'est le cartel, la légende. Dans l'image, il y a non seulement des formes graphiques mais aussi un élément typographique, textuel, qui est le titre de l'œuvre inclus dans l'image, avec une certaine police de caractère et l'heure de l'ordinateur. L'heure présente est un élément vraiment important dans l'œuvre, parce qu'il est subliminal et qu'il permet au spectateur de savoir – même si ce n'est pas un savoir conscient – que ça se passe ici et maintenant et que ce n'est pas un enregistrement : c'est l'heure présente et le spectateur peut s'en rendre compte. Ces différents éléments, la police de caractère, les couleurs, la composition, sont donc des éléments importants de l'image.

Et par rapport à la résolution de cette image ?

Disons que ce n'est pas fondamental. Ce qui est important, c'est de respecter au maximum ce qui est possible : si un jour il faut faire tourner cette œuvre sur des écrans de résolution beaucoup plus grande, non prévue au départ, il sera nécessaire de simuler la résolution originale. L'idée c'est de quand même garder le grain, le détail tel qu'il est aujourd'hui ainsi que la taille, parce qu'évidemment, si on réduit l'image à la résolution d'un écran qui en a une beaucoup plus grande, l'image sera beaucoup plus petite, et ça n'ira pas. L'idéal est de garder la dimension physique de l'image telle qu'elle était au départ au maximum. On est donc obligé de s'adapter aux résolutions plus grandes et de simuler une résolution plus basse sur une forme géométrique qui reste constante.

Que pensez-vous de la vitesse du déroulement de cette image, de la fluidité du mouvement ?

Pour le mouvement, il est important de ne pas descendre en dessous d'une certaine dynamique, d'une certaine puissance de calcul. Dans le cas contraire, les mouvements vont être ralentis, ramollis et cela sera perceptible. Par contre, si on va sur des ordinateurs beaucoup plus puissants, il n'y aura aucun problème, car j'ai bridé la vitesse de défilement, de calcul, pour que les mouvements ne soient pas plus rapides. Mais si l'ordinateur n'est pas assez puissant, alors évidemment il n'arrivera pas à

calculer assez vite et par conséquent les mouvements seront ralentis. C'est donc là où se situe la problématique.

Par rapport aux données externes au programme, vous avez mentionné l'importance de l'heure qui s'affiche. Est-ce qu'il y a d'autres données externes à votre programme qui interagissent avec celui-ci ou non ?

Il n'y a pas d'autres éléments extérieurs : il n'y aucune donnée provenant d'Internet ou liées au spectateur, ni à un capteur quelconque. Le programme est donc totalement autonome, sauf pour l'heure. D'ailleurs, il est important que l'ordinateur soit à l'heure, parce que le programme prend l'heure de l'ordinateur pour l'afficher. Il est donc important de bien le vérifier quand on expose l'œuvre, car un décalage peut se produire. Soit on utilise un ordinateur connecté sur Internet qui se met à l'heure automatiquement, soit on vérifie régulièrement que l'ordinateur est plus ou moins à l'heure. On ne va pas non plus être trop précis, à la seconde près, mais il faut au moins que le spectateur sente que c'est l'heure de maintenant.

Pensez-vous que vous devriez être systématiquement consulté à propos d'une restauration, d'une reprogrammation ?

Ce n'est pas indispensable, mais autant que possible, oui.

De façon plus générale, quelle est votre approche de la conservation et de la restauration ?

C'est quelque chose dont je prends de plus en plus conscience. J'en ai pris conscience de manière assez forte le jour où j'ai voulu exposer une œuvre et que je n'ai pas pu, n'ayant pas trouvé d'ordinateur pour la faire fonctionner. Cette œuvre était faite sur des ordinateurs – ou plutôt des systèmes d'exploitation – qui ne se font plus aujourd'hui. À partir de ce moment-là, j'ai donc été amené à me poser cette question. J'ai plusieurs approches concernant ces problématiques. Aujourd'hui, je suis passé à un environnement open-source pour mon développement. Je fais quasiment tout avec Processing [langage de programmation open-source] – pour ne pas le nommer – qui est un environnement très polyvalent me permettant de faire tout ce que je veux. Dans la mesure où il est open-source, il a – *a priori* – une durée de vie supérieure à celle d'un environnement propriétaire.

Avez-vous des préférences pour une ou plusieurs stratégies de conservation comme le stockage, la migration, l'émulation, la reprogrammation, la réinterprétation ou des stratégies hybrides ? *A priori*, pensez-vous que certaines stratégies de conservation sont plus pertinentes pour votre travail en général et pour *still living* en particulier ?

Je pense qu'il faut toujours privilégier les stratégies qui préservent l'œuvre, c'est une espèce de tautologie : il faut rester au plus proche de l'existant. Mais je n'ai pas vraiment de préférence. Toutefois, j'aime bien l'émulation, parce qu'elle préserve en quelque sorte le cocon dans laquelle l'œuvre existe. Si on pouvait émuler sans transformer... Tout portage d'œuvre ou réécriture va forcément impliquer des modifications dans le matériau lui-même, alors qu'émuler un système, c'est plutôt comme découper ce qui est autour de l'œuvre, puis le transporter. On peut imaginer que ça va moins altérer l'œuvre, mais ce n'est

pas aussi évident, car l'émulation a toujours des défauts qui ont tout de même une influence sur l'œuvre. Je dirai donc que je n'ai pas, *a priori*, de préférences.

Y a-t-il d'autres éléments autour de ces questions qui vous semblent importants de mentionner ?

Oui, il me semble important de parler des matériaux qui constituent l'œuvre et du fait qu'aujourd'hui, dans ma stratégie de conservation j'ai décidé de fournir l'œuvre en tant que programme exécutable, de fournir son mode d'emploi, c'est-à-dire des instructions d'installation technique et esthétique. Cela permet aux gens qui l'exposent, qui l'ont acheté et qui veulent l'exposer d'avoir des garde-fous et de disposer de préconisations sur les meilleures conditions d'exposition et les limites acceptables dans une certaine mesure, même si c'est subjectif, pour l'exposition elle-même et pour la préservation de l'œuvre telle quelle. Pour ce qui est de la conservation dans le futur, je fournis le code source de l'œuvre dans son environnement actuel, Director, ce qui facilite une maintenance dans un futur plus lointain pour l'acheteur éventuel. À un niveau purement contractuel, l'œuvre est constituée de l'exécutable sur Macintosh et sur Windows, des codes source ainsi que des instructions d'installation. Tout ceci fait partie de ma stratégie par rapport au problème de l'obsolescence et du portage, de la maintenance et de la conservation de l'œuvre. Je trouve que c'était important de le dire. Je n'ai pas d'autres choses à ajouter.