A.0
RÉFLEXION

Extrait tiré de : BOHNACKER, Hartmut, Benedikt GROSS, Julia LAUB et Claudius LAZZERONI (2010). *Design génératif. Concevoir, programmer, visualiser*, Pyramyd, Paris. pp. 460-465.

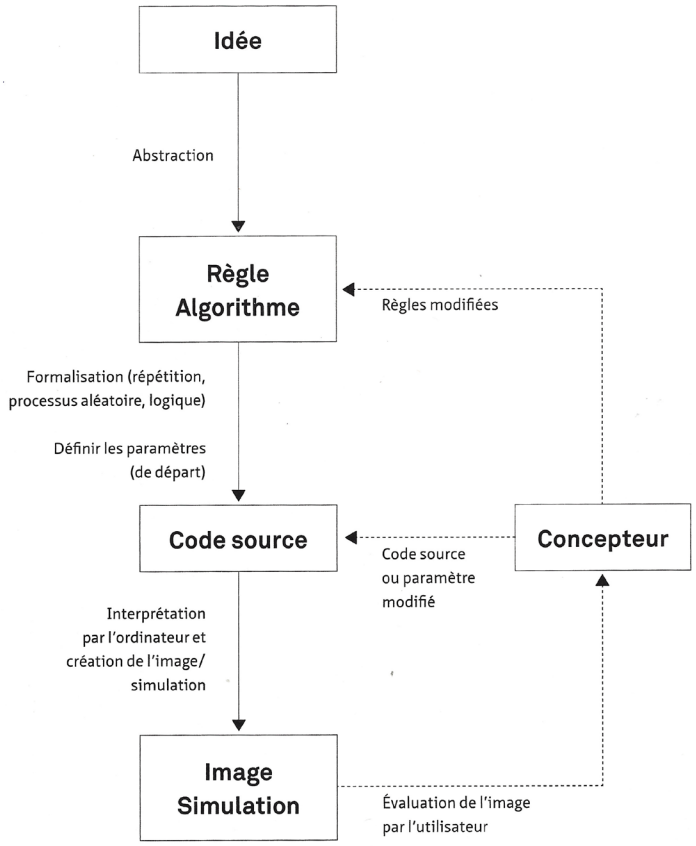
Dans les deux principales parties de cet ouvrage «Principes fondamentaux» et «Méthodes complexes», il a été question d'illustrer de façon pratique comment générer des graphiques. En expérimentant avec les programmes exemples, il est possible de se forger une première représentation du design génératif. Dans les pages suivantes, après le «faire», nous allons nous efforcer de susciter une réflexion sur les programmes exemples et de replacer le contenu acquis en contexte. En outre, cette partie se veut aussi un guide vers des thèmes et des sujets connexes.

STATU QUO

Pour la conception, nous utilisons toujours naturellement cette machine universelle qu'est l'ordinateur. Cependant, les outils les plus répandus en matière de conception comme Adobe Creative Suite, AutoCAD, 3ds Max etc. ont principalement une fonction de représentation et d'illustration. La marche triomphale de l'ordinateur n'a fait que rendre les outils tels que les pinceaux, les ciseaux ou les laboratoires photo virtuels et plus efficaces. Ce faisant, tout est devenu plus rapide et plus confortable, mais en ce qui concerne l'essentiel — à savoir le processus de conception —, aucun changement fondamental n'est intervenu. Souris ou pinceau, le concept, par exemple la création d'une ligne, reste identique. En quoi le design génératif se différencie-t-il d'un procédé de conception conventionnel ? Avant tout, sur deux points : le processus de conception est autre et, de ce procédé modifié, découlent fondamentalement de nouvelles possibilités.

UN PROCESSUS DE CONCEPTION MODIFIÉ

La modification fondamentale apportée par le design génératif au processus de conception concerne l'aspect artisanal, qui se voit relégué à l'arrière-plan, tout en devenant une abstraction et une source d'information pour l'élément principal. La question ne se pose plus tant dans les termes «comment dessiner ?» que «comment rendre abstrait ?». Car le processus permettant de passer de l'idée à l'image terminée implique désormais un ensemble de règles — une couche intermédiaire systématique que l'ordinateur interprète et traite. Chacune des images générées, avant d'apparaître dans la zone d'affichage, doit déjà être complètement décrite par un système de règles. Dès lors, deux problèmes se posent pour le concepteur : premièrement, comment réaliser l'abstraction d'une idée vague ? Et, deuxièmement, comment entrer l'idée formalisée dans l'ordinateur ? Il n'existe aucune recette en matière d'abstraction. Dès qu'une idée complexe doit être implémentée, il existe toujours un moyen de décomposer ce problème en éléments plus petits. Cette stratégie de résolution de problème est également dénommée «diviser pour régner». Exemple : une surface doit être remplie de façon dense au moyen de cercles au diamètre aléatoire sans pour autant que ces cercles se superposent. La première étape consiste à traduire cette idée en une «recette de cuisine» concrète et simple : créer un nouveau cercle ; si celui-ci n'a aucune intersection avec les autres cercles de la zone d'affichage, le dessiner aussi grand que possible ; s'il existe une intersection, recommencer au début. --> P.2.2.5 Cette décomposition permet de formuler les différentes étapes dans un langage de programmation qui autorise l'exécution du programme par un ordinateur. Un langage de programmation nécessite des composantes élémentaires — comme la répétition, la logique et le caractère aléatoire —, sur lesquelles nous allons revenir dans la suite de cette section.



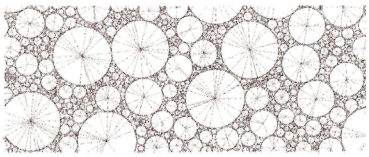
→ Exemple : Structure concentrée à partir d'agents

Idée : créer une structure concentrée de cercles, présentant des similitudes avec des bulles de mousse.

Définir les règles d'abstraction :
1. Créer un nouveau cercle.
2. Si celui n'a aucune intersection avec les autres cercles de la zone d'affichage, le dessiner aussi grand que possible.
3. S'il existe une intersection, recommencer du début.

Formalisation dans la syntaxe du code :

```
for(int i=0; i < currentCount; i++) {
  float d = dist(newX,newY, x[i],y[i]);
  if (newRadius > d-r[i]) {
    newRadius = d-r[i];
    closestIndex[currentCount] = i;
  }
}
```



→ Chap. P.2.2.5
Structure concentrée à partir d'agents

La répétition permet à un ordinateur de travailler sur une tâche le temps nécessaire à sa résolution, ou encore de manipuler une quantité phénoménale d'objets. L'importance de la répétition se reflète clans la fréquence des boucles for implémentées dans les programmes, dans ce livre comme ailleurs, ou encore dans le fait que la boucle Loop constitue la fonction centrale de Processing.

L'aléa permet de créer des variantes ou de briser la régularité rigide de l'ordinateur. L'aléa incontrôlé produit des résultats peu intéressants. Des choses intéressantes surviennent en revanche lorsque l'aléa est employé de façon limité ou dosé --> Chap. M.1.2. Dans Processing, l'aléa est représenté par les mots clés random et noise.

La logique, dans le sens de structures de contrôle, est utilisée pour orienter le processus génératif. Il est ainsi possible de créer des conditions chargées d'orienter le flux du programme dans différentes ramifications. Par exemple, dans le chapitre «Texte sous forme de plan de construction» --> Chap. P.3.1.2, une structure Switch/Case est employée pour exécuter différentes portions du programme en fonction du texte entré : tous les caractères sont écrits normalement, mais chaque signe de ponctuation modifie le sens d'écriture. Le signe de ponctuation est en outre modifié par un élément courbe. Les mots clés les plus fréquemment utilisés pour les structures de contrôle sont if, else, switch et case.

Si l'idée conceptuelle est traduite en code, elle peut dès lors être interprétée par l'ordinateur, et des images émergent du néant, sans que la main ne dessine le moindre trait. Quoi qu'il en soit, un résultat satisfaisant ne s'obtient pas du premier coup. Le résultat doit être évalué, évaluation servant de base à d'éventuelles améliorations. Contrairement au processus conventionnel, il ne s'agit pas de modifier manuellement l'image, mais d'intervenir sur l'abstraction sous-jacente ou sur certains paramètres du programme. Cette interaction entre résultat et créateur affine le système génératif à chaque étape itérative pour aboutir au résultat final.

L'interaction permet d'accélérer plus encore le processus. Un système génératif en soi n'est pas nécessairement interactif, et dans ce livre, l'interaction n'est pas appréhendée de manière explicite. Mais le surcroît d'opérations nécessaire à l'implémentation d'éléments interactifs (boutons, réglettes, etc.) s'avère payant. En effet, la modification des paramètres peut dès lors s'effectuer et être pilotée en temps réel. Raison pour laquelle nous avons choisi, dans la partie «Méthodes complexes», de proposer un outil interactif à la fin de chaque chapitre.

DE NOUVELLES POSSIBILITÉS DE CONCEPTION

Comme les langages de programmation accèdent au statut de composantes du processus de conception, l'espace des possibles s'ouvre considérablement pour le concepteur. Comme par le passé, la compétence en matière de conception reste du côté du concepteur, l'ordinateur assumant le rôle de l'assistant toujours disponible. Qu'aujourd'hui il soit plus rapide et aisé de concevoir grâce à l'informatique ou de créer une composition dotée de milliers d'éléments, plus personne ne s'en étonne dans ce monde numérique et automatisé qui est le nôtre. En revanche, les possibilités du design génératif appréhendées sous l'angle de l'émergence, de la simulation et de l'outil s'avèrent dignes d'intérêt.

Dans le contexte du design génératif, «l'émergence» prend la signification suivante : premièrement, les processus génératifs sont émergents lorsque leur résultat est en devenir et, deuxièmement, lorsque l'interaction des différentes composantes aboutit à une complexité supérieure à celle obtenue à partir des propriétés de ces éléments. Un exemple d'émergence tiré du quotidien : le vol des oiseaux constitue un comportement complexe, dont le développement n'est pas prévisible avec exactitude. Dans ce livre aussi, dans le chapitre «Structures de croissance à partir d'agents» --> Chap. P.2.2.4, un algorithme très simple à deux opérations aboutit à la création de structures organiques totalement inattendues.

La simulation des règles de la nature constitue une autre méthode très prisée dans le design génératif. Ainsi, nous utilisons dans le chapitre «Structures de données dynamiques» --> Chap. M.6 un graphe de placement par force pour visualiser la structure des liens d'articles Wikipedia. L'idée d'un graphe de placement par force consiste à répartir, par la simulation d'un modèle physique — ici l'élasticité et la force de répulsion --> Chap. M.4 —, un diagramme de structure sur une surface. Tous les éléments s'influencent les uns les autres, jusqu'à ce que la structure conforme aux lois physiques aboutisse à un état de tension équilibré. Ce qui frappe avant tout, c'est que le détournement

des deux formules physiques aboutit de manière autonome à la résolution du problème. Comme il arrive souvent, la transposition d'un domaine de savoirs à un autre génère des résultats étonnants. Quoi qu'il en soit, il existe encore dans la nature de nombreux autres modèles pouvant être transposés avec succès à un système génératif.

Peut-être que l'aspect le plus important de l'extension des possibilités pour le concepteur est que, désormais, le créateur peut être son propre «outil». En effet, tout programme génératif est aussi simultanément un outil logiciel personnalisé, grâce auquel le concepteur peut ouvrir de nouvelles voies et contribuer au panel des médias visuels. Il est étonnant de constater le niveau de dialogue possible avec des outils développés, comme par exemple dans le chapitre «Dessiner» --> Chap. P.2.3. Même ces simples outils de dessin étendent considérablement les possibilités. En outre, étant donné la capacité des concepteurs d'affiner sans cesse ces outils, ceux-ci deviendront des instruments de travail parfaitement opérationnels. Par ailleurs, le pas n'est pas si grand séparant un programme génératif d'un outil exploitable (voir les outils proposés dans la partie «Méthodes complexes»).

PERSPECTIVE
Il semble clair que le design génératif sera amené à se généraliser. Le portfolio proposé dans la première partie illustre clairement qu'il est déjà utilisé dans nombre de domaines. Bases de visualisation, d'information, œuvres d'art, installations médiatiques, modèles d'architectures, clips vidéo, polices..., le champ d'application ne cesse de s'étendre, jusqu'à des concepts de lampes génératives. En outre, différents facteurs favorisent ce développement.

Le volume d'informations auquel nous sommes confrontés au quotidien va encore augmenter avec l'essor des systèmes en réseau dont l'accès se généralise toujours plus. Le flux d'informations sous forme de données préparées en vue de leur visualisation est une tâche cruciale pour la conception. Le design génératif trouve ici une application incontournable.

L'extension des possibilités techniques conduit à des impulsions continues en faveur du design génératif Ainsi, il y a encore quelques années, il était pratiquement impossible de créer des mondes tridimensionnels complexes ; aujourd'hui, ces créations sont réalisables, même sur un ordinateur portable. Ce potentiel technique va contribuer à assurer l'essor du design génératif

Par ailleurs, il convient de ne pas sous-évaluer la dynamique de la collaboration, à l'œuvre dans le champ du design comme dans peu d'autres. Il est étonnant de constater le nombre de communautés web ayant participé à l'élaboration d'outils génératifs comme Processing ou VVVV, sous forme de bibliothèques, de didacticiels, de programmes-exemples, d'articles, de contributions au sein de forums, etc. Une partie de cette dynamique repose sur le fait que le design génératif, par son principe même, est très adapté au processus collaboratif. En effet, le code en tant que support de conception convient parfaitement au travail en équipe impliqué dans le développement logiciel. En outre, l'un des avantages du code par rapport à d'autres médias — les données vidéos par exemple — est qu'il est beaucoup plus léger et facile à diffuser.

Néanmoins, il convient de garder la chose suivante à l'esprit : les concepteurs qui programment ne sont encore qu'une infime minorité. Ce phénomène s'explique par des raisons historiques et culturelles : on est soit artiste, soit technicien. Il est difficilement envisageable d'être les deux, comme le prouvent aujourd'hui encore les cursus de formation. En outre, ce nouveau processus de conception consistant à partir d'un code source mathématique-analytique pour aboutir à un résultat visuel reste encore pour beaucoup un obstacle insurmontable. Contribuer à le dépasser est l'un des objectifs de ce livre.

Nous sommes très optimistes quant à l'avenir du design génératif. Et nous pouvons nous réjouir de voir que le plus gros obstacle, à savoir la transposition technologique sous forme d'outils de développement appropriés et l'amélioration des performances des ordinateurs, est déjà tombé. Le design génératif va devenir une évidence grâce au potentiel des nouvelles générations d'ordinateur. Nous adhérons au point de vue selon lequel la programmation et, avec elle, le design génératif, va s'imposer comme une technique universelle, à l'instar de la photographie et du film au siècle précédent. Les possibilités existent ; à nous de les utiliser.