
.intuition, intention, programmation

Intuition, intention, programmation.

Une introduction à la programmation
en préambule à l'utilisation du logiciel PROCESSING,
(environnement et langage de programmation open-source,
librement téléchargeable).

Erg (École de Recherche Graphique) - Bruxelles.
Arts numériques 1e, 2e, 3e & 4e année.
Professeur: Marc Wathieu.

Mise à jour: 10 novembre 2006.

Ce livret PDF a été conçu comme un diaporama destiné à être projeté et commenté.
Pour un affichage optimisé, je vous recommande une résolution 1024 X 768,
une visualisation avec Acrobat Reader
et le raccourci ctrl+l (Windows) ou pomme+l (Mac OSX).

Télécharger ici Acrobat Reader.

---

http://www.multimedialab.be

«L'intuition, c'est ce qui relie notre conscience à notre inconscient. C'est le subconscient qui surgit et nous prévient que l'on devrait réfléchir à quelque chose. C'est l'intuition qui nous ouvre continuellement de nouvelles portes de la pensée». [1]

Richard Buckminster Fuller *(1895 - 1983),*
architecte, designer, inventeur et écrivain américain.

[1] in «Buckminster Fuller: scénario pour une autobiographie», Robert Snyder, éditions Images Modernes, Paris, 2004.

«[...] Pour les artistes qui créent par la programmation, le langage logique sert d'intermédiaire entre l'intention et l'œuvre finale, de la même manière que les compositeurs créent au moyen de notes de musique. La méthode de travail se dissocie de l'expérience sensorielle par le fait que la tâche de l'artiste consiste à orchestrer un ordre symbolique au moyen d'un code d'écriture, plutôt que par l'interaction physique entre le matériau (telle la peinture) et les sens. Cela oblige l'artiste à traduire de manière conceptuelle les événements du monde réel en séquences complexes de décisions assujetties à des règles ainsi qu'à visualiser par anticipation l'image d'une manière tellement précise que même le hasard doit être déterminé et codé. Une fois ces séquences logiques et ces commandes emmagasinées dans la mémoire, toute erreur sur le plan esthétique ou logique peut être corrigée par simple modification du code, modification possible parce que l'ordinateur permet de retracer la démarche entreprise et de la reprendre [...]».

George Legrady,
«La synthèse: image, langue et croyances»,
in «George Legrady: From Analogue to Digital» (CD-Rom),
National Gallery of Canada, 1998.

‣ Accéder à cet article en ligne.

George Legrady est artiste et professeur d' «Interactive Media»
au «Media Arts & Technology program» de l'université de Santa Barbara, Californie.

«Vous avez, tous, déjà vu un film projeté à l'envers, avec des gens qui sortent de l'eau et se retrouvent sur le plongeoir.

Je vais vous projeter, à l'envers, un film dont vous êtes l'acteur. Vous venez de prendre votre petit-déjeuner; la nourriture sort de votre bouche et retourne dans l'assiette; les assiettes et les bols retournent sur le plateau; tout repasse dans la poêle, et regagne le réfrigérateur; sortis du réfrigérateur, les aliments sont réemballés et réintègrent les rayons du supermarché, la marchandise retourne chez le grossiste, puis à l'usine, puis sur des camions, des bateaux, pour finalement redevenir, par exemple, des ananas à Hawaï.

Puis les ananas se désagrègent, retournent dans l'air; les gouttes d'eau regagnent le ciel, et ainsi de suite.

Si l'on remonte rapidement le temps, il ne faut pas beaucoup plus d'un mois pour que tout ce qui se trouve sur la table, devant vous, et qui est destiné à constituer petit à petit vos cheveux, votre peau, etc., pour que tout cela redevienne de l'air au-dessus des montagnes.

Pour, qu'en d'autres termes, vous soyez comme totalement dispersé. Je voudrais que vous commenciez à vous intéresser à vous-même comme à chacun de ces éléments.

S'il était possible de suivre à la trace les particules chimiques sur un film, vous pourriez les voir se rapprocher de plus en plus, jusqu'à se faire légumes et viandes, entrer dans des boîtes de conserves, à l'étal des supermarchés, et, finalement, devenir ce que nous sommes vous et moi - devenir pour un temps cheveux, oreilles, ou peau - et puis se disperser, se dissoudre, se réduire en poussière.

Chacun de nous est une entité structurelle complexe, programmée dès la naissance». [1]

Richard Buckminster Fuller *(1895 - 1983),*
architecte, designer, inventeur et écrivain américain.

[1] *in «Buckminster Fuller: scénario pour une autobiographie», Robert Snyder, éditions Images Modernes, Paris, 2004.*

Dans ce document, le texte en rouge est cliquable.

Avant-propos.

Les nouvelles technologies engendrent de profondes mutations pratiques et conceptuelles, tout en posant des questions d'accès aux outils et aux moyens de production: apprentissage, détournement, high/low-tech, etc. Compte-tenu des spécificités pédagogiques de l'Erg (interdisciplinarité et recherche), l'atelier d'arts numériques est le lieu privilégié des expérimentations dans ce domaine.

Après quelques approches pédagogiques hybrides, j'en suis arrivé à repenser le cours, en utilisant comme point de départ ce matériau singulier et originel: le code. Pour des étudiant(e)s peut-être fâché(e)s avec les mathématiques et la logique, une initiation à la programmation peut sembler une entrée en matière un peu brutale... C'est en effet une étape sensible mais captivante, dont vous pourrez assez rapidement cerner et mesurer les enjeux.

Á moyen terme, le gain est énorme: l'acquisition d'une méthode de travail, et surtout une compréhension approfondie de l'outil numérique.

Par un effort de documentation, j'espère vous aider à progresser de manière pertinente dans des technologies parfois longues à apprivoiser, mais dont les possibilités surprenantes valent tous les détours.

En attendant, pour démarrer, il faut une étincelle : l'objectif de ce livret est de contribuer à mettre le feu aux poudres...

Inspiré par l'article de Georges Legrady et la réflexion de Buckminster Fuller, j'ai structuré cette introduction à la programmation en 3 phases :

Intuition.

C'est une impression diffuse : le monde qui nous entoure semble structuré, modulaire, séquencé. Toute forme entière peut être perçue comme un ensemble complexe, susceptible d'être recomposé par un enchainement d'élément plus simples. En tentant de susciter votre curiosité, ce premier chapitre va poser une série de repères en images, destinés à identifier des indices, à les faire émerger et à les mettre en relation, afin d'ancrer et d'interpréter cette probable intuition. Les notions de géométrie, de mesure, de paramètres ou de valeurs sont ici sous-jacentes, ce qui explique l'approche plutôt formelle du classement des images.

Intention.

Après avoir distillé cette intuition, l'étape suivante suppose une intention, une motivation, un objectif, donc un projet. Ce chapitre va montrer comment la notion d'algorithme formalise une stratégie à adopter pour obtenir un résultat.

Programmation.

Une fois le projet défini dans sa chronologie, la programmation est la traduction en langage informatique de la procédure à suivre pour arriver au résultat souhaité. Il s'agira donc de se familiariser avec la structure et la syntaxe d'un langage.

Le langage choisi ici est également un environnement de programmation : il s'agit de PROCESSING, créé par Casey Reas et Benjamin Fry. Ce choix s'est imposé pour plusieurs raisons :
- PROCESSING est un outil créé par des artistes pour des artistes.
- PROCESSING est un logiciel libre et open-source*, gratuit et accessible.
- PROCESSING favorise l'expérimentation l'échange d'information.
- une communauté croissante d'utilisateurs enrichi sa documentation.

Accéder au site de PROCESSING :
http://processing.org/

** Concernant la culture libre (logiciel libre, open-source...), voir l'article de Nicolas Malevé en annexe.*

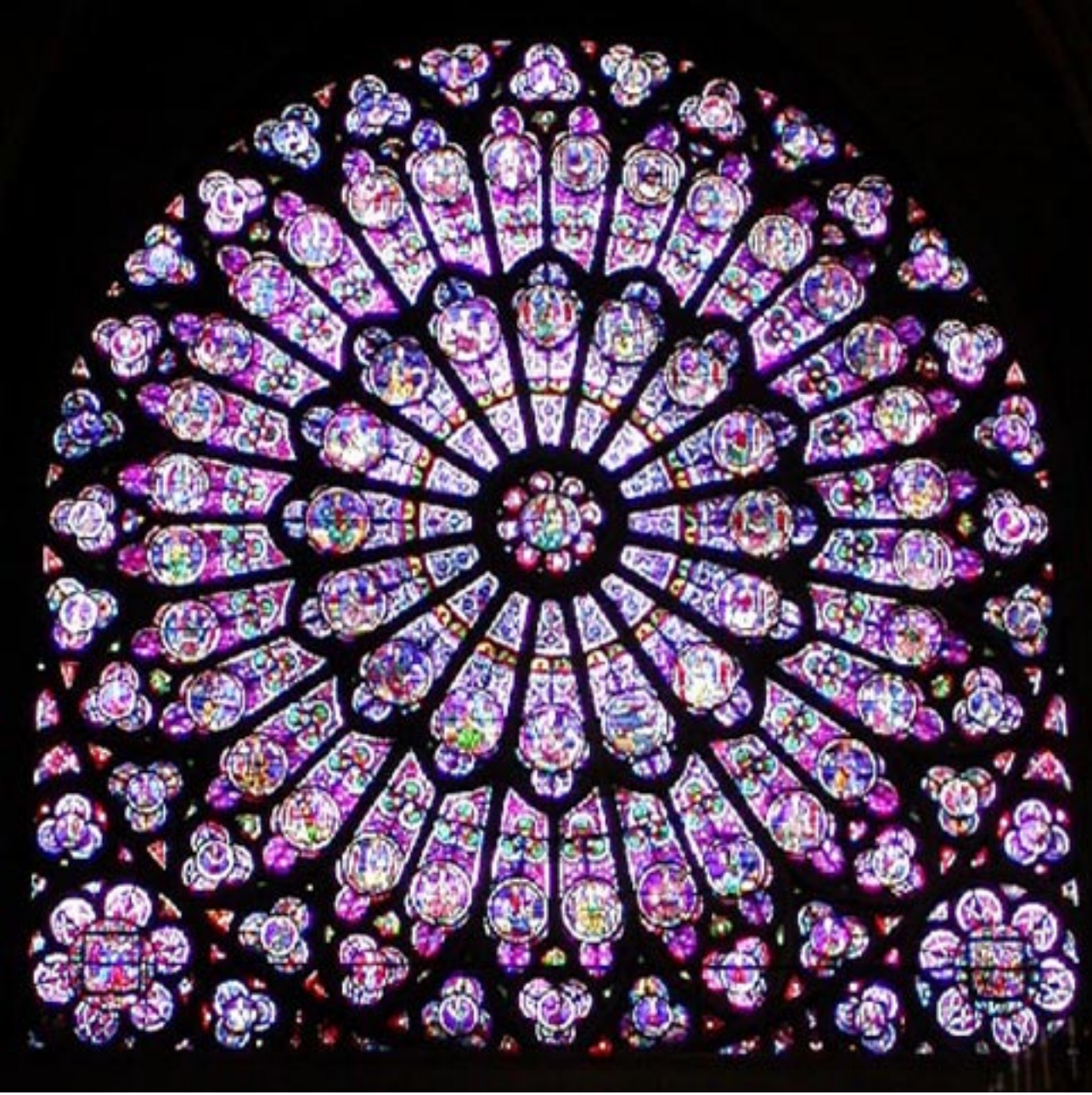
Intuition.

Points, grains, grilles.

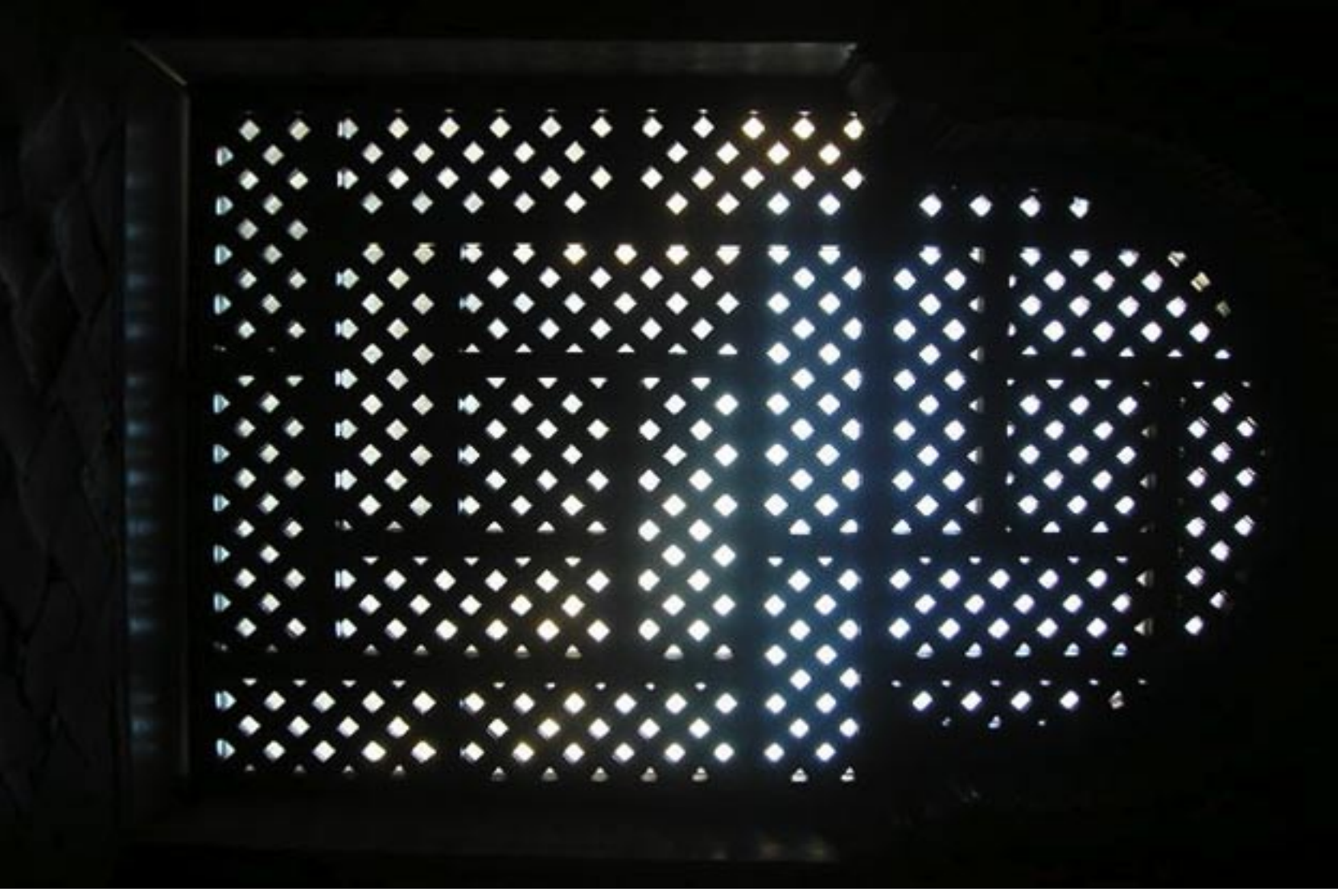
http://www.etacarina.co.za/my-astronomy/my_astronomy.htm

«Cave Canem», Pompei.
Musée National d’Archéologie de Naples.

La tapisserie de Bayeux (1066-1082).
http://fr.wikipedia.org/wiki/Tapisserie_de_Bayeux



Notre-Dame de Paris (1163-1345).

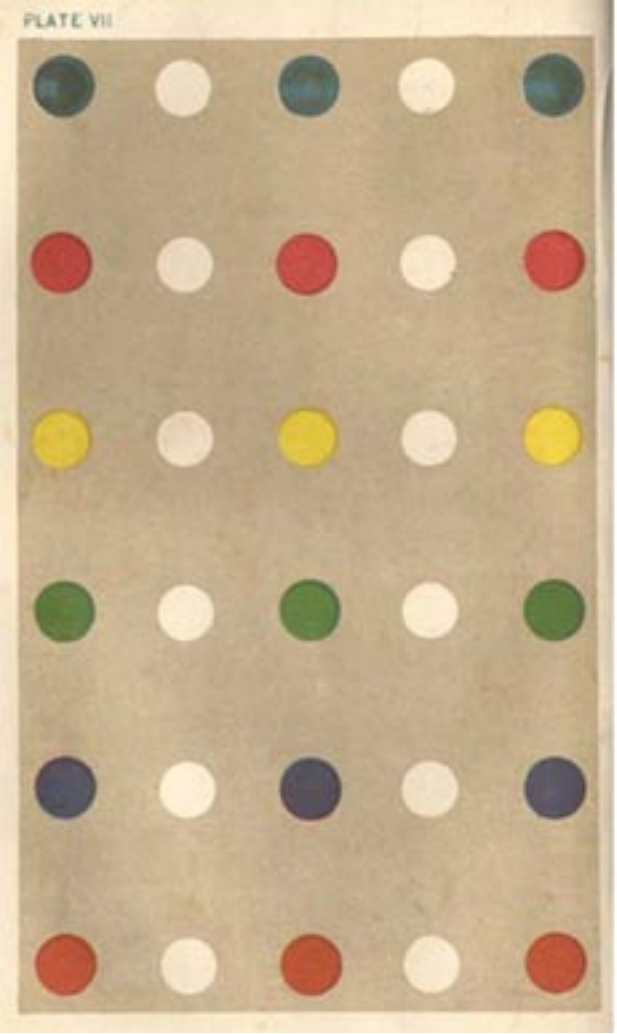


Alhambra de Grenade (XIVe siècle).
http://fr.wikipedia.org/wiki/Alhambra

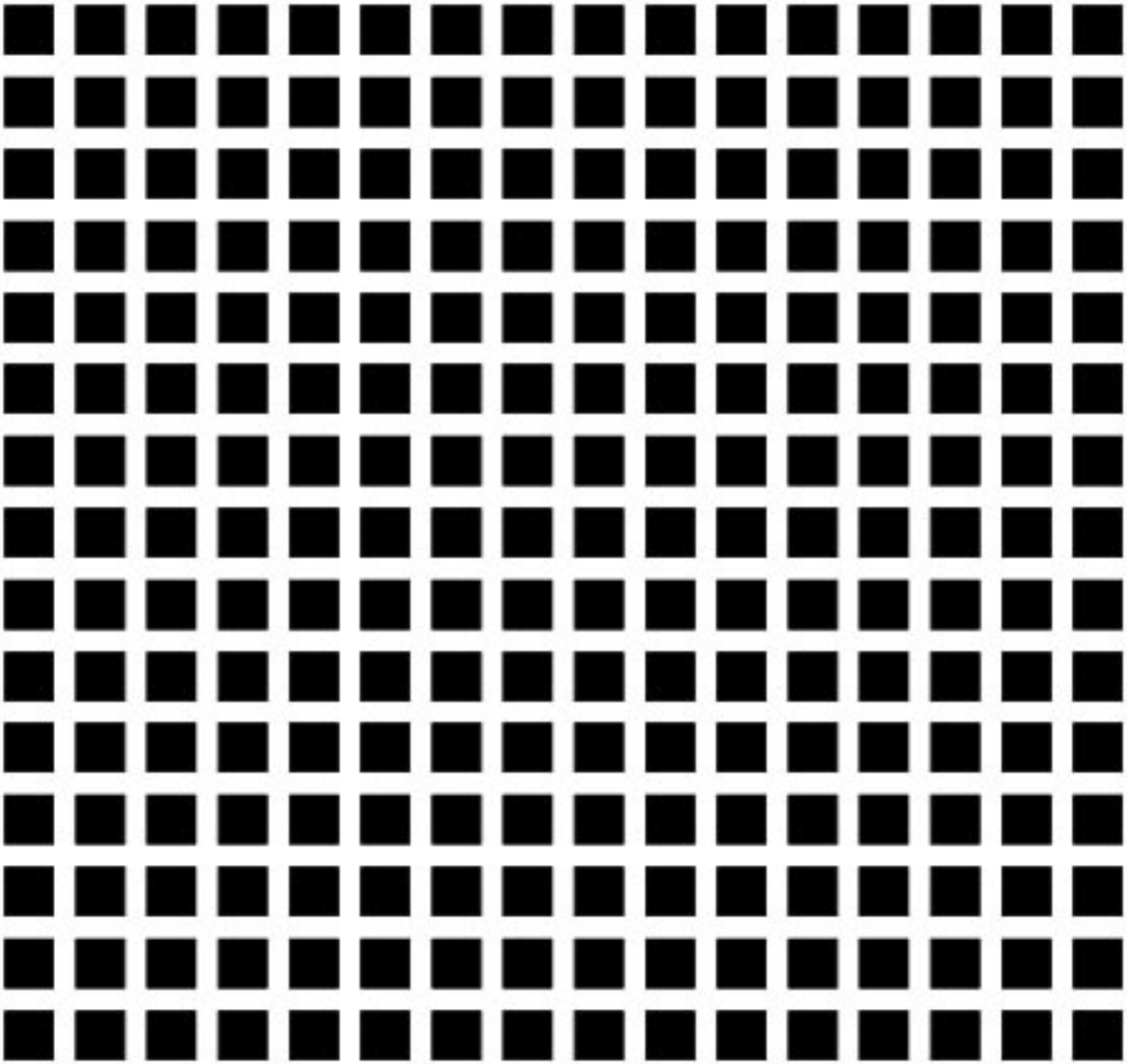
Alhambra de Grenade (XIVe siècle).
http://fr.wikipedia.org/wiki/Alhambra

Albrecht Dürer : Vier Bücher von menschlicher Proportion (1538).
http://en.wikipedia.org/wiki/Albrecht_Dürer
http://www.acmi.net.au/AIC/DRAWING_MACHINES.html

Joseph Marie Jacquard : métier à tisser semi-automatique (1807).
http://en.wikipedia.org/wiki/Jacquard_loom

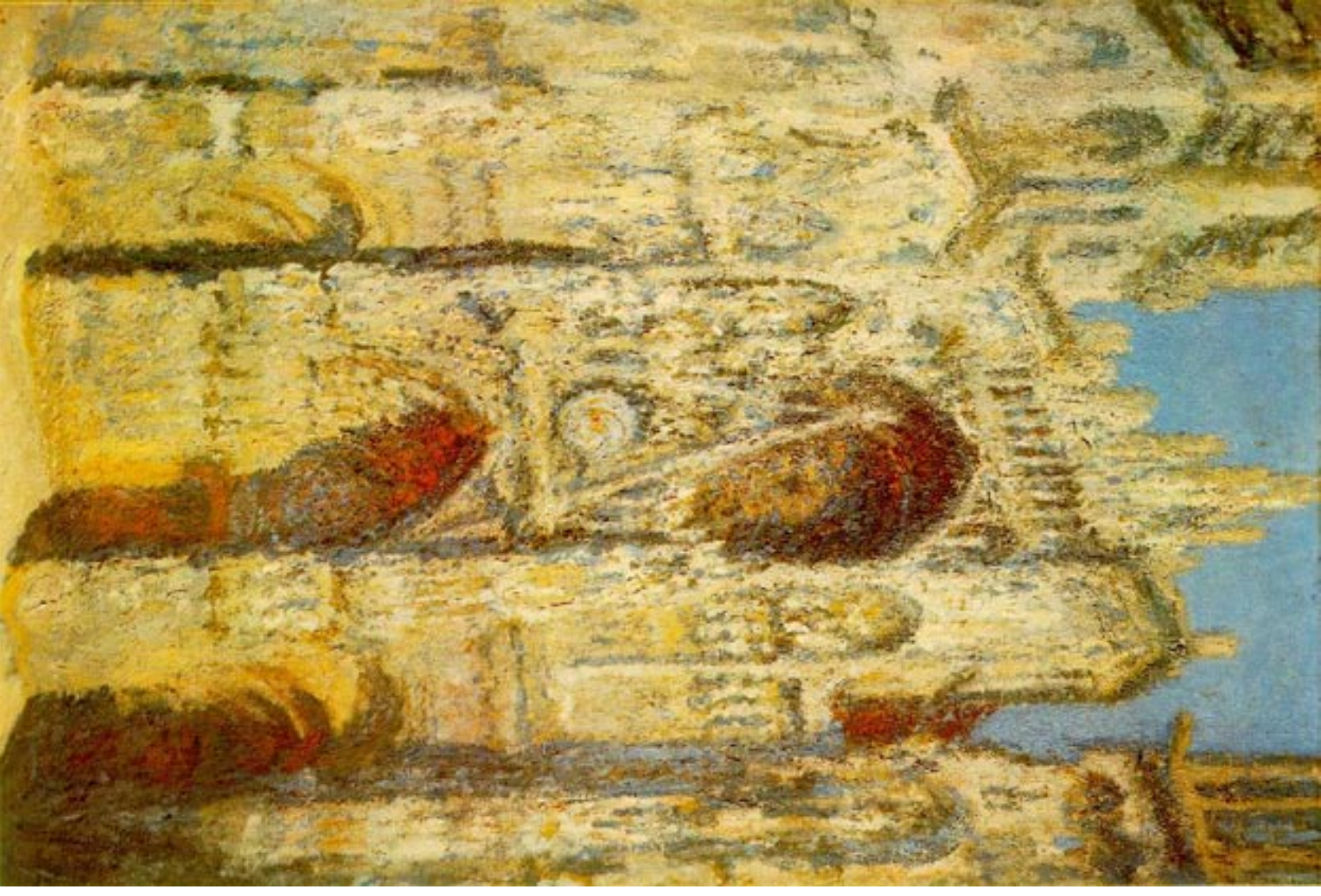


Eugène Chevreul : Planches extraites
de «De la loi du contraste simultané des couleurs
et de l'assortiment des object colorés» (1839).
http://www.fulltable.com/vts/c/cbk/c/c.htm

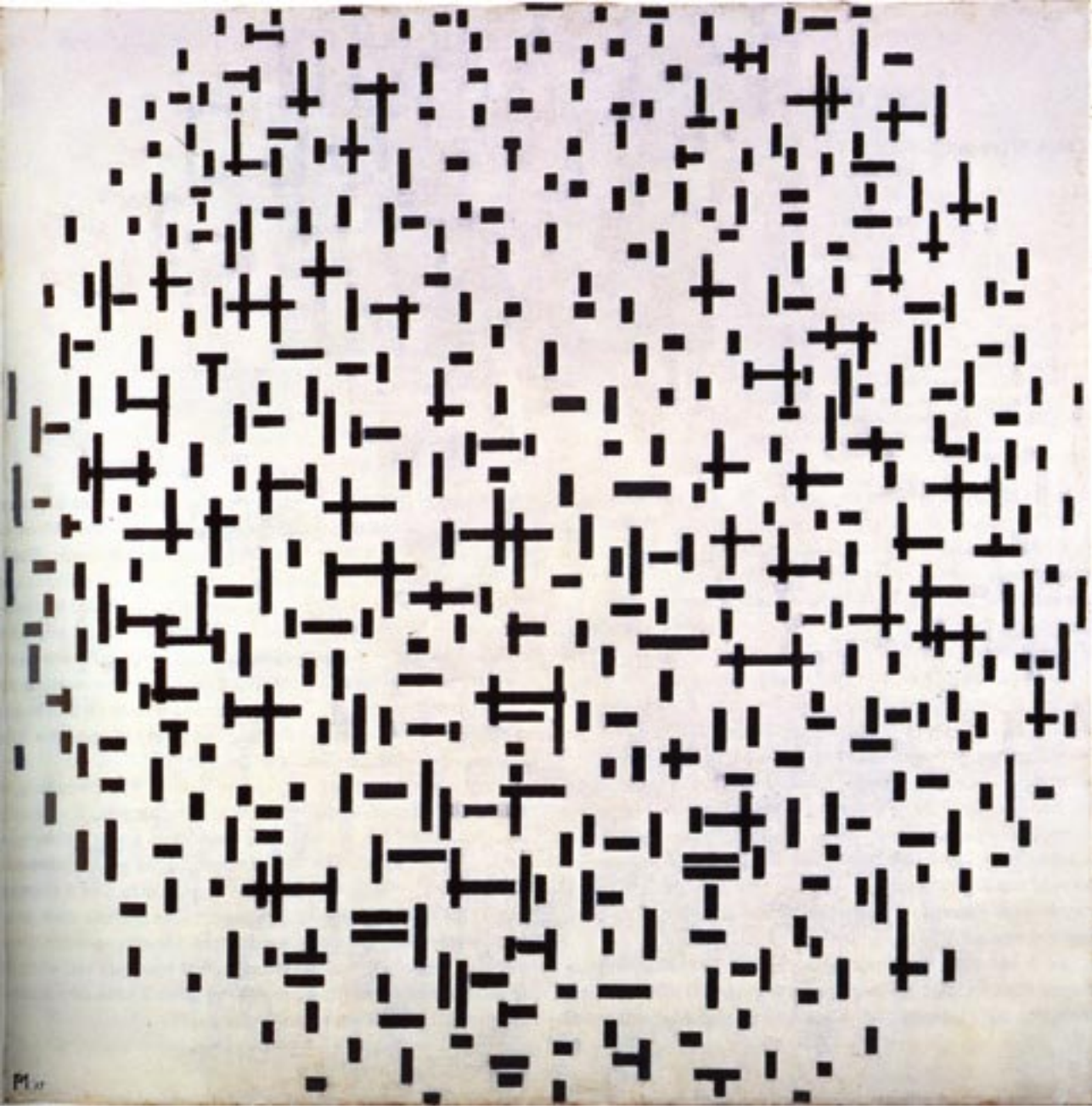


Ludimar Hermann : Grille de Hermann (1870).
http://mathworld.wolfram.com/HermannGridIllusion.html

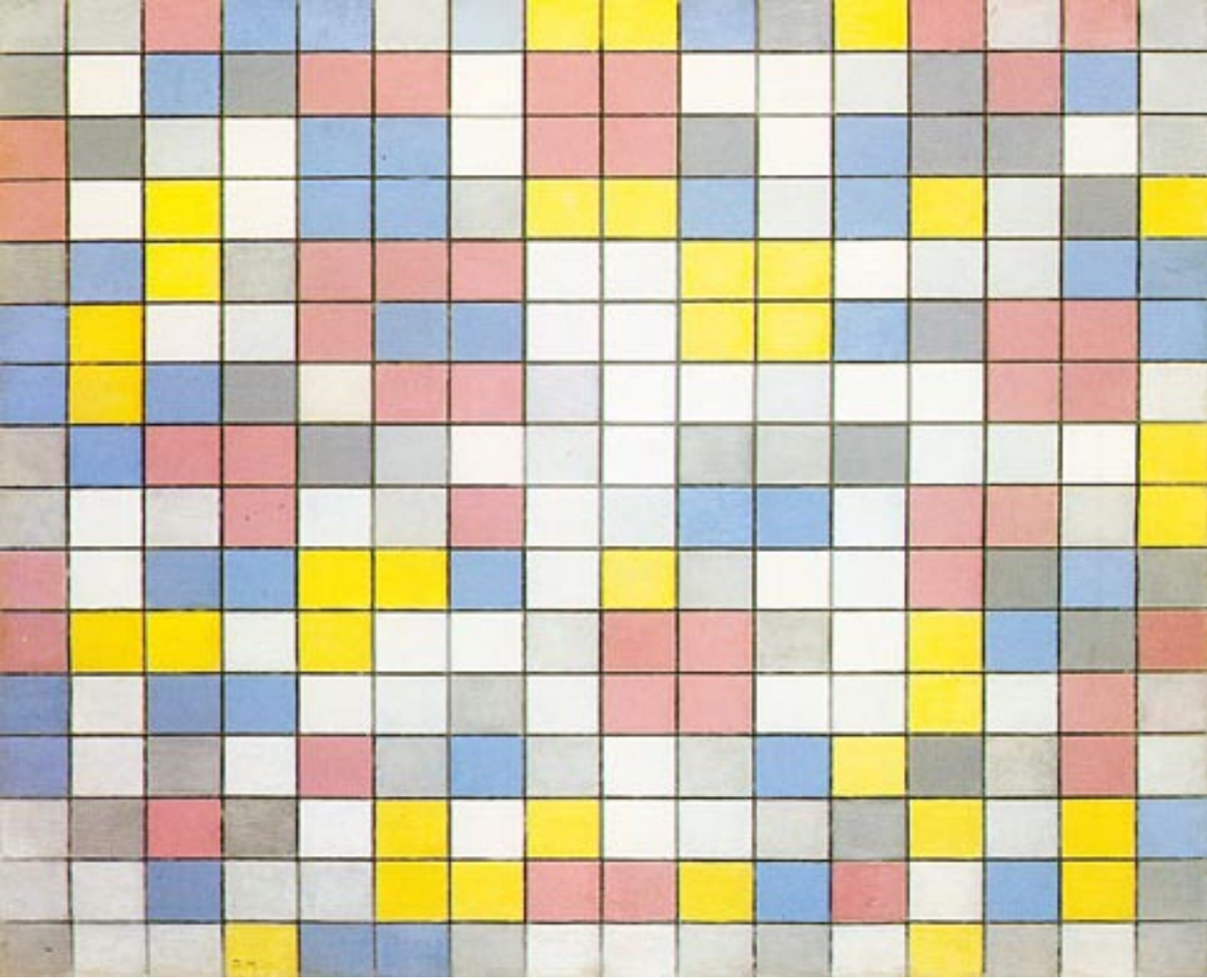
Georges Seurat : La Seine à la Grande Jatte (1888).
http://www.artchive.com/artchive/S/seurat/spring.jpg.html



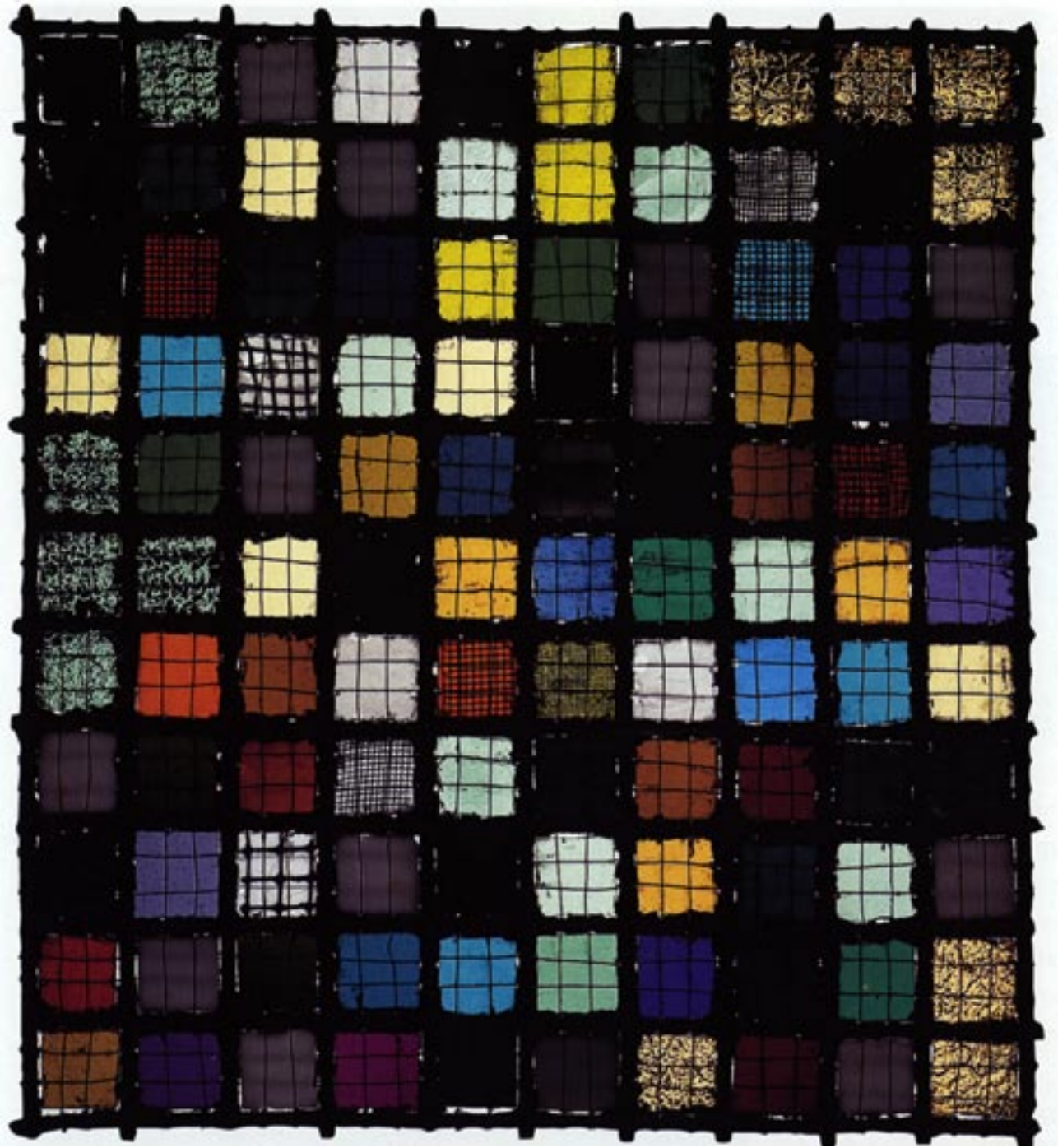
Claude Monet : Cathedrale de Rouen, plein soleil (1893).
http://arthistory.westvalley.edu/images/M/MONET/ROUEN2.JPG



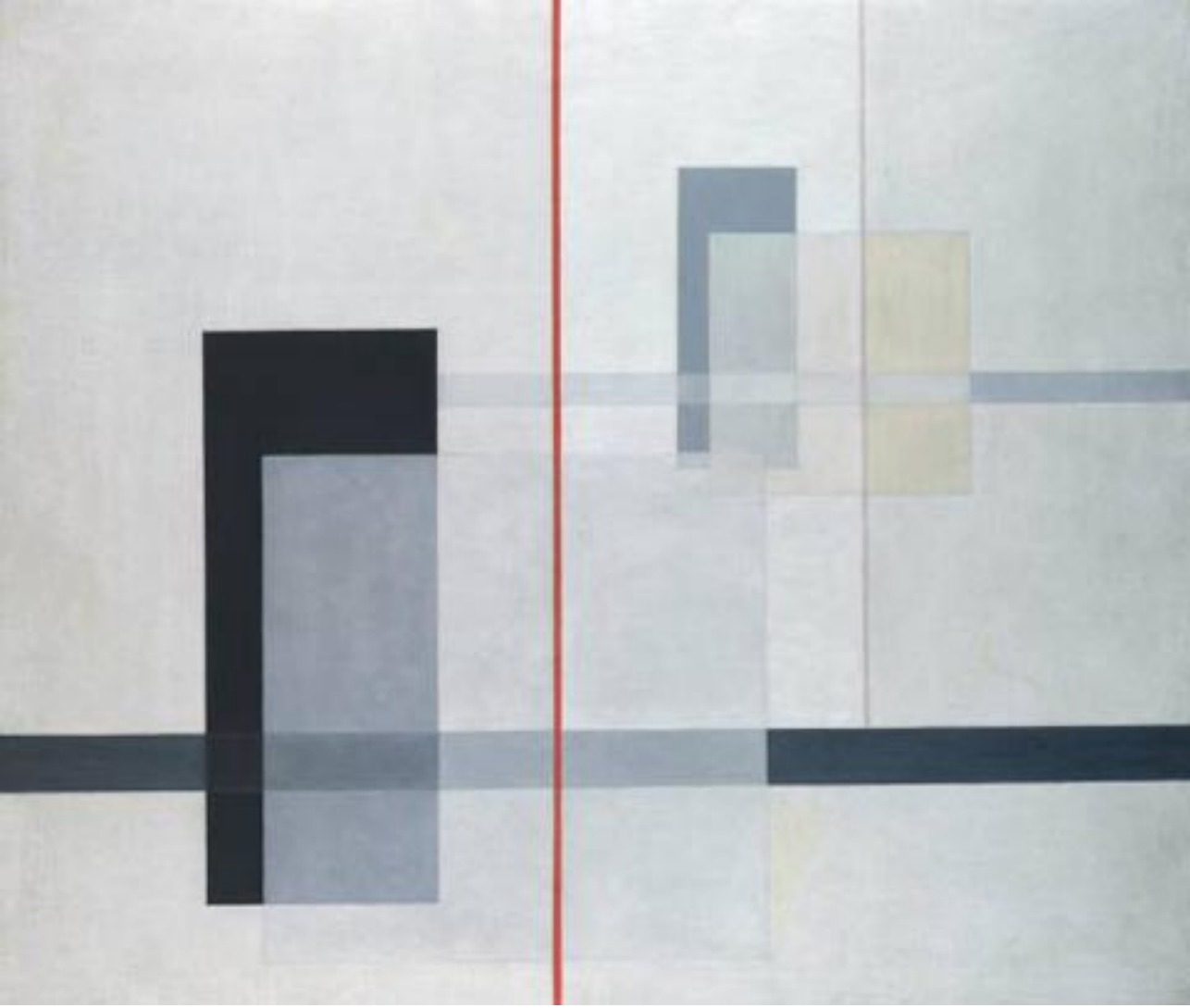
Piet Mondrian : Compositie in lijn (Composition de lignes) (1917).
http://en.wikipedia.org/wiki/Piet_Mondrian



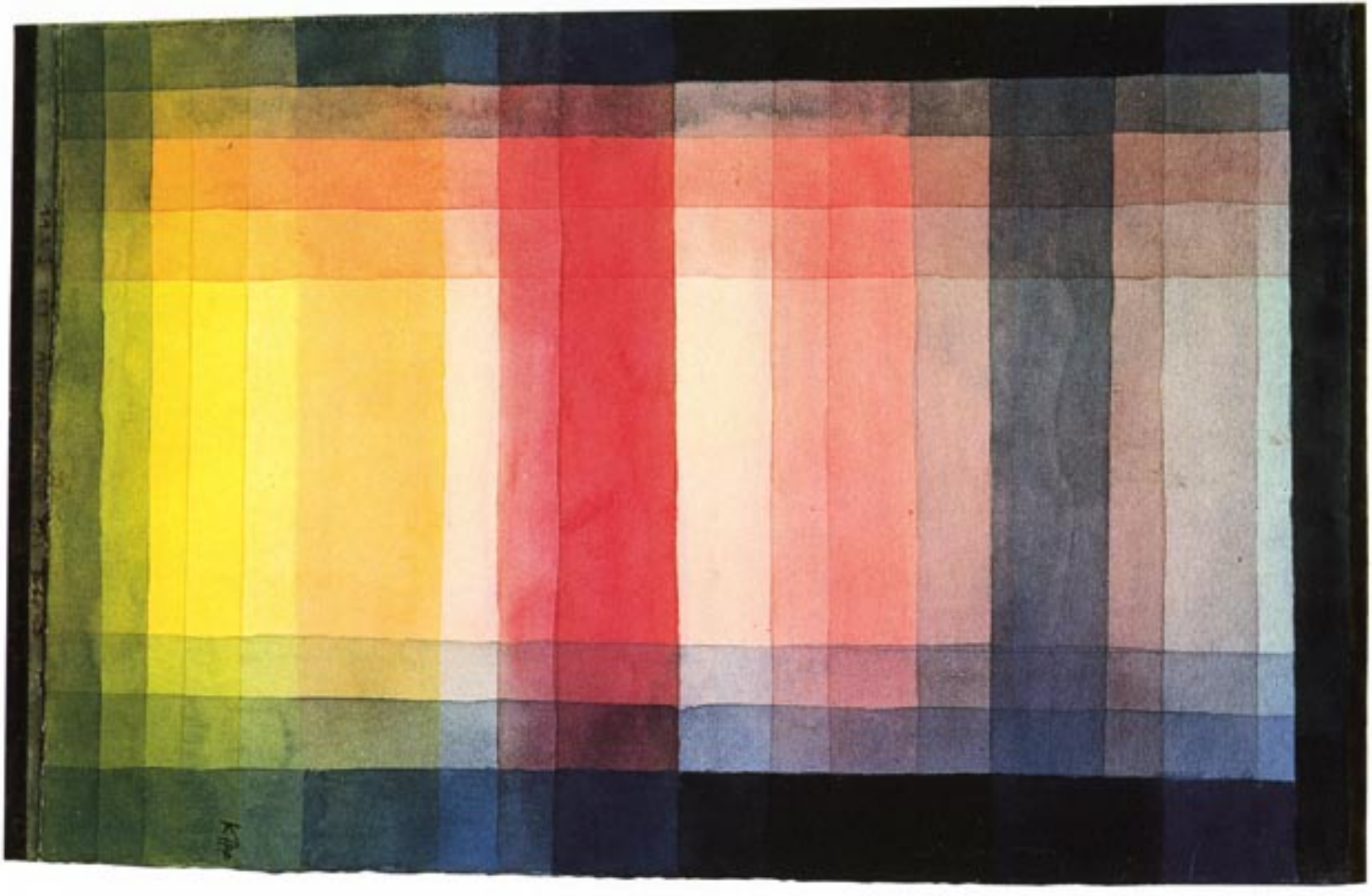
Piet Mondrian : Composition avec grille modulaire 9 (1919).
http://en.wikipedia.org/wiki/Piet_Mondrian



Josef Albers : Image-grillage (1922).
Assemblage de verre.
http://www.albersfoundation.org/

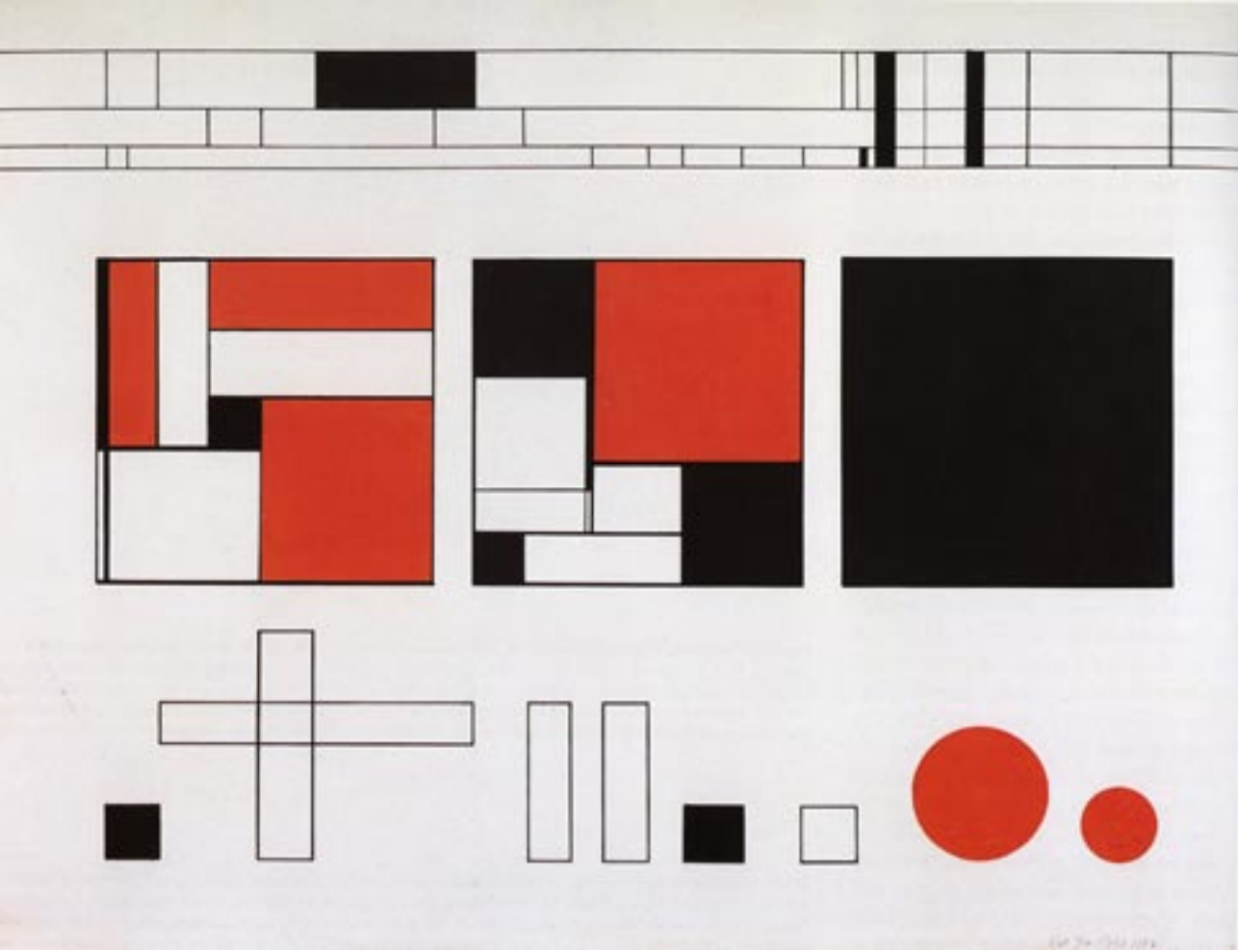


László Moholy-Nagy : K VII (1922).
http://www.moholy-nagy.org/



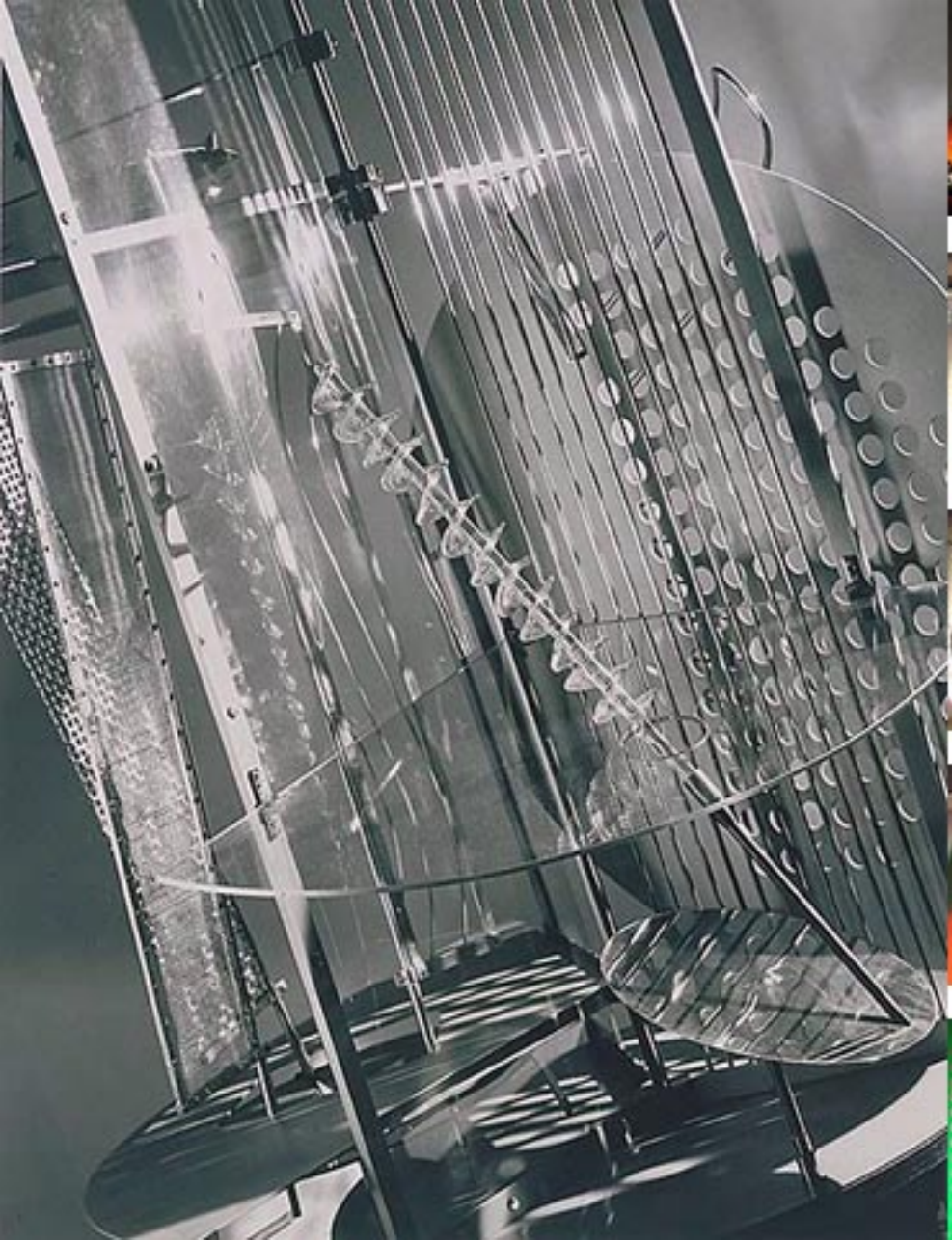
Paul Klee : Architecture de la plaine (1923).
http://en.wikipedia.org/wiki/Paul_Klee

Gunta Stölzl : Tapisserie n°539 (1926).
Tissu ajouré, soie artificielle, laine.
http://en.wikipedia.org/wiki/Paul_Klee

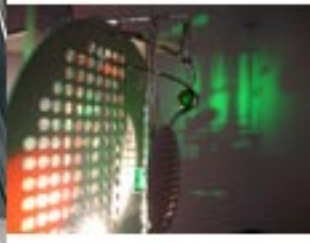


Karl-Peter Rölh : Composition abstraite (1926).

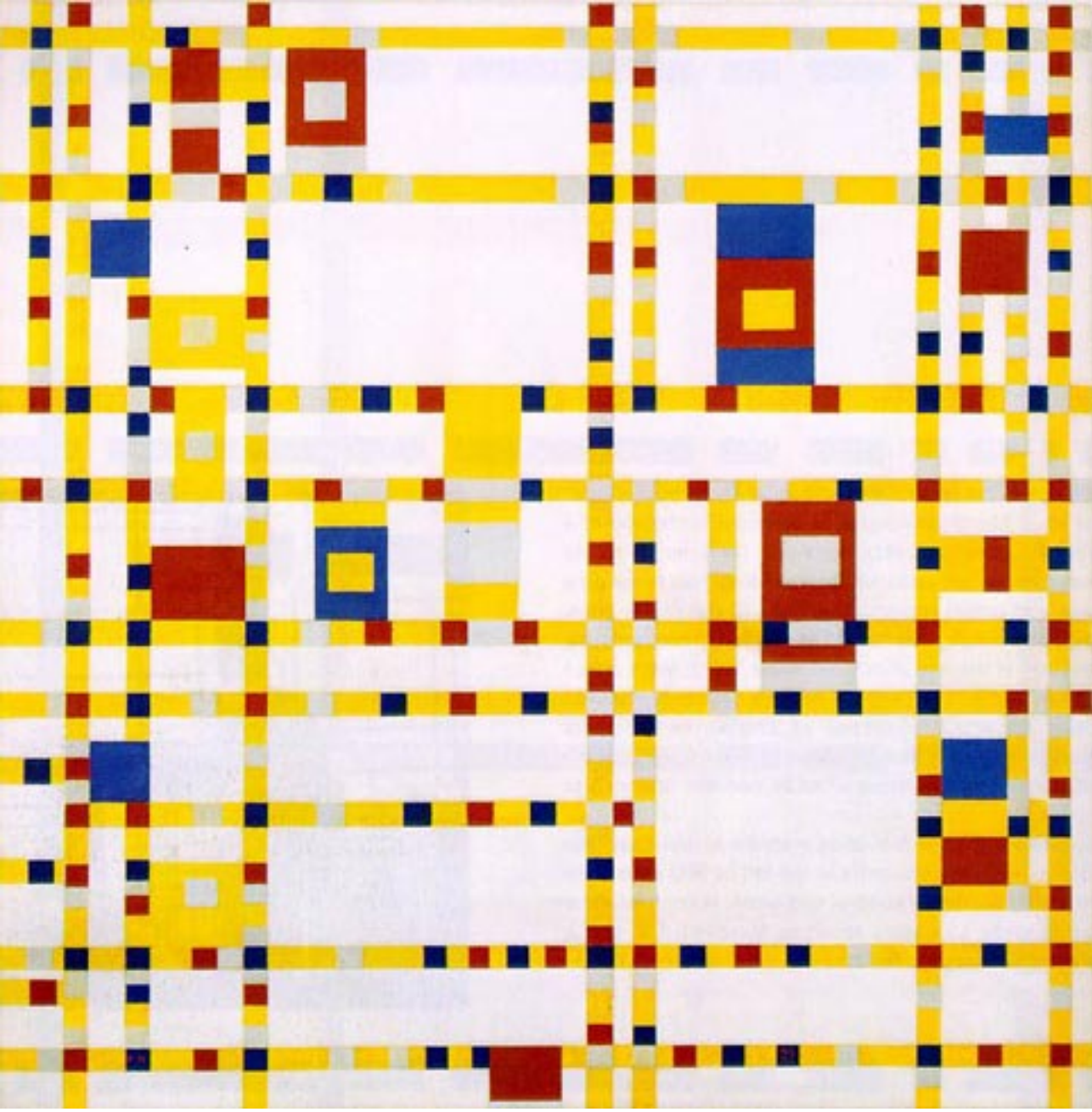
El Lissitzky : Running in the city (1926).
http://www.getty.edu/research/conducting_research/digitized_collections/lissitzky/








László Moholy-Nagy : Light-Space-Modulator (1930).
http://www.bauhaus.de/english/bauhaus1919/kunst/kunst_modulator.htm

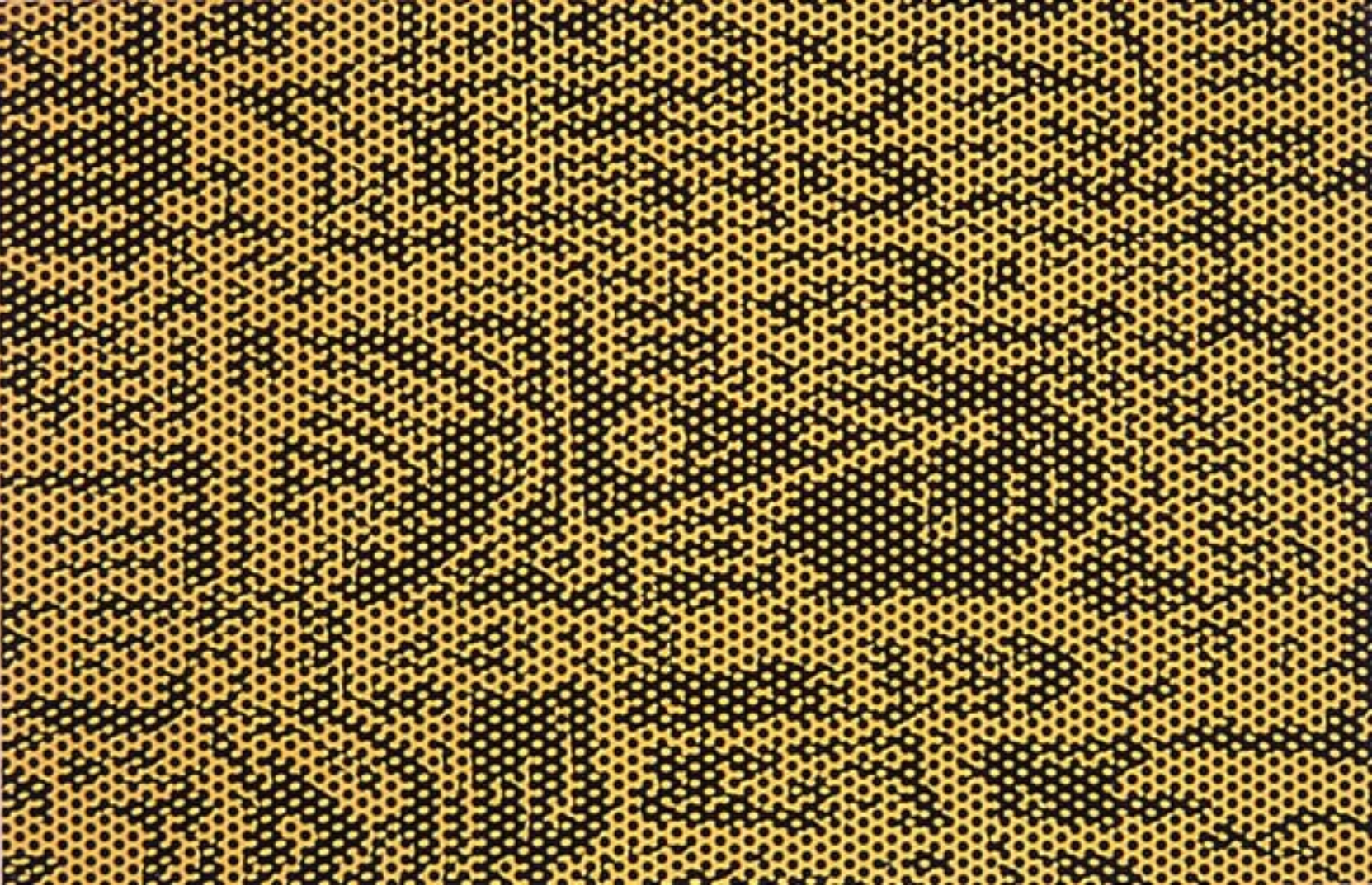


Piet Mondrian : Broadway Boogie Woogie (1942-1943).
http://www.artchive.com/artchive/M/mondrian/broadway.jpg.html

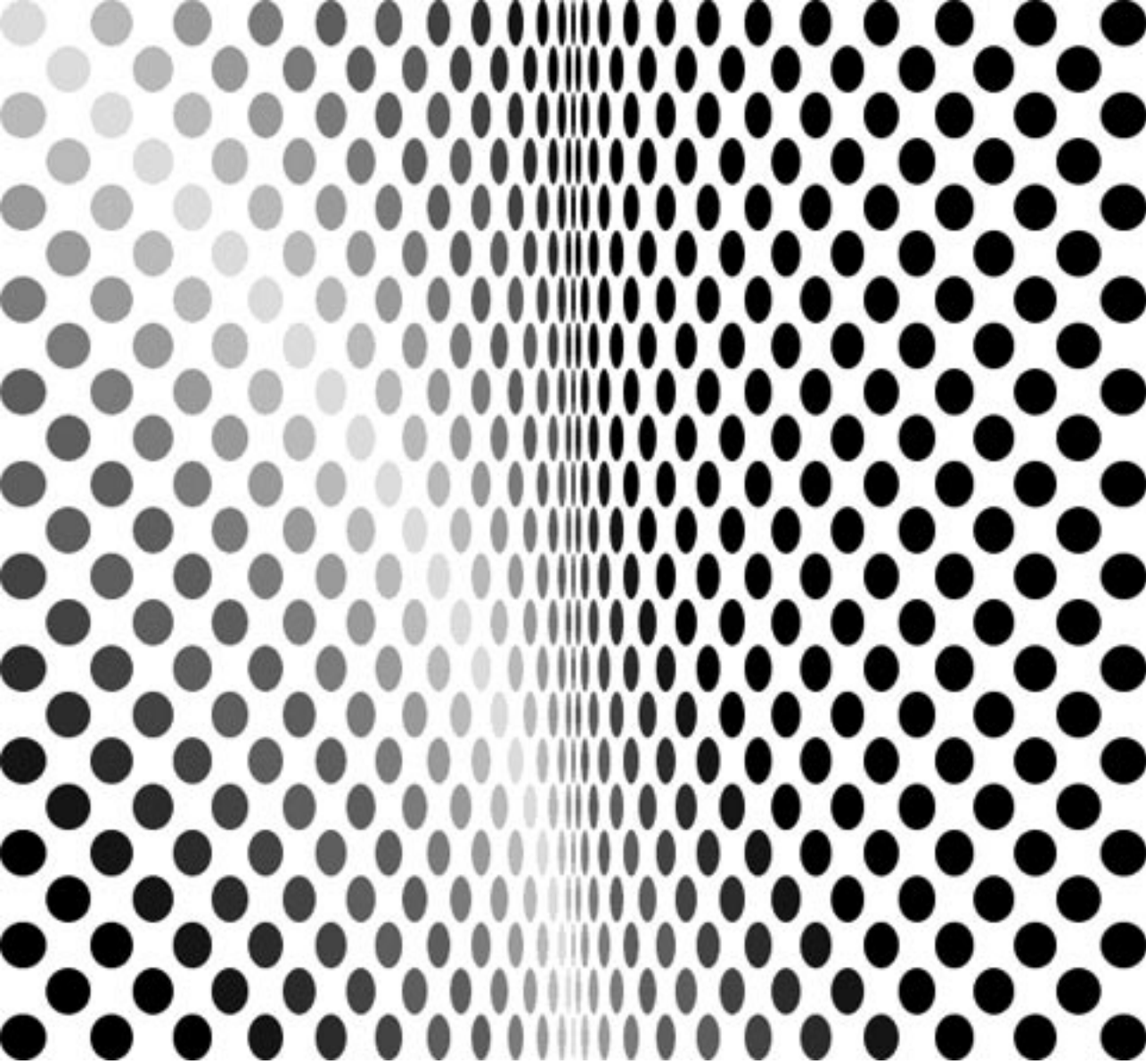
Jackson Pollock : Lavender Mist, Number 1 (1950).

Nicolas Schöffer : Salle d’exposition, Villa des arts (Paris),
oeuvres réalisées entre 1949 et 1974.
http://www.olats.org/schoffer/

Nicolas Schoffer : Sculpture cybernétique (c.1960)
Photo: Robert Doisneau.

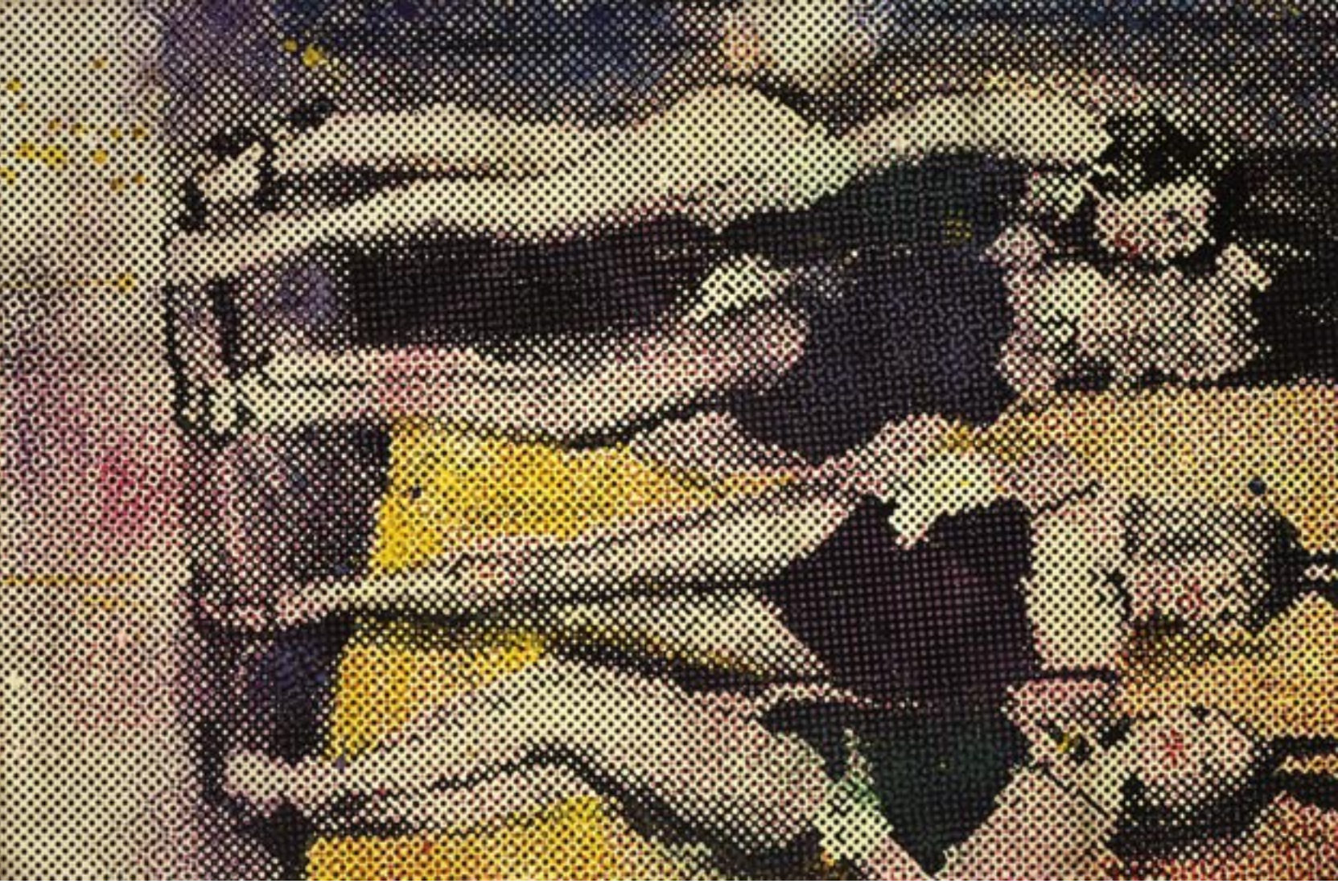


Roy Lichtenstein : Rouen Cathedral (1969).
http://www.lichtensteinfoundation.org/



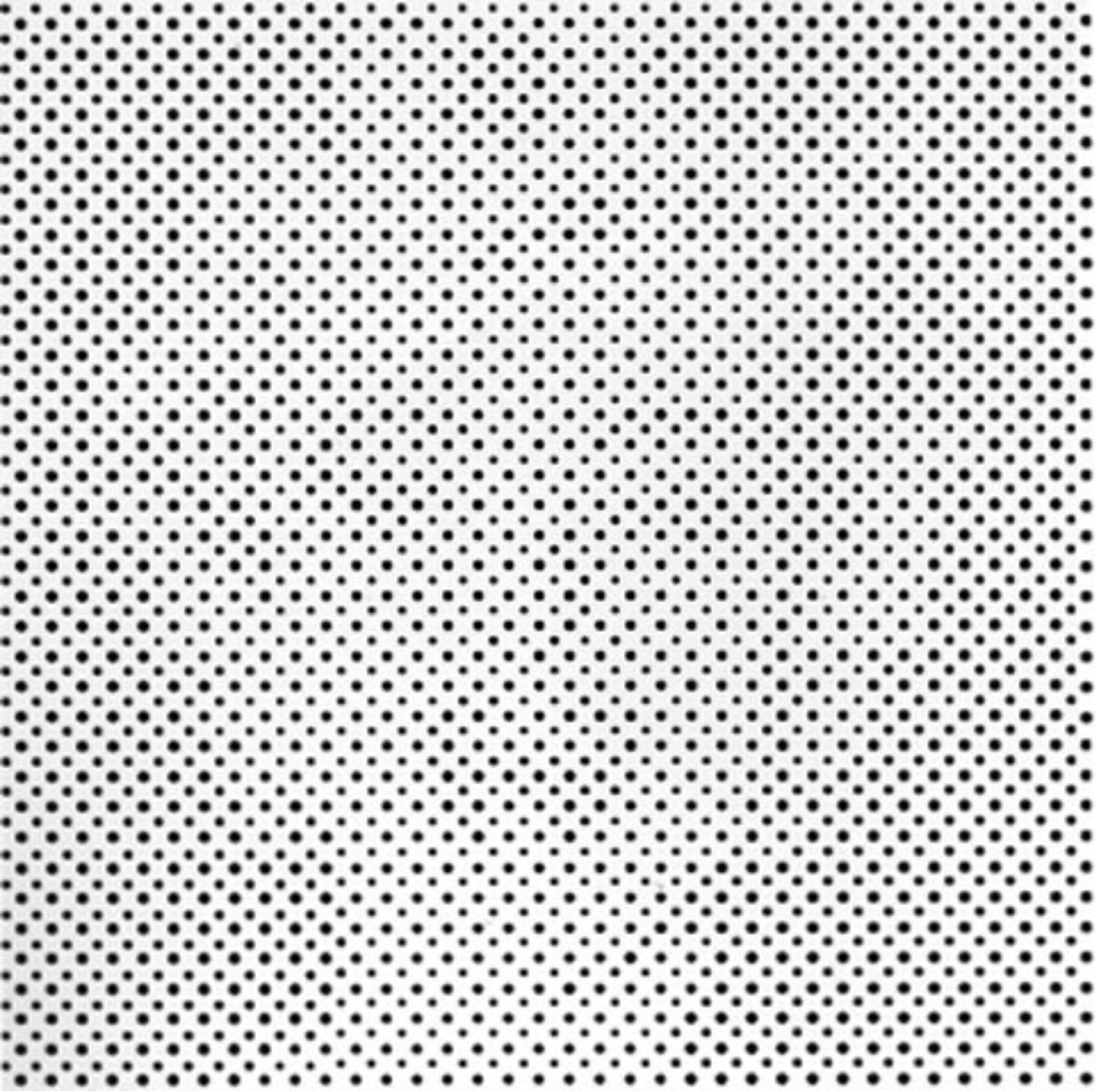
Bridget Riley : Loss (1964)
http://nadav.harel.org.il/Bridget_Riley/

Alain Jacquet : Le déjeuner sur l’herbe (1964).
http://www.centrepompidou.fr/

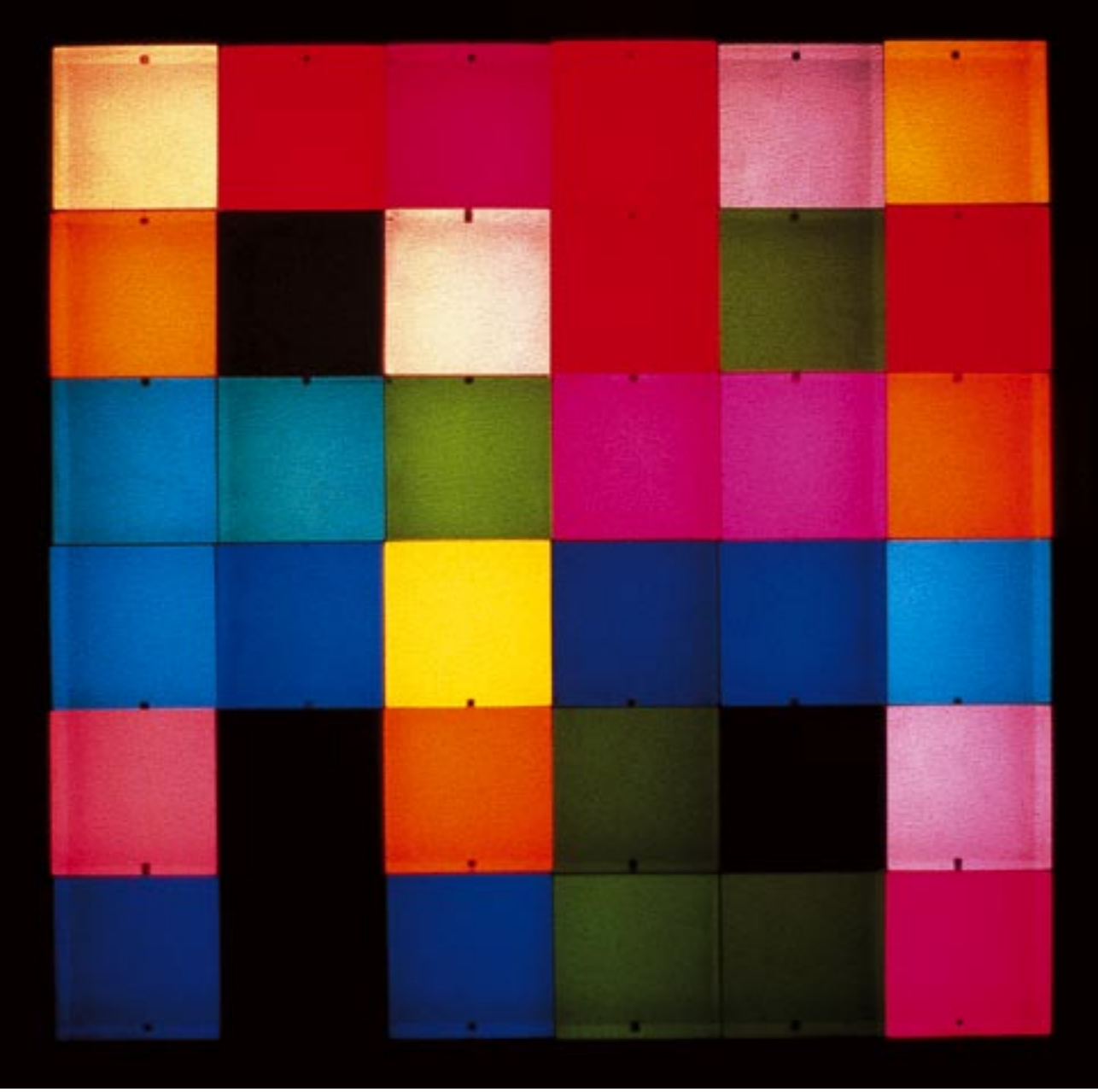


Sigmar Polke : bunnies (1966).
http://de.wikipedia.org/wiki/Sigmar_Polke

Richard Gregory : Dalmatien.
image inspirée par la Gestalt theory (c.1920).
http://en.wikipedia.org/wiki/Gestalt_psychology
http://fr.wikipedia.org/wiki/Psychologie_de_la_forme
http://www.richardgregory.org/

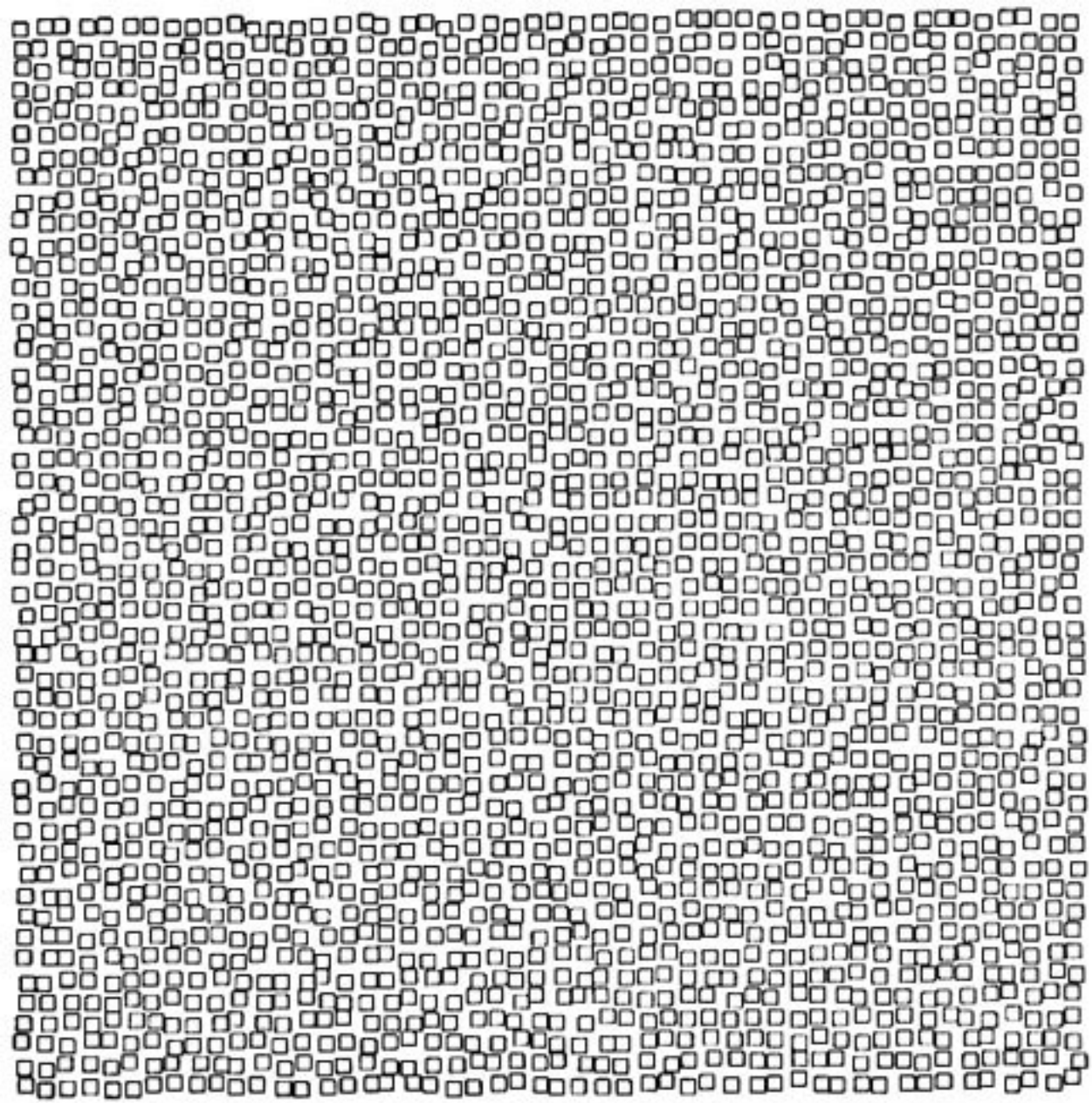


Gestalt theory (c.1920).
http://en.wikipedia.org/wiki/Gestalt_psychology
http://fr.wikipedia.org/wiki/Psychologie_de_la_forme

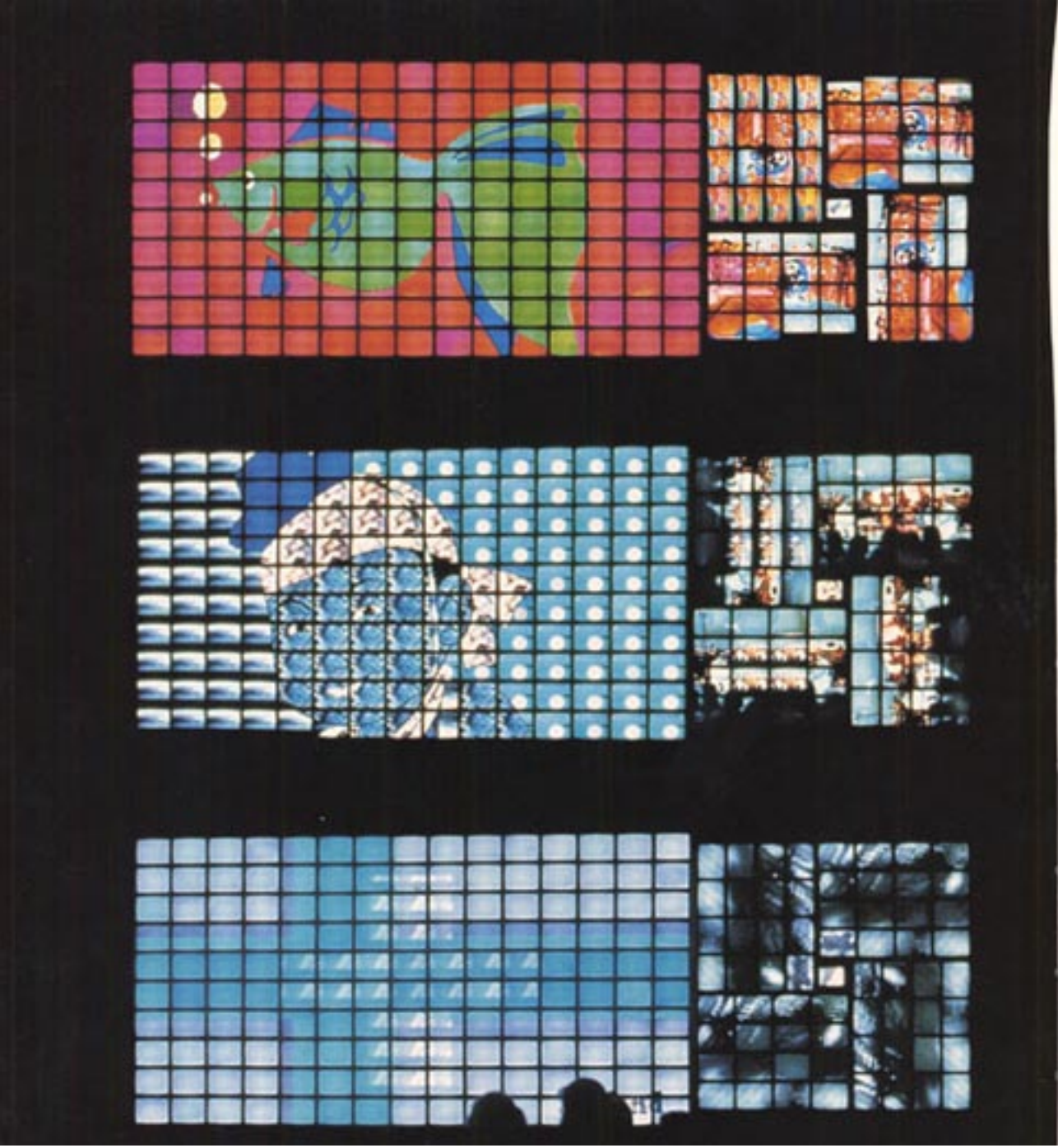


Davide Boriani : Pantachrome n°5 (1967-1976).
Boîte lumineuse, aluminium ,lampe, filtre, bois, moteur électrique.
http://www.davideboriani.com/

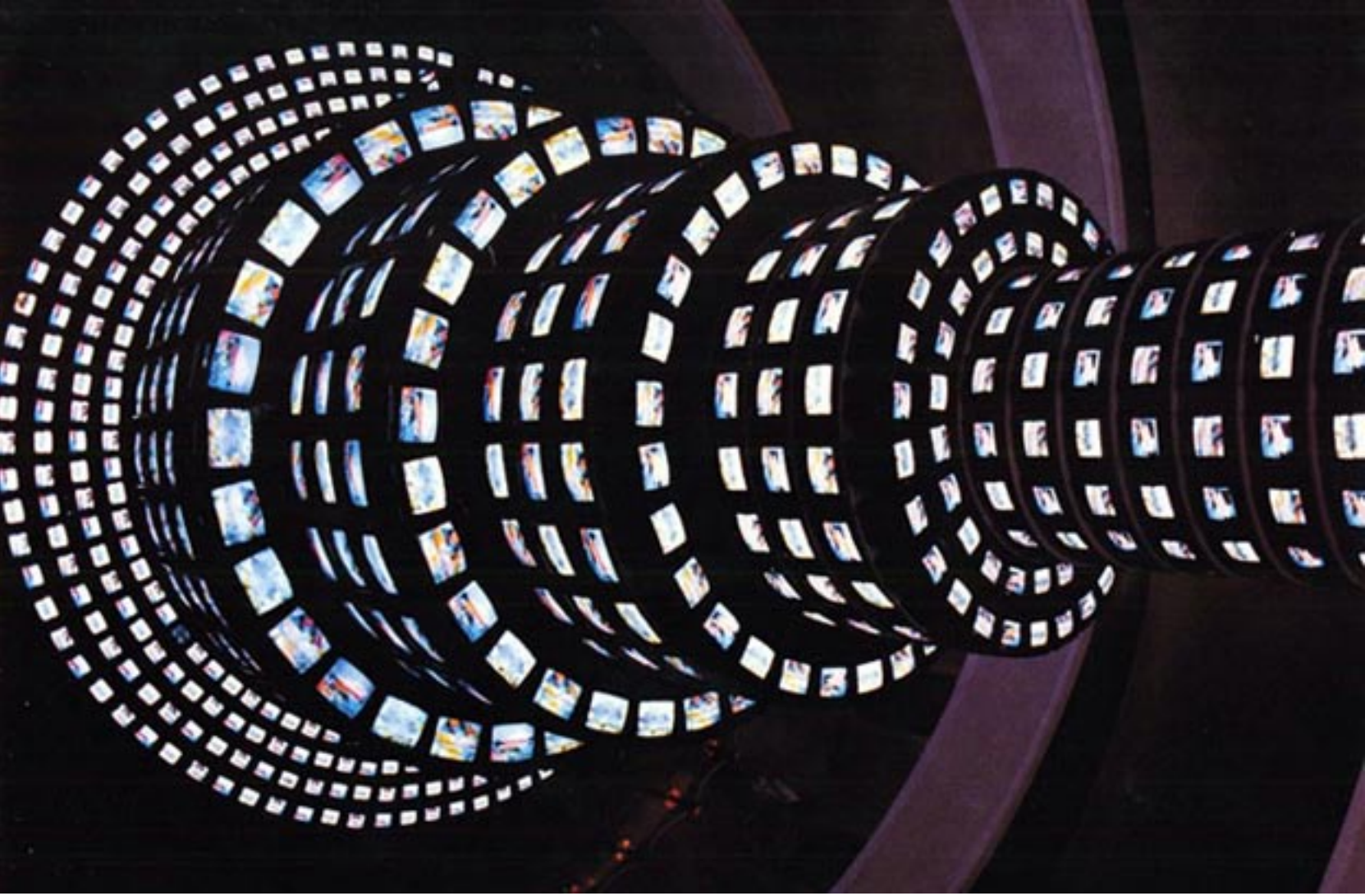
James Turrell : Ondoe Blue (1967).
http://www.kultureflash.net/archive/102/priview.html



Vera Molnar : 196 squares series (1975).



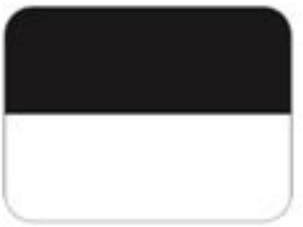
Nam June Paik : Megatron/Matrix (1995).
Eight channel computer driven video installation, 215 monitors.
http://www.paikstudios.com/



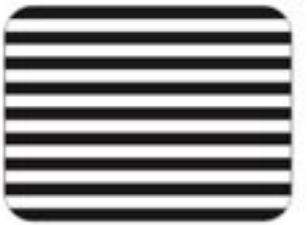
Nam June Paik : The More the Better, (1988).
Three channel video installation, 1.003 monitors, steel structure.
http://www.paikstudios.com/

David Rokeby : Very Nervous System (1986-1990).
video camera, image processor, computer, synthesizer, sound system.
http://homepage.mac.com/davidrokeby/home.html

impulswaveform 50 hz



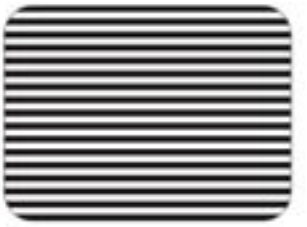
squarewaveform 50 hz



squarewaveform 400 hz

impulswaveform 100 hz

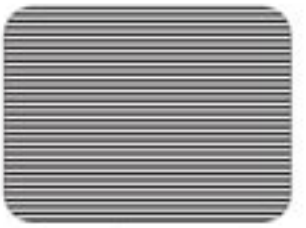
squarewaveform 100 hz



squarewaveform 800 hz

impulswaveform 150 hz

squarewaveform 150 hz



squarewaveform 1600 hz

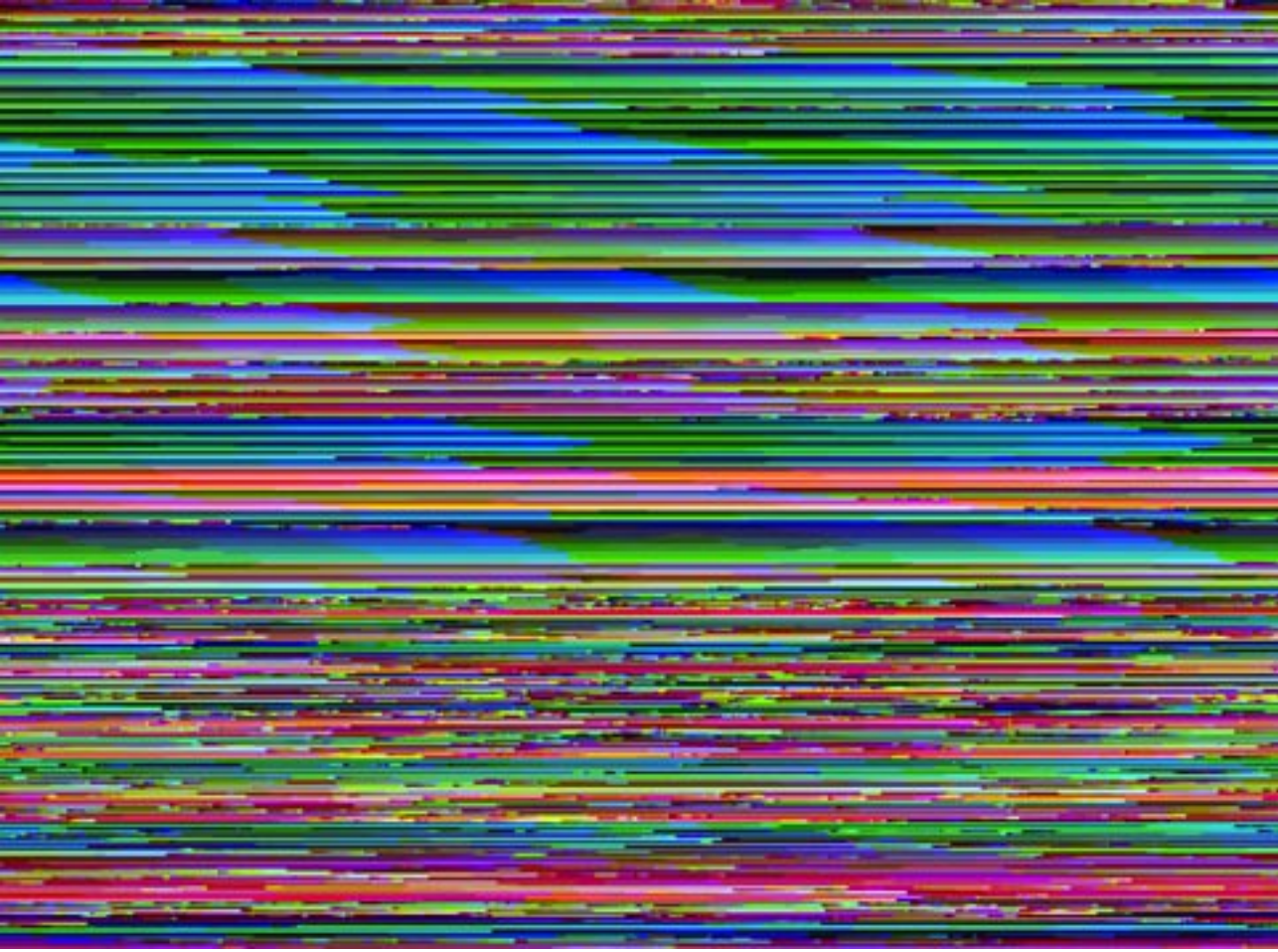
impulswaveform 400 hz

squarewaveform 200 hz

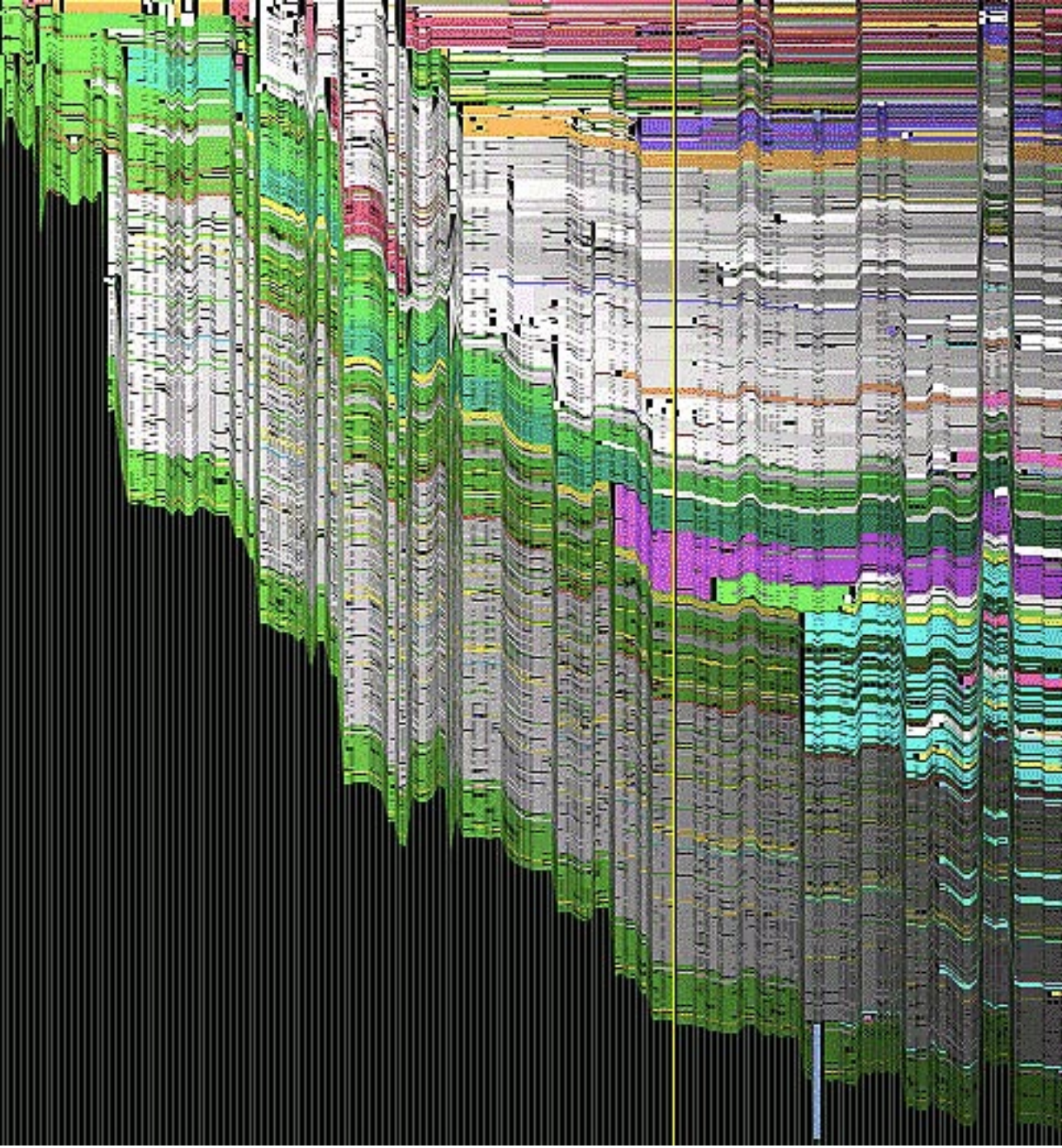
Carsten Nicolai : Telefunken (2000).
http://www.eigen-art.com/

Carsten Nicolai : Telefunken (2000).
http://www.eigen-art.com/

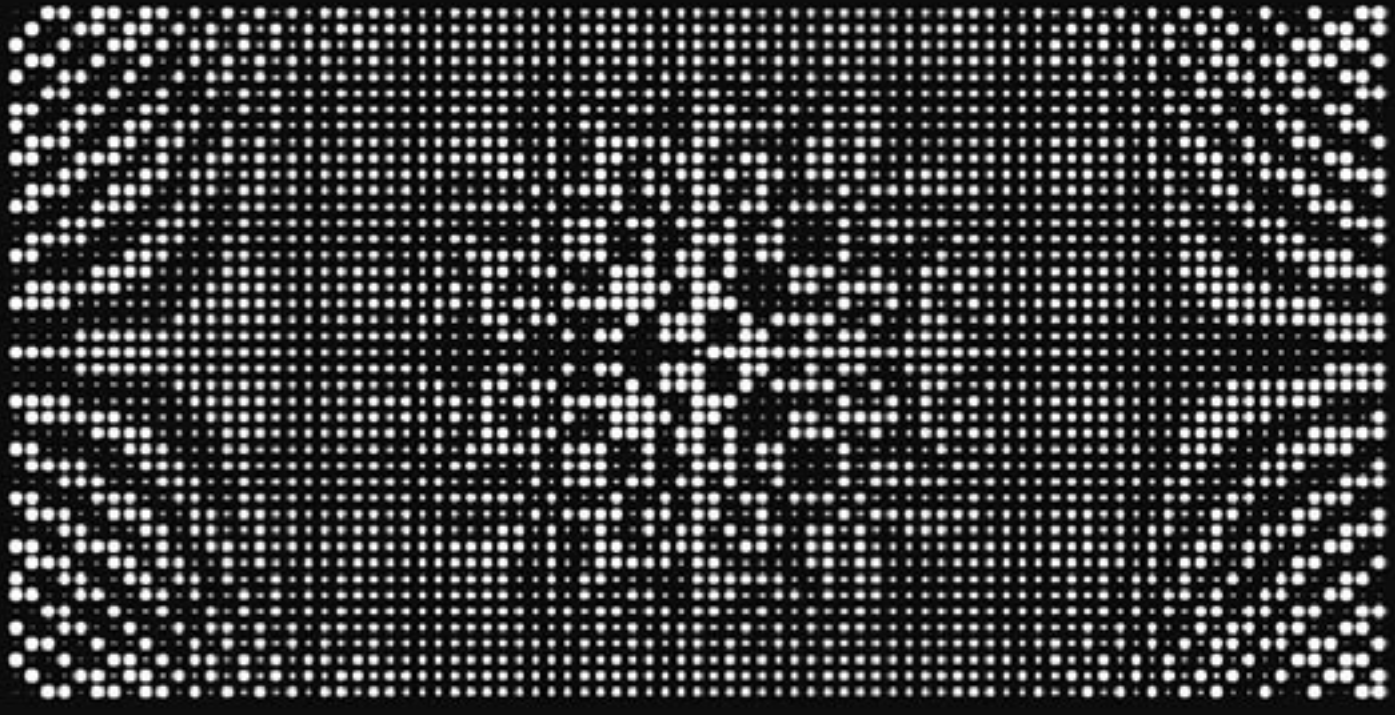
Limiteazero : Laptop Orchestra (2004),
installation, logiciel, interface, sonorisation.
http://www.limiteazero.com/l_o/index.html



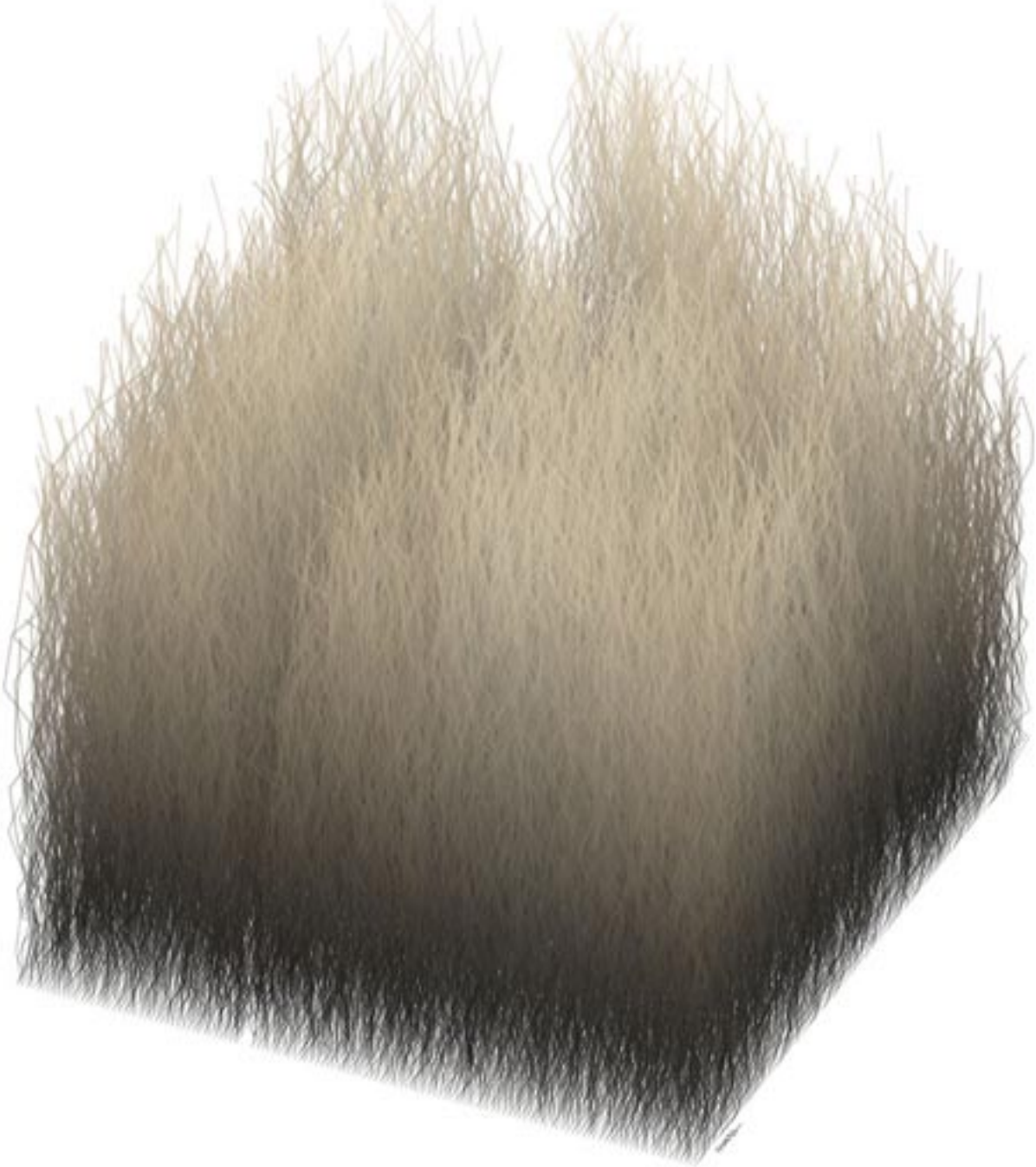
Lisa Jevbratt :
Database Imaginary (2004),
projet web et impression.
http://jevbratt.com/



Martin Wattenberg : History Flow (2003).
http://www.bewitched.com/

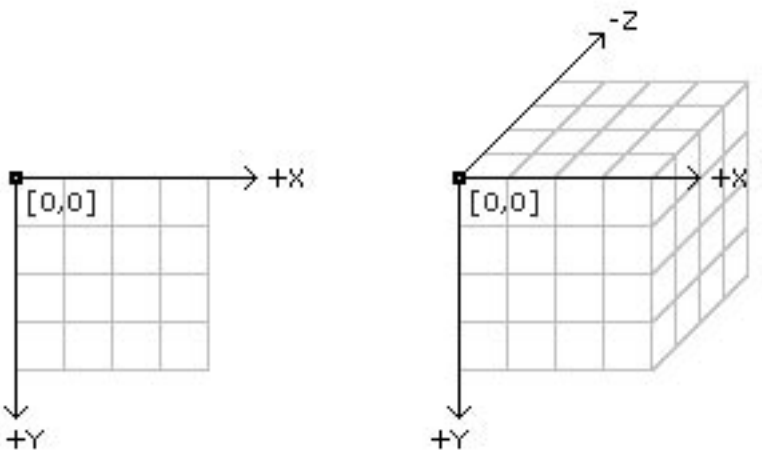


John Maeda : Dot Rectangles Series.
http://www.maedastudio.com/



John Maeda :
illustration inspired by Issey Miyake’s APOC (A-Piece-Of-Clothing)(2004).
http://www.maedastudio.com/

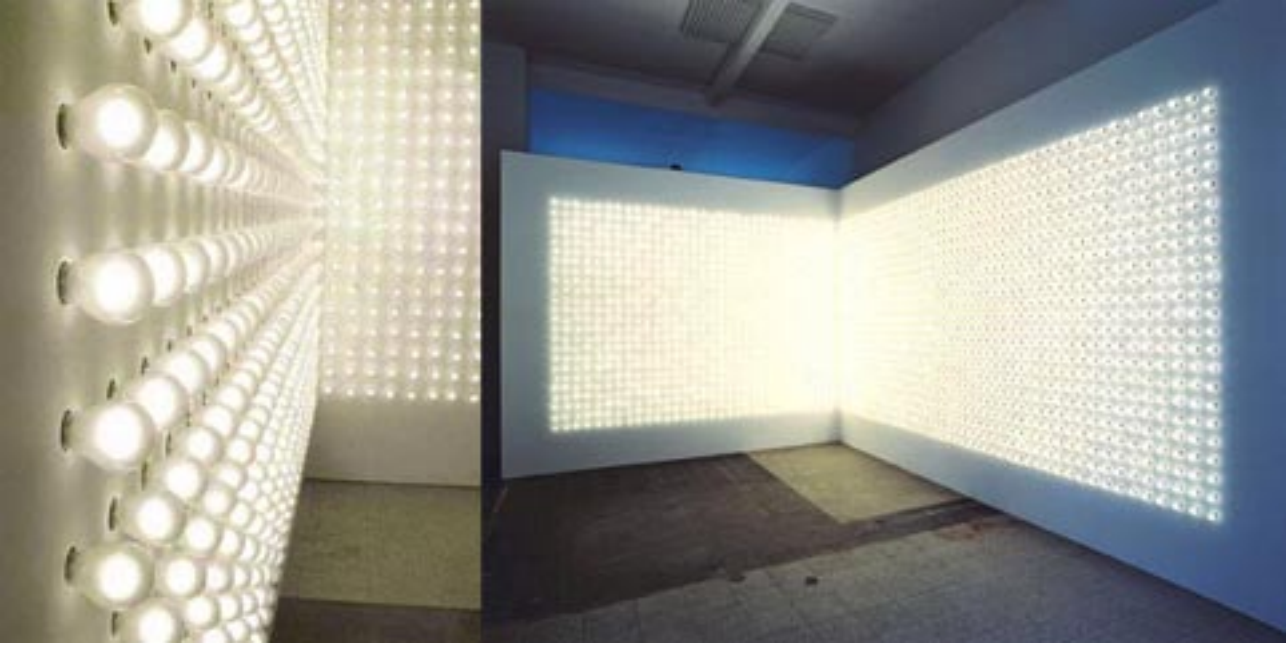
Eva Hesse : Accession (1967).



Coordonnées d’un point.

Jan Robert Leegte : Scrollbar (2005).
http://www.leegte.org/

Erwin Redl : Matrix I (2000/2002)
LED Installation.
http://www.paramedia.net/



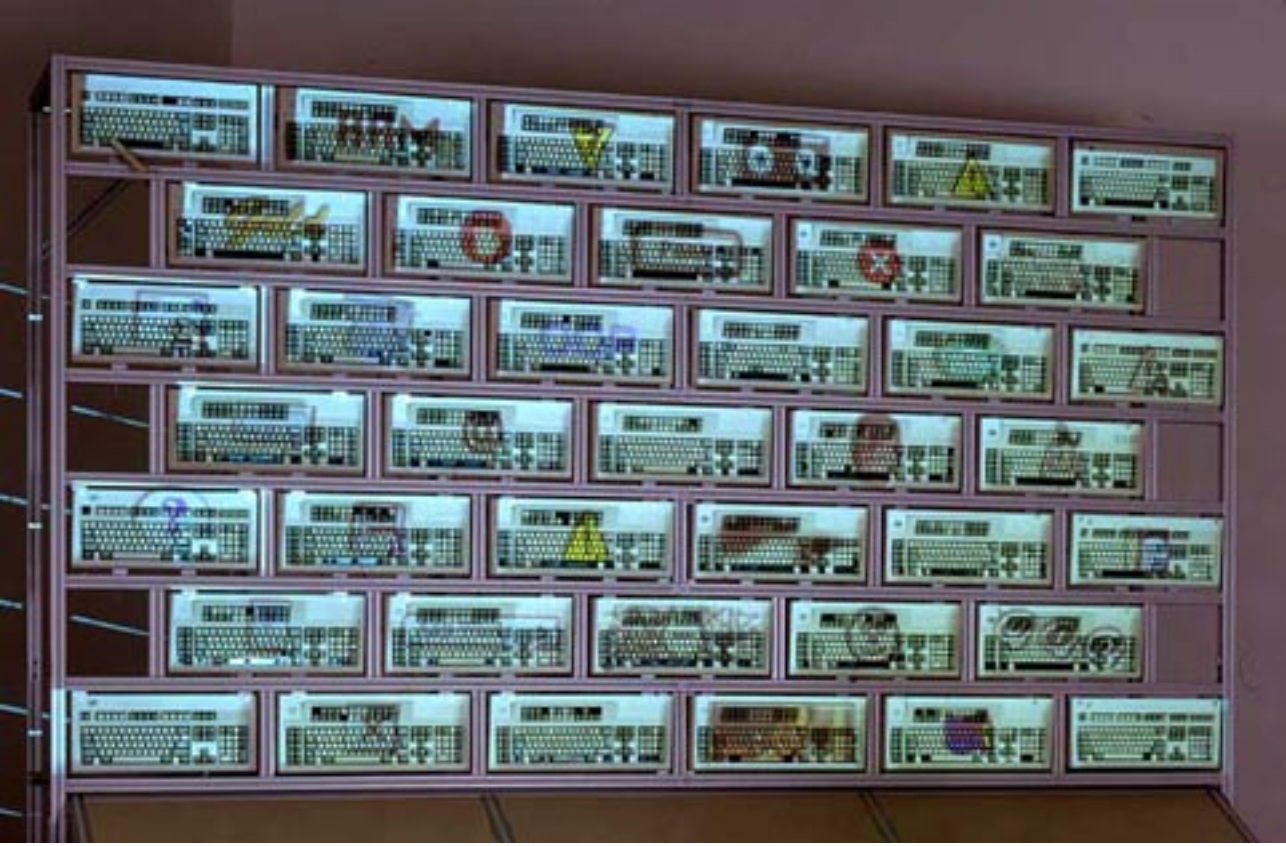
Carsten Höller : Light Corner (2001).
http://www.airdeparis.com/

Cornelia Parker : Edge of England (1999),
Chalk retrieved from a cliff fall at Beatch Head, South Downs, England.
http://www.tate.org.uk/colddarkmatter/

Daniel Rozin : Shiny Balls Mirror (2003),
921 hexagonal black-anodized aluminum tube extrusion, 921 chrome-plated plastic balls, 819 motors, control electronics, video camera, computer.
http://smoothware.com/danny/

Daniel Rozin : Wooden Mirror (1999),
830 square pieces of wood, 830 servo motors, control electronics,
video camera, computer, wood frame.
http://smoothware.com/danny/

Olafur Eliasson : Quasi brick wall (2003).
http://www.olafureliasson.net/

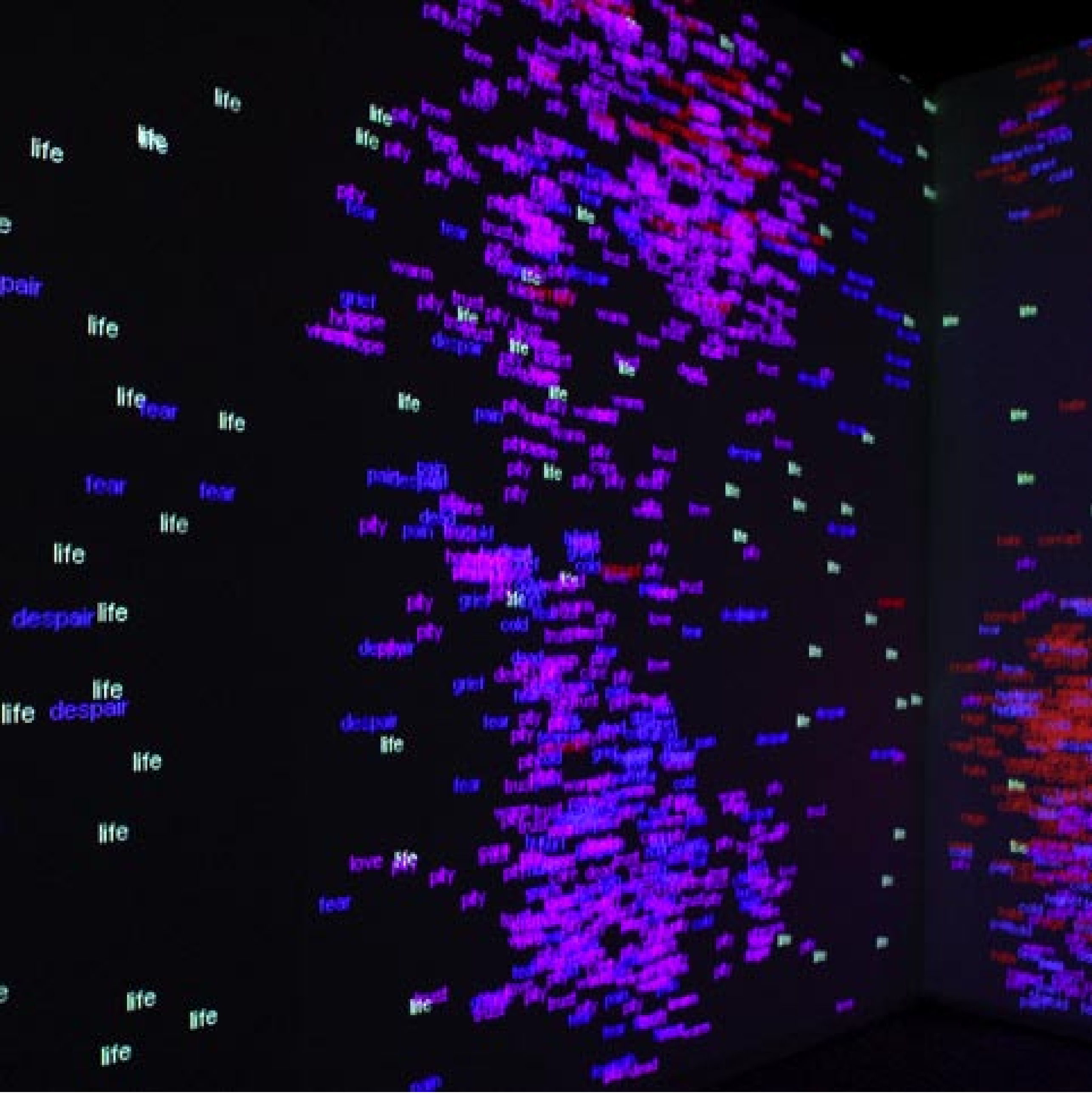


Perry Hoberman : Cathartic User Interface (1995-2000),
installation interactive, PC obsolètes, projection.
http://www.perryhoberman.com/

Carsten Höller : Phi wall (2002).
http://www.airdeparis.com/

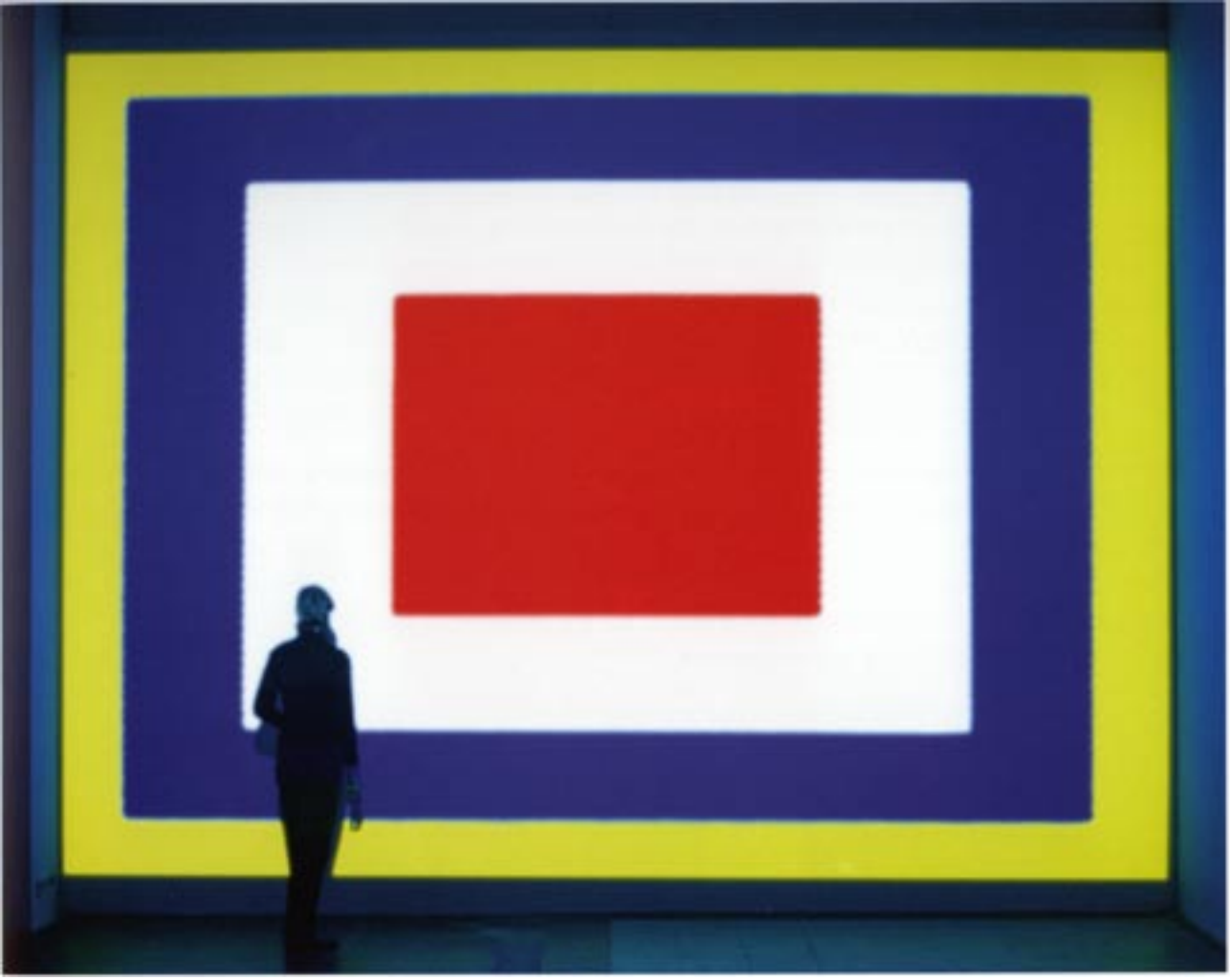
Frédéric Eyl & Gunnar Green : Aperture (2004).
http://www.fredericeyl.de/aperture/

David Small : Talmud Project (2000).
http://www.davidsmall.com/

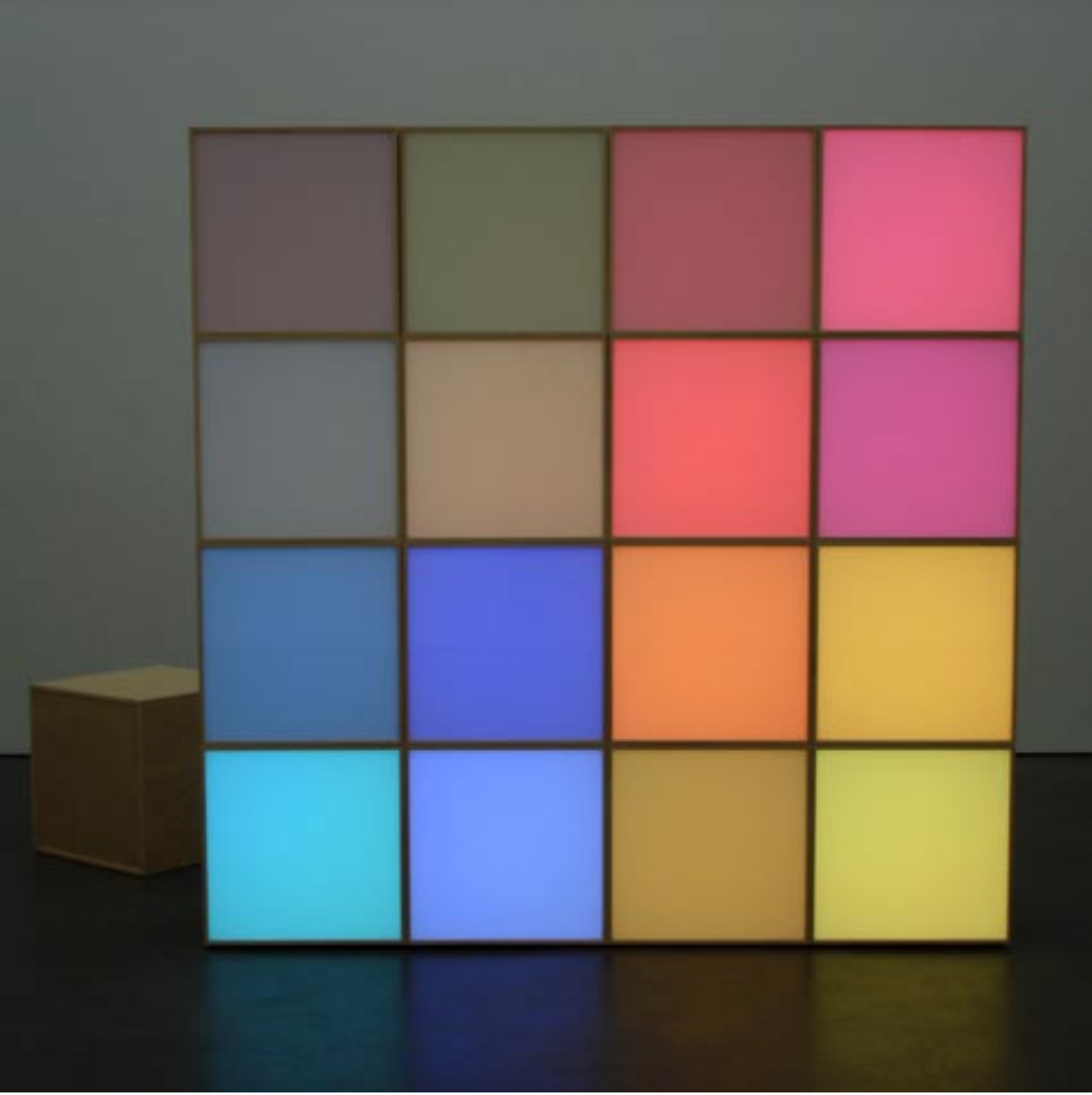


Charles Sandison : Rage love & despair (2003).
http://www.charlessandison.com/

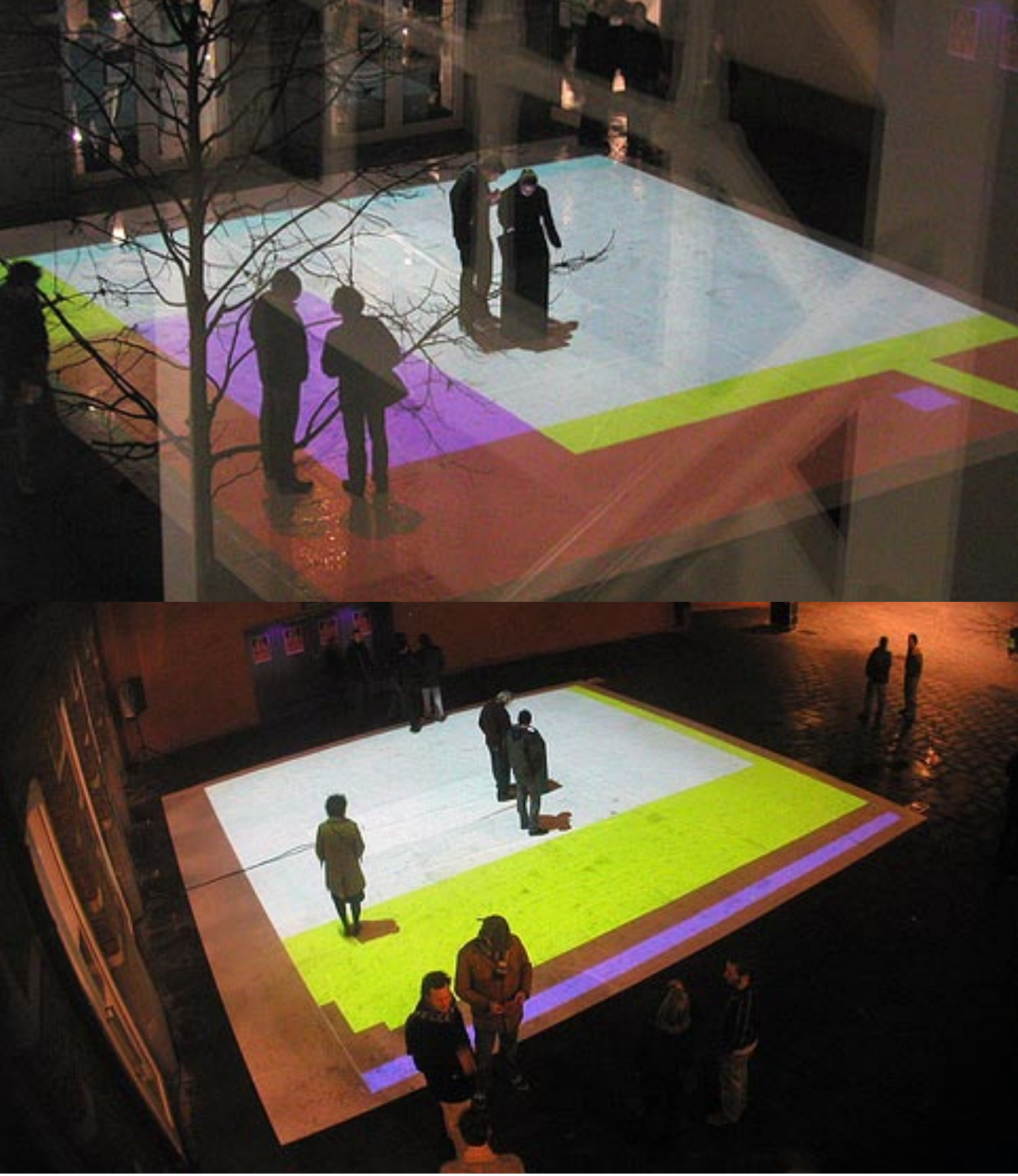
Peter Luining : Touch screen installation (2004).
http://www.ctrlaltdel.org/



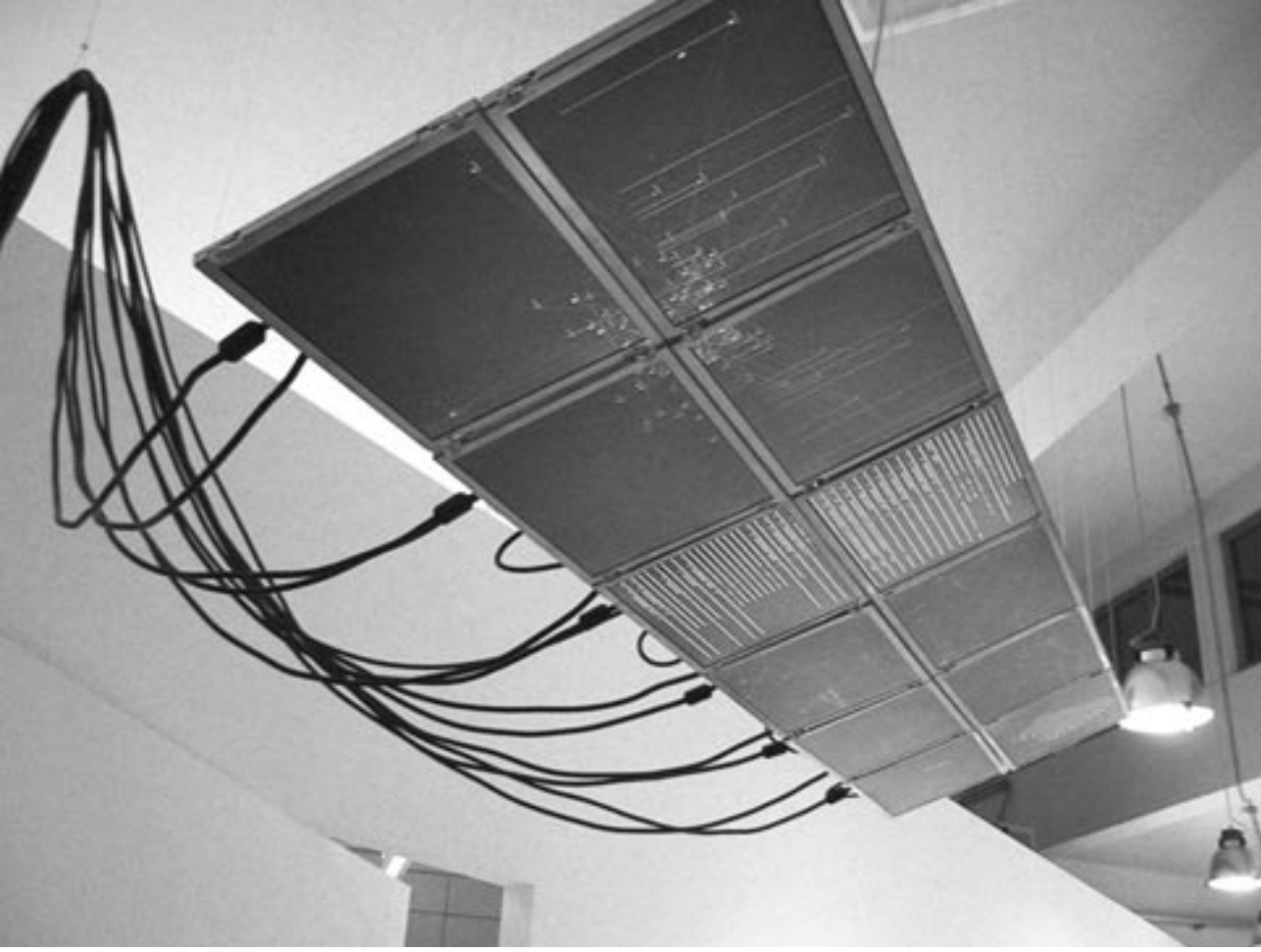
Ann Veronica Janssens : Scrub (2002).
http://www.gms.be/



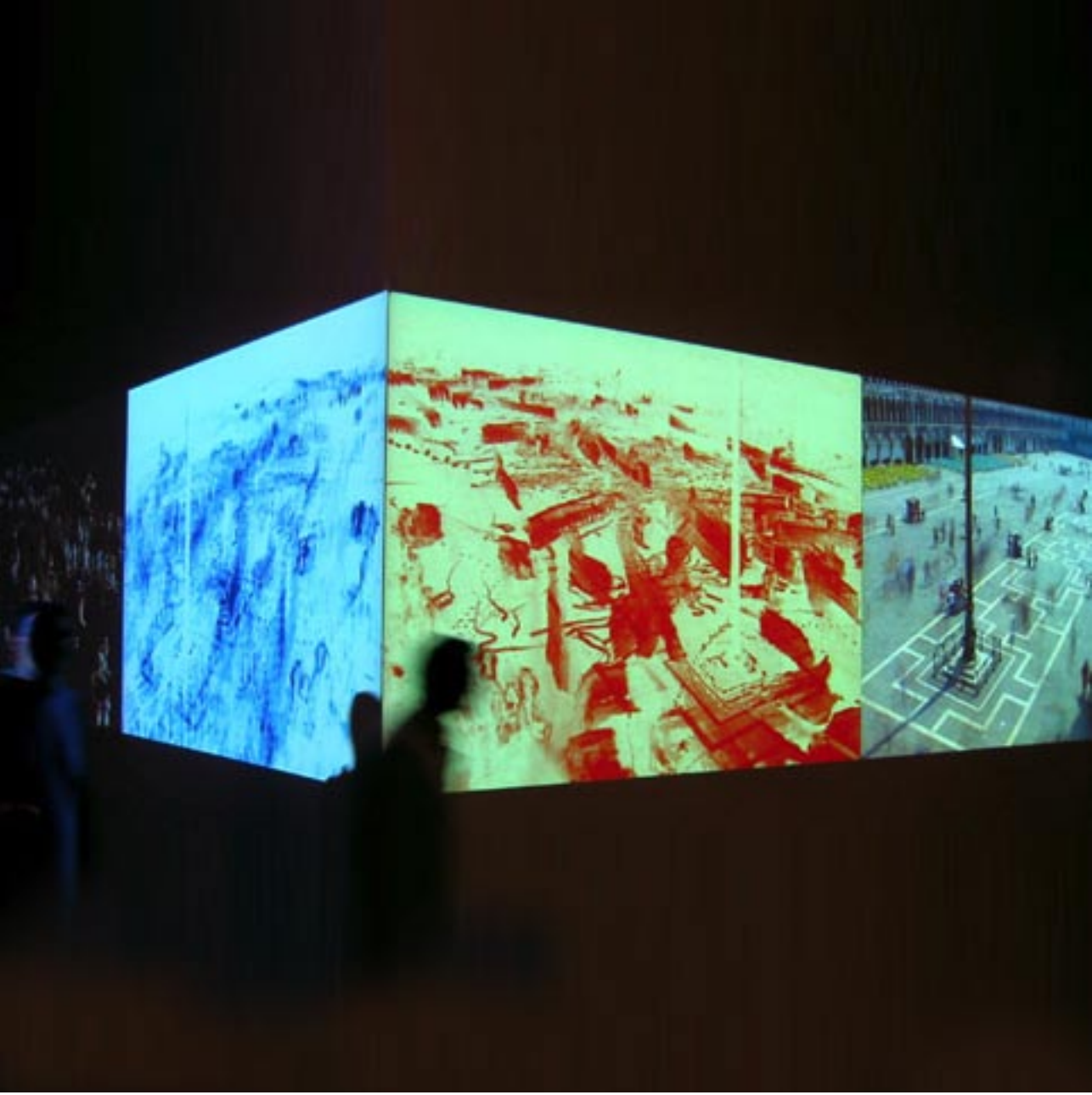
Angela Bulloch : Pixel Boxes (2000).
http://www.gms.be/



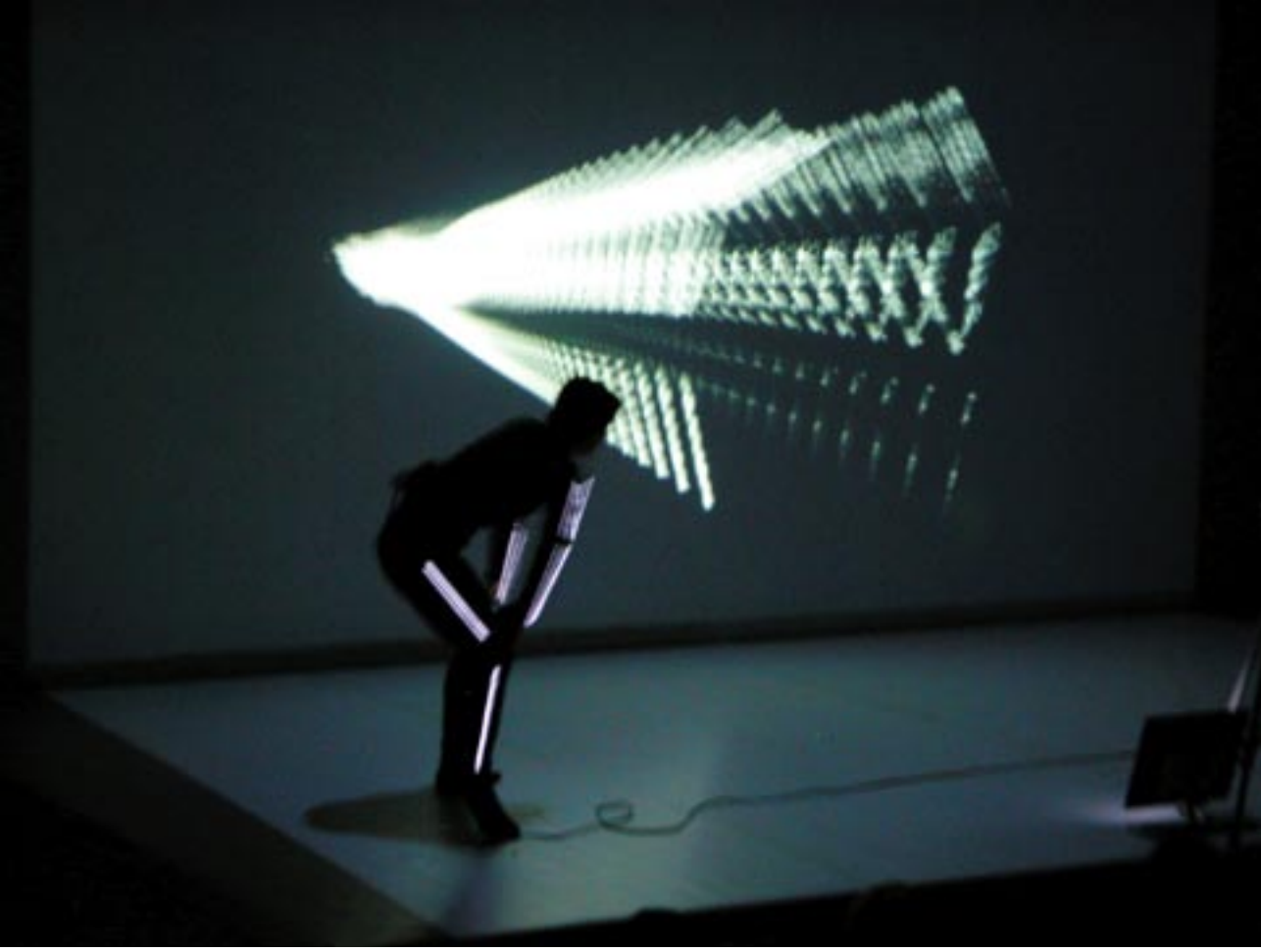
LAB[au] : Point, Line, Surface computed in seconds (2005),
clavier tactile, projection au sol.
http://www.lab-au.com/



Limiteazero : Network is speaking (2004),
installation, logiciel «Carnivore Client», réseau,
visualisation et sonorisation.
http://www.limiteazero.com/xyz/network.html



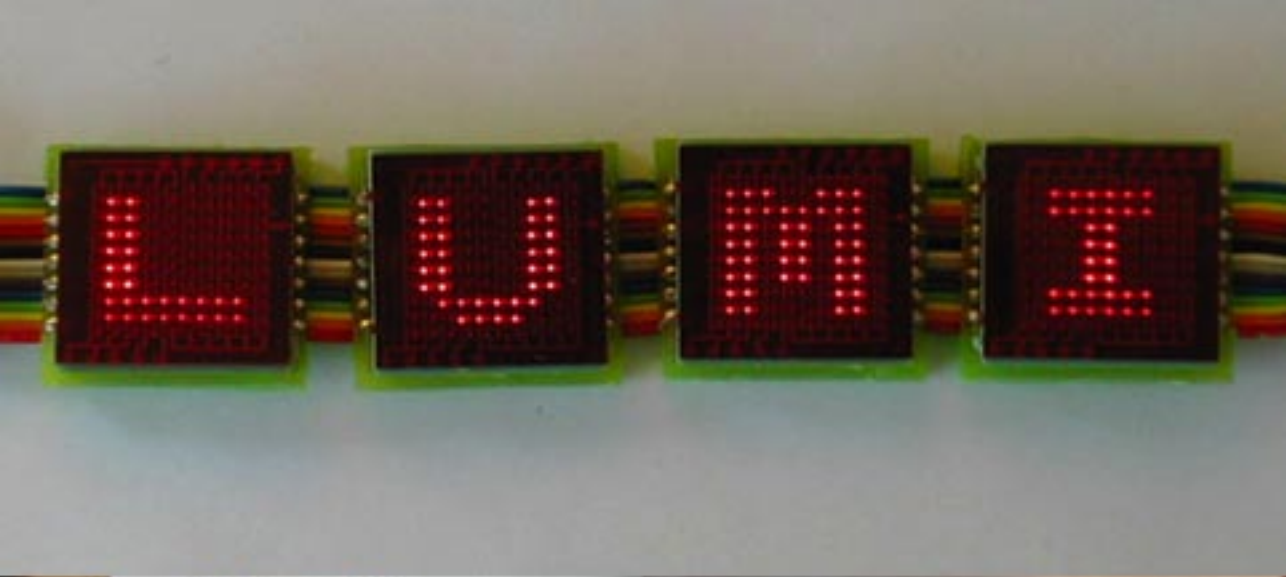
David Rokeby : Seen (2002),
captation video, software, projection.
http://homepage.mac.com/davidrokeby/seen.html
http://homepage.mac.com/davidrokeby/watch.html

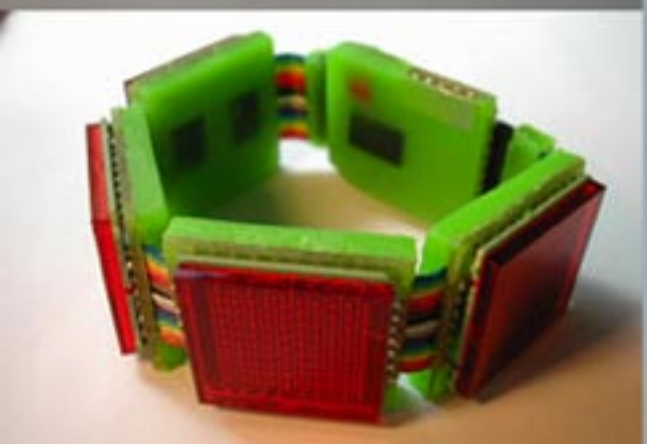


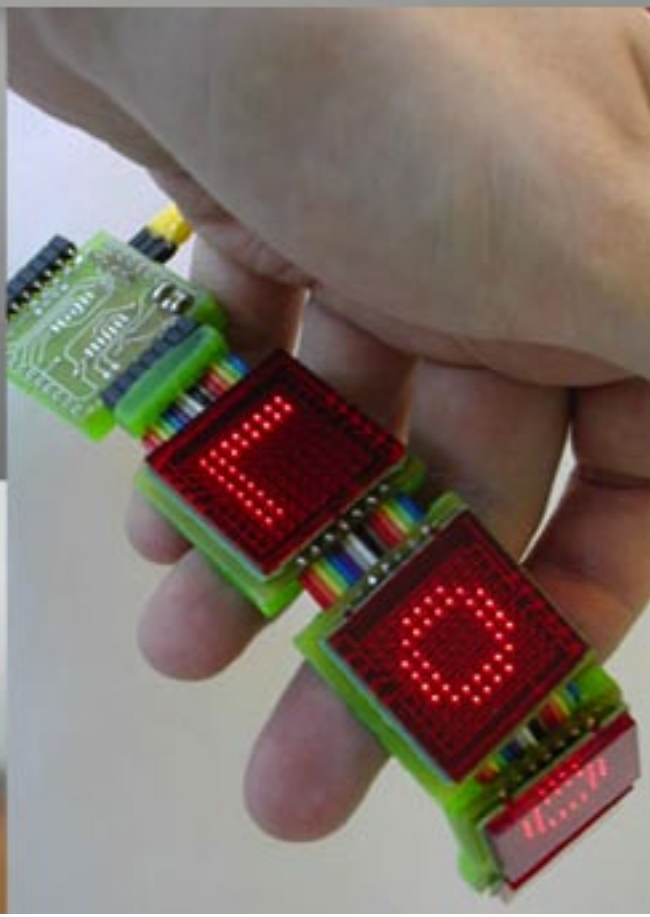
LAB[au] : Man in eSPACE.mov, installation-performance (2004/2006), captation video, software, projection multi-écrans.
http://www.lab-au.com/
http://www.mast-r.org/

Carsten Nicolai : Alva Noto Tour (2006).
http://www.eigen-art.com/

Carsten Nicolai : Milch (2000-2005).
http://www.eigen-art.com/





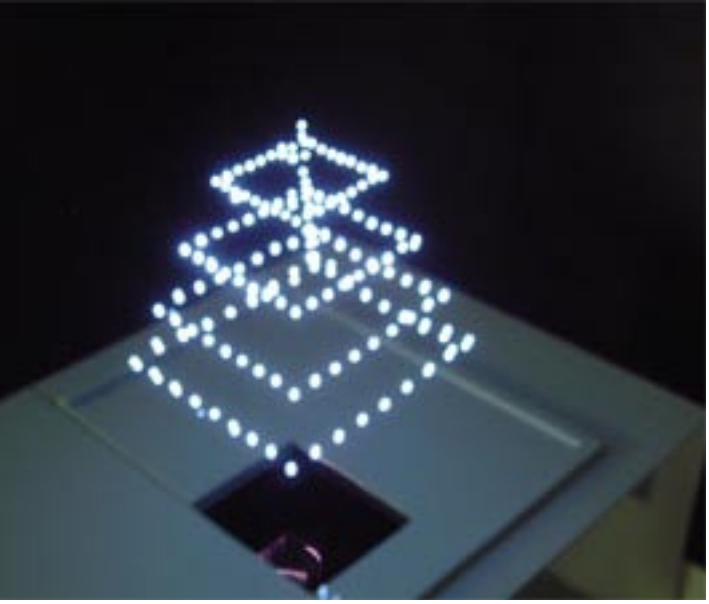


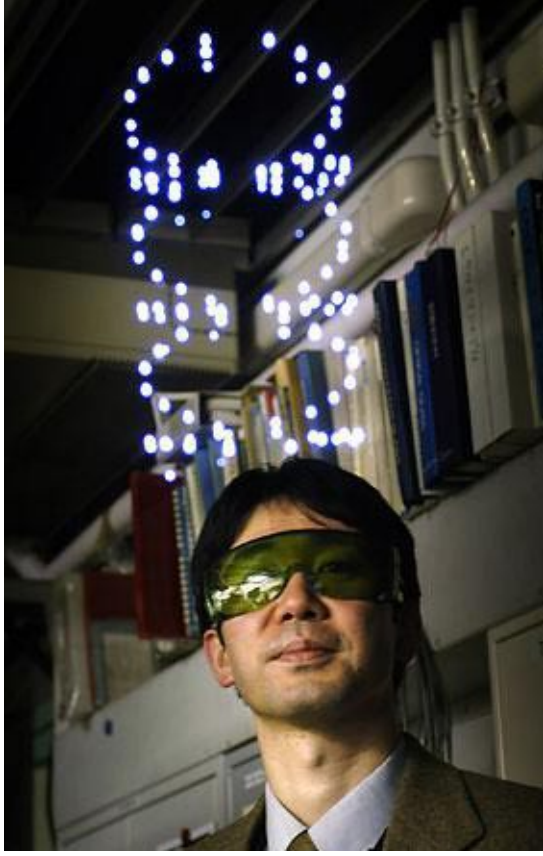
Elise Co : Lumiloop (système modulaire de rpogramme et de panneaux d'affichage emboîtables sous forme de bracelet réactif).
http://www.mintymonkey.com/

Philips Research : systèmes interactifs d’émission lumineuse appliqués aux textiles.
http://www.research.philips.com/

Cameron McNall & Damon Seeley : Target Interactive Breezeway (2005), installation interactive, détection de mouvement, murs intelligents.
http://electroland.net/

Cameron McNall & Damon Seeley : Interactive walkways (2005), installation interactive, détection de mouvement, sol intelligent.
http://electroland.net/





AIST : 3D-object displayed using a 3D-image spatial drawing with laser plasma.
http://www.aist.go.jp/

Realities United : Bix (2003).
Permanent light and media installation for the Kunsthaus Graz, Austria.
http://www.bix.at/
http://www.realities-united.de/

Realities United : Bix (2003).
Permanent light and media installation for the Kunsthaus Graz, Austria.
http://www.bix.at/
http://www.realities-united.de/

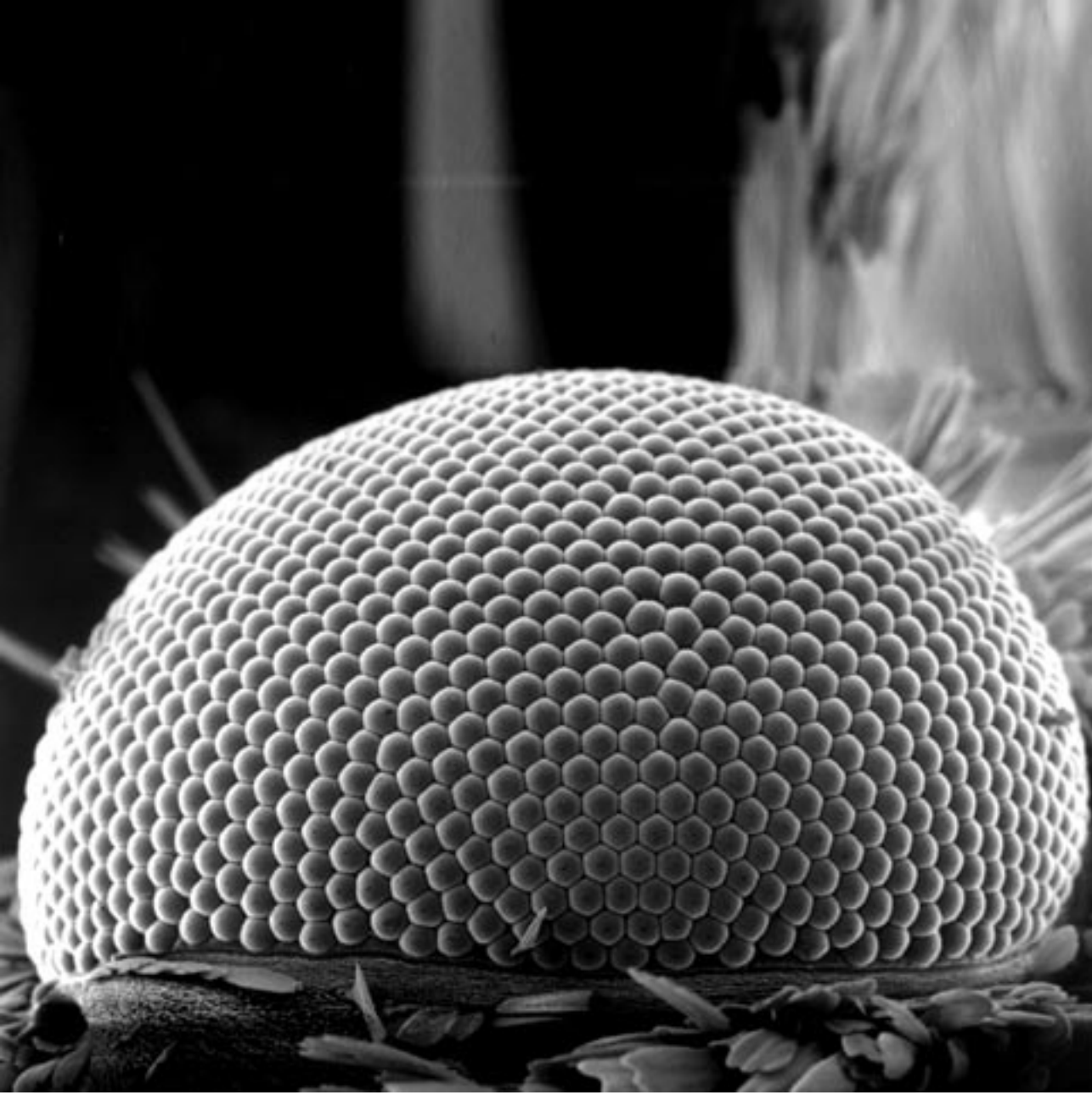
Realities United (Tim Edler and Jan Edler) : Spots (2006).
http://www.spots-berlin.de/
http://www.realities-united.de/

Realities United (Tim Edler and Jan Edler) : Spots (2006).

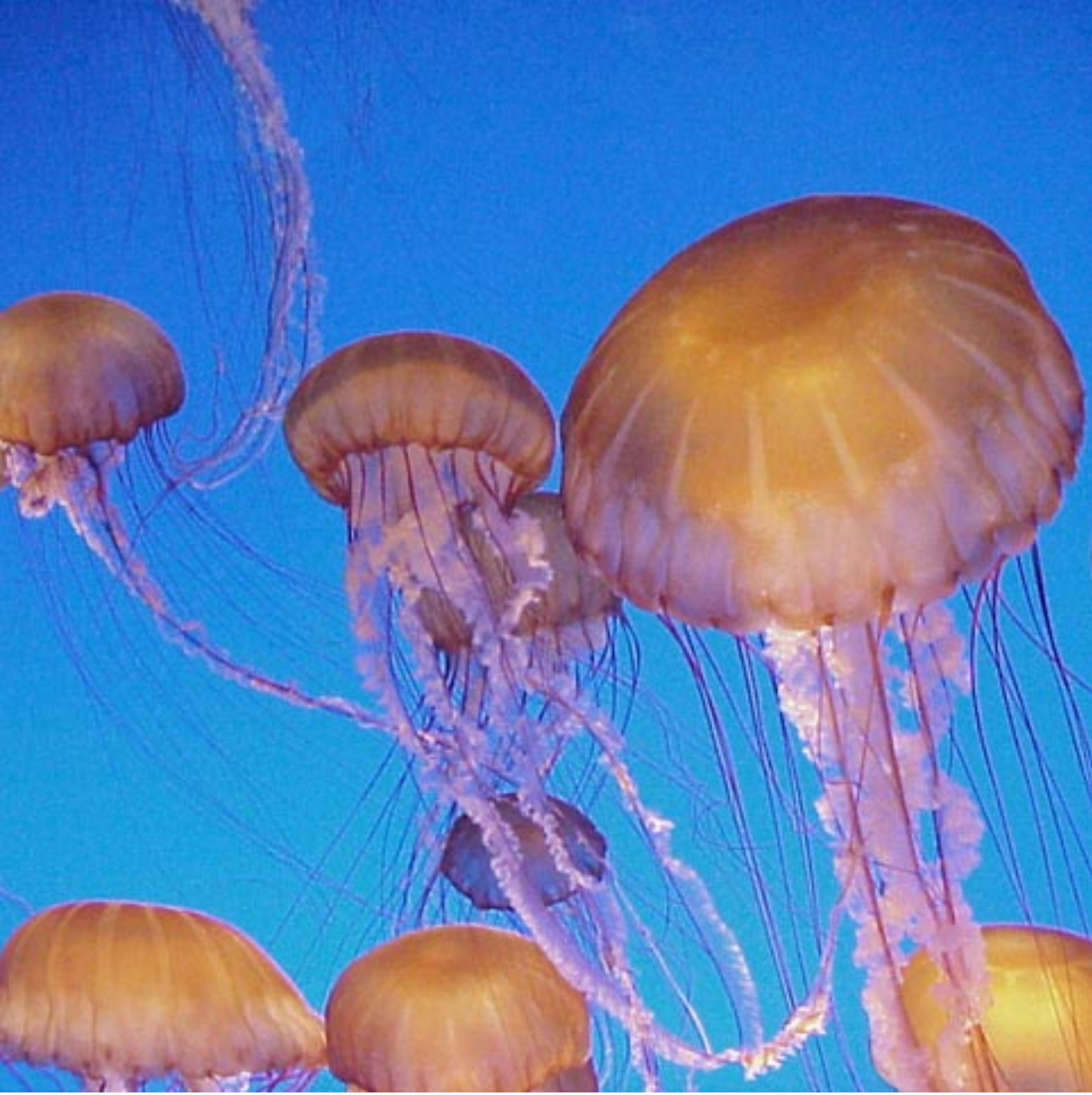
Sphères, dômes, disques.

http://www.astrolab.be/html/actueel_bestfoto_nl.html

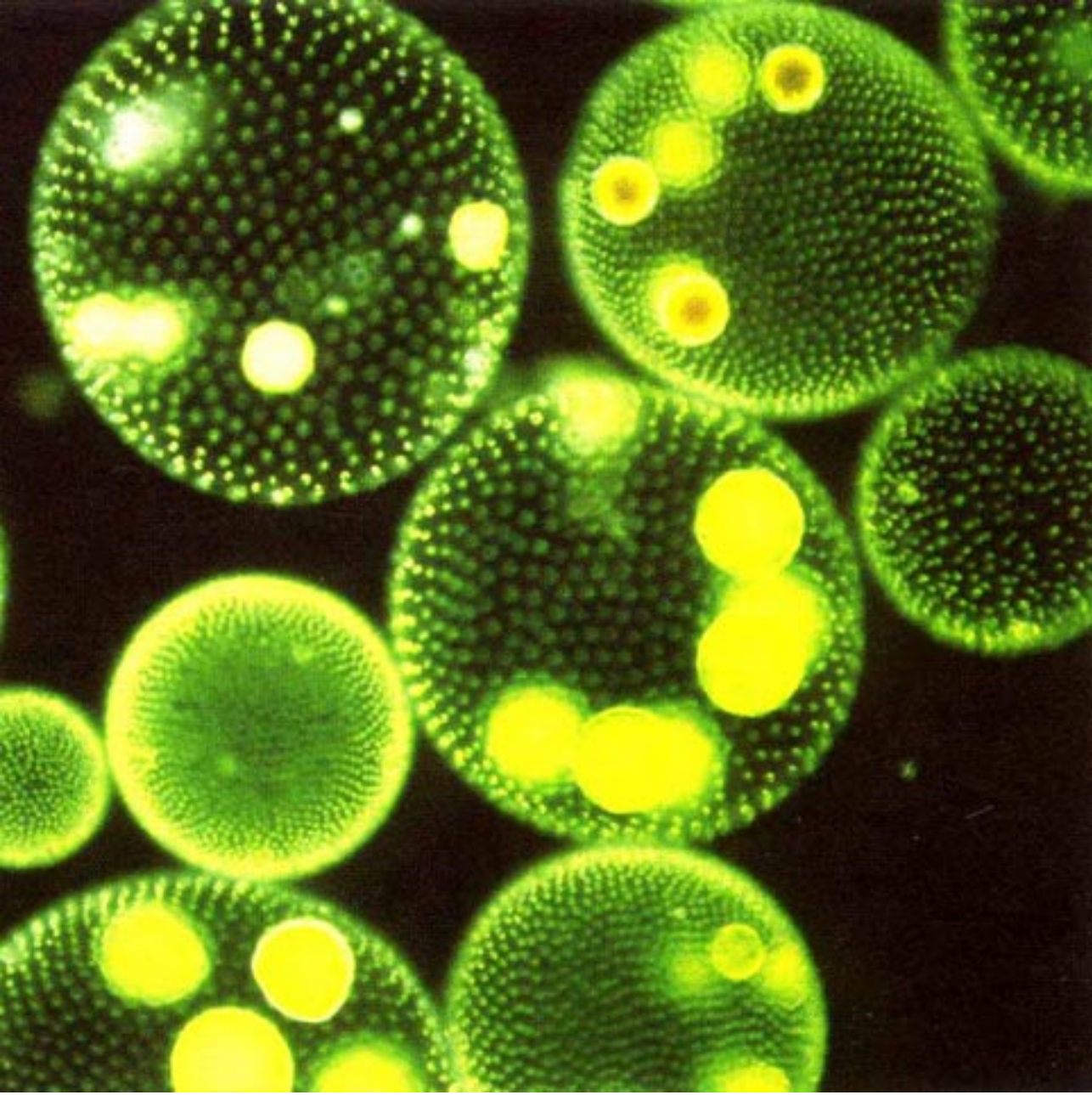
David Quéré & Claudius Laburthe : La physique d’une goutte d’eau.
http://www.agrobiosciences.org/article.php3?id_article=1053



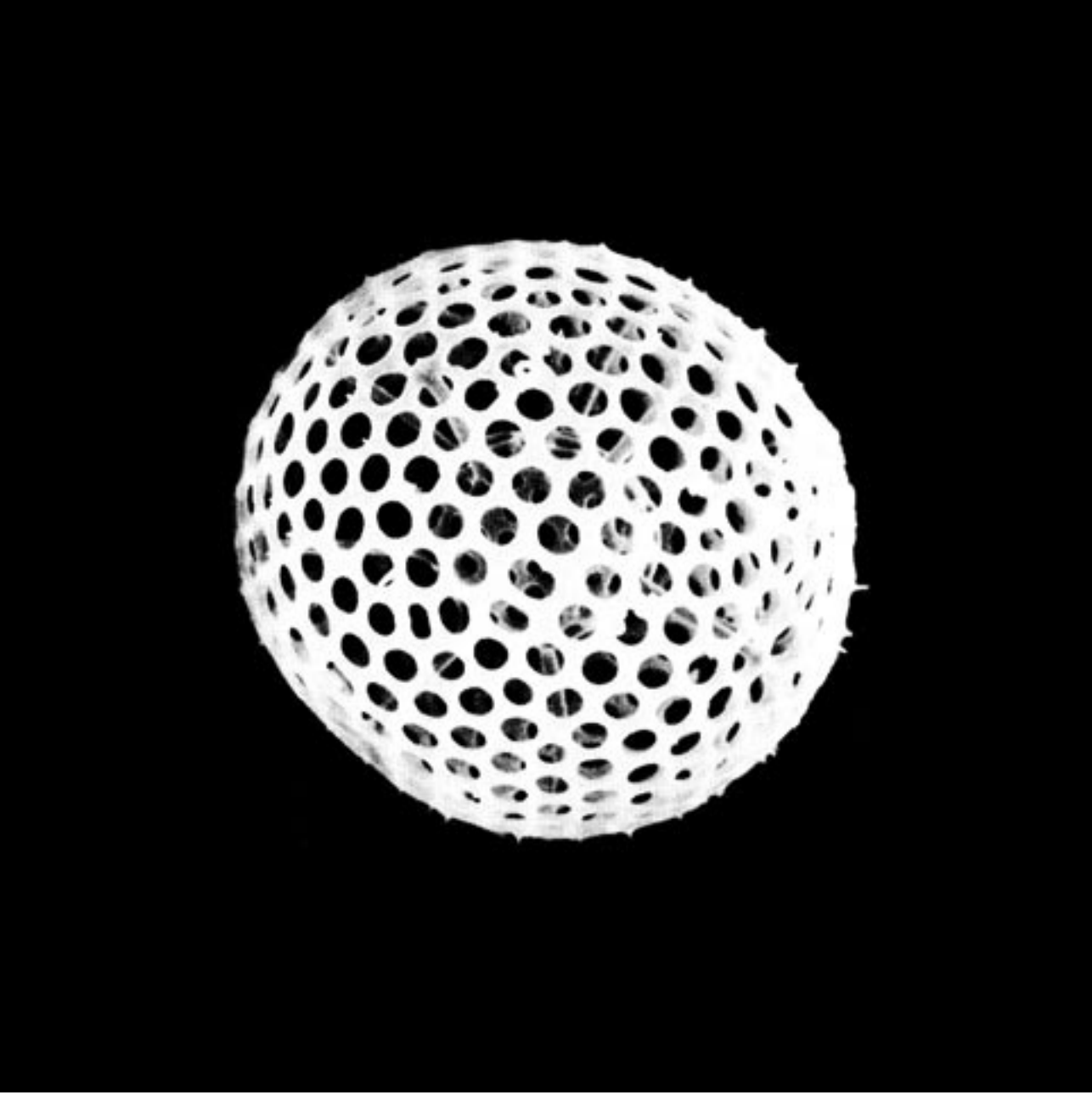
http://remf.dartmouth.edu/images/insectPart3SEM/



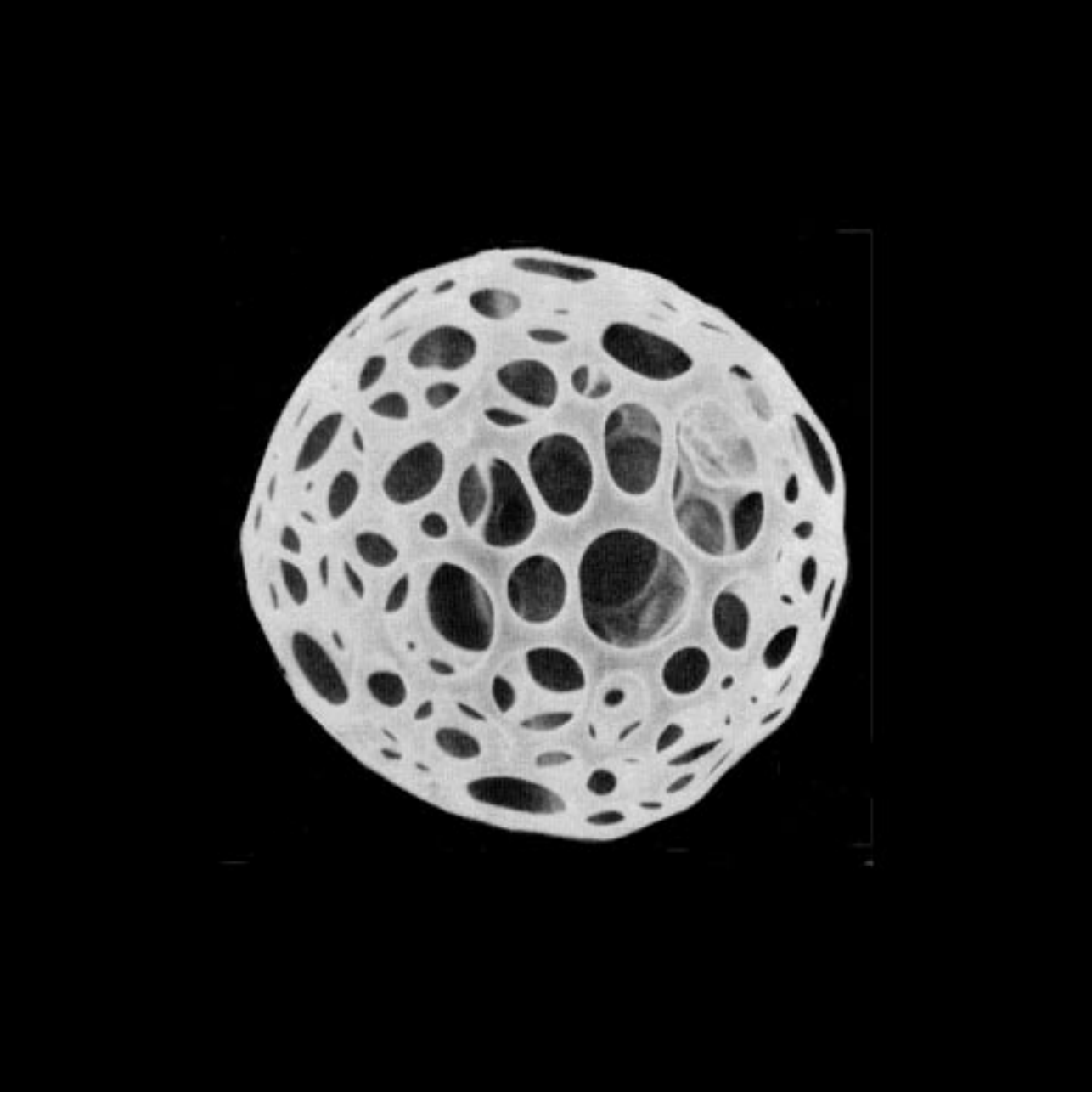
http://en.wikipedia.org/wiki/Jellyfish



Plancton.
http://semsci.u-strasbg.fr/radiolai.htm

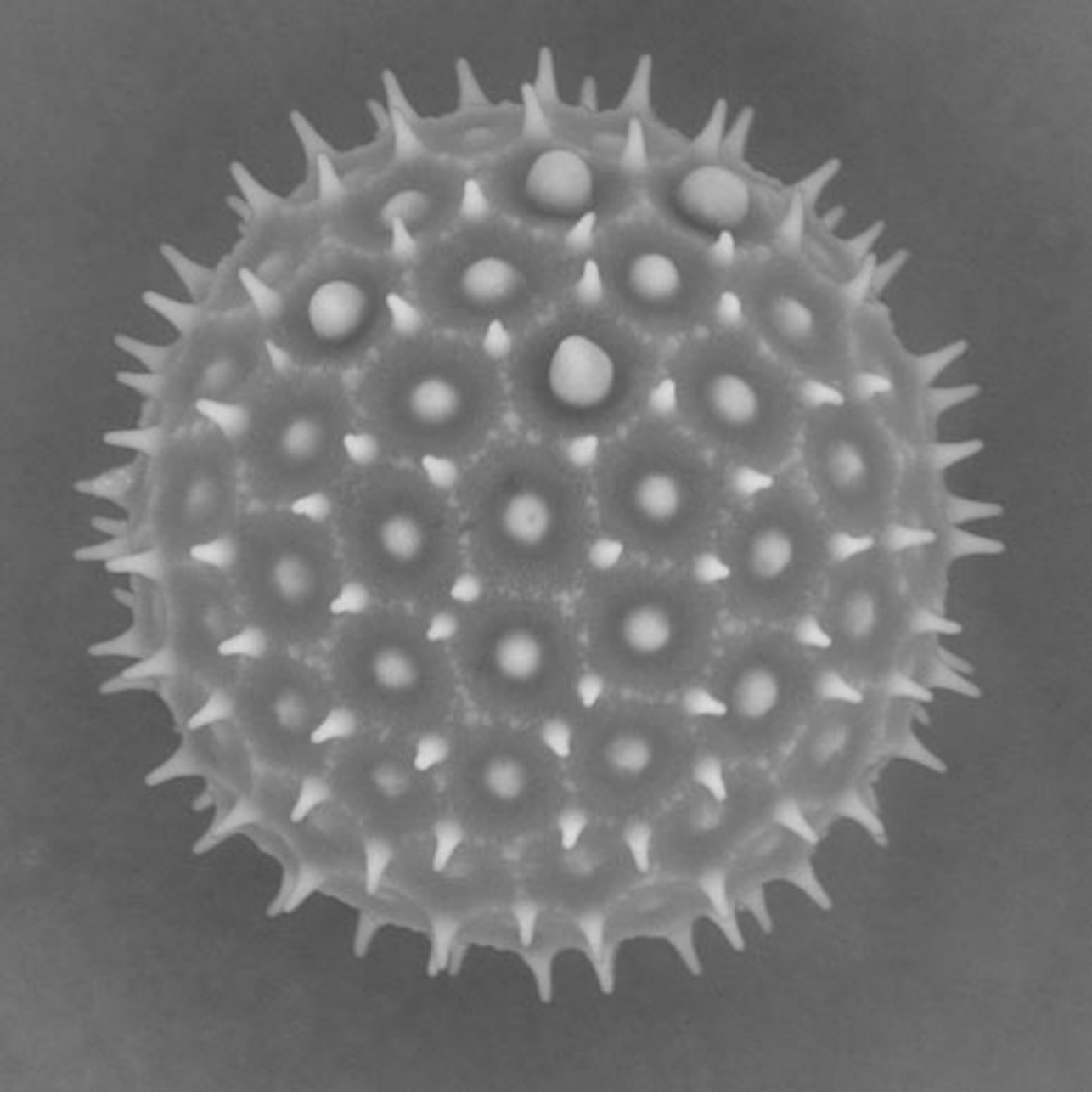


Radiolaire.
http://oceanlink.island.net/oinfo/radiolarians/radiolarian.html



Radiolaire.
http://oceanlink.island.net/oinfo/radiolarians/radiolarian.html

Grain de pollen.
http://www.immediart.com/

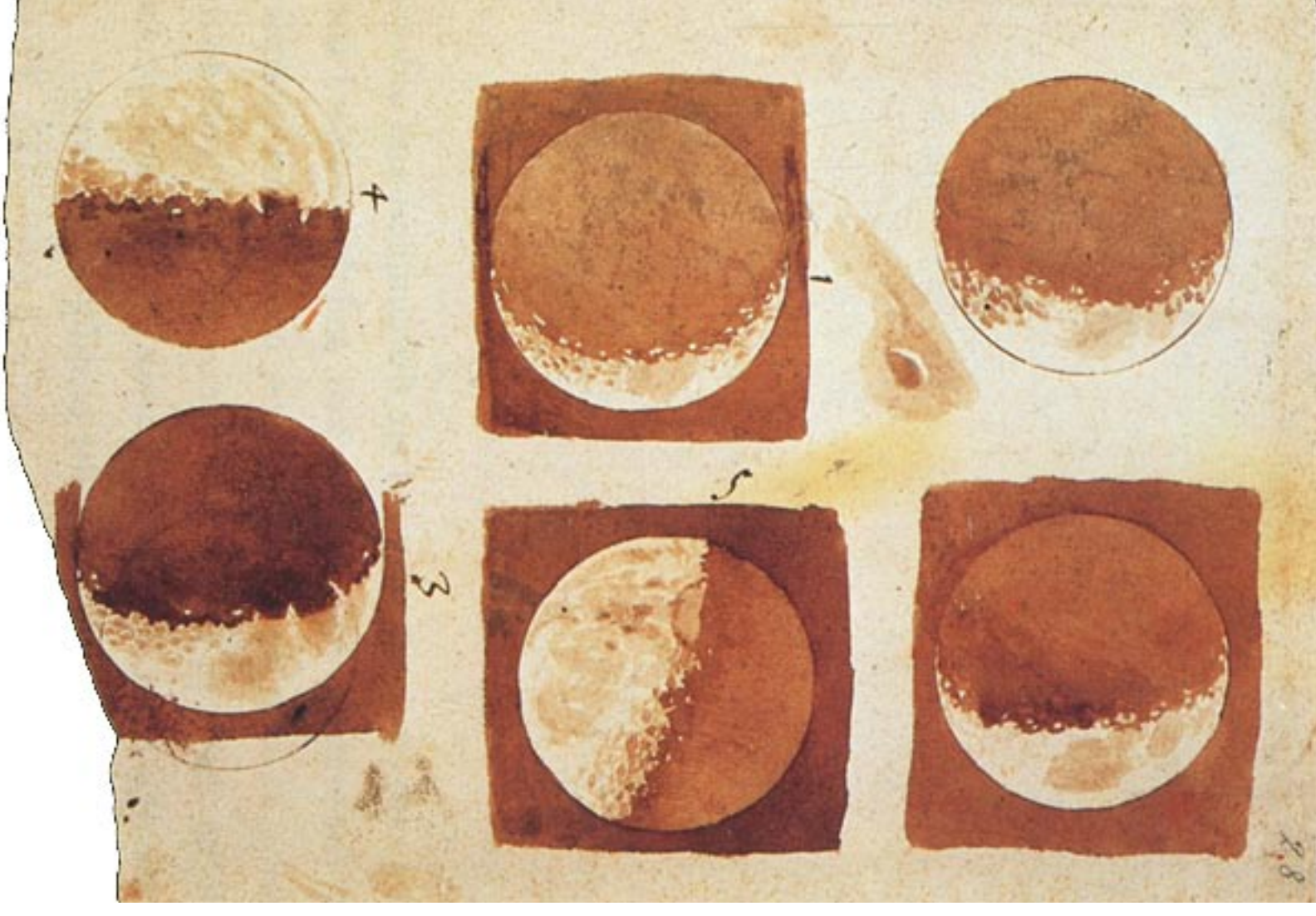


Grain de pollen.
http://www.westga.edu/~geosci/wgmc/plants_pics.htm

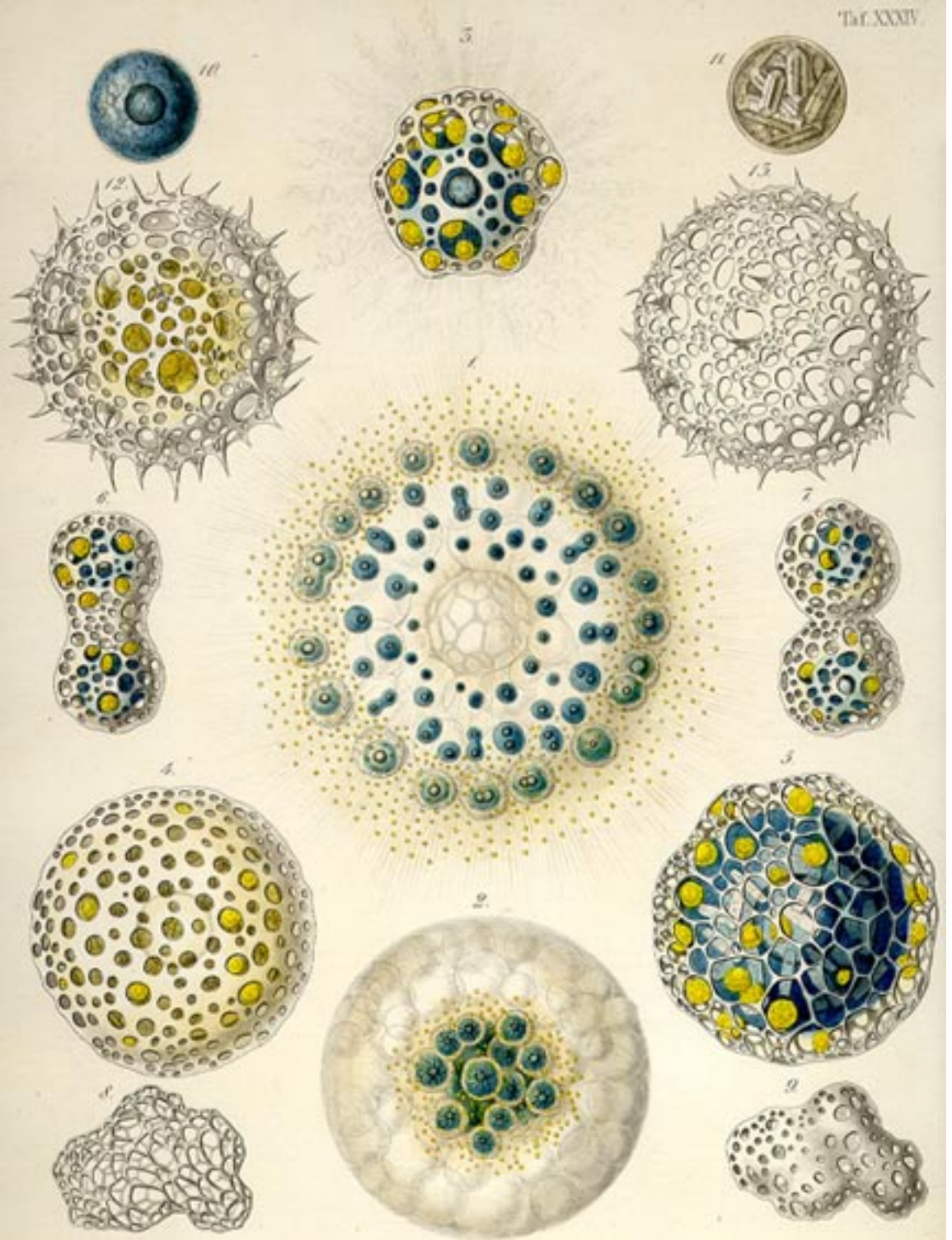
Akènes de pissenlit.

Igloo.
http://home.no.net/gedra/igloo_bg.htm

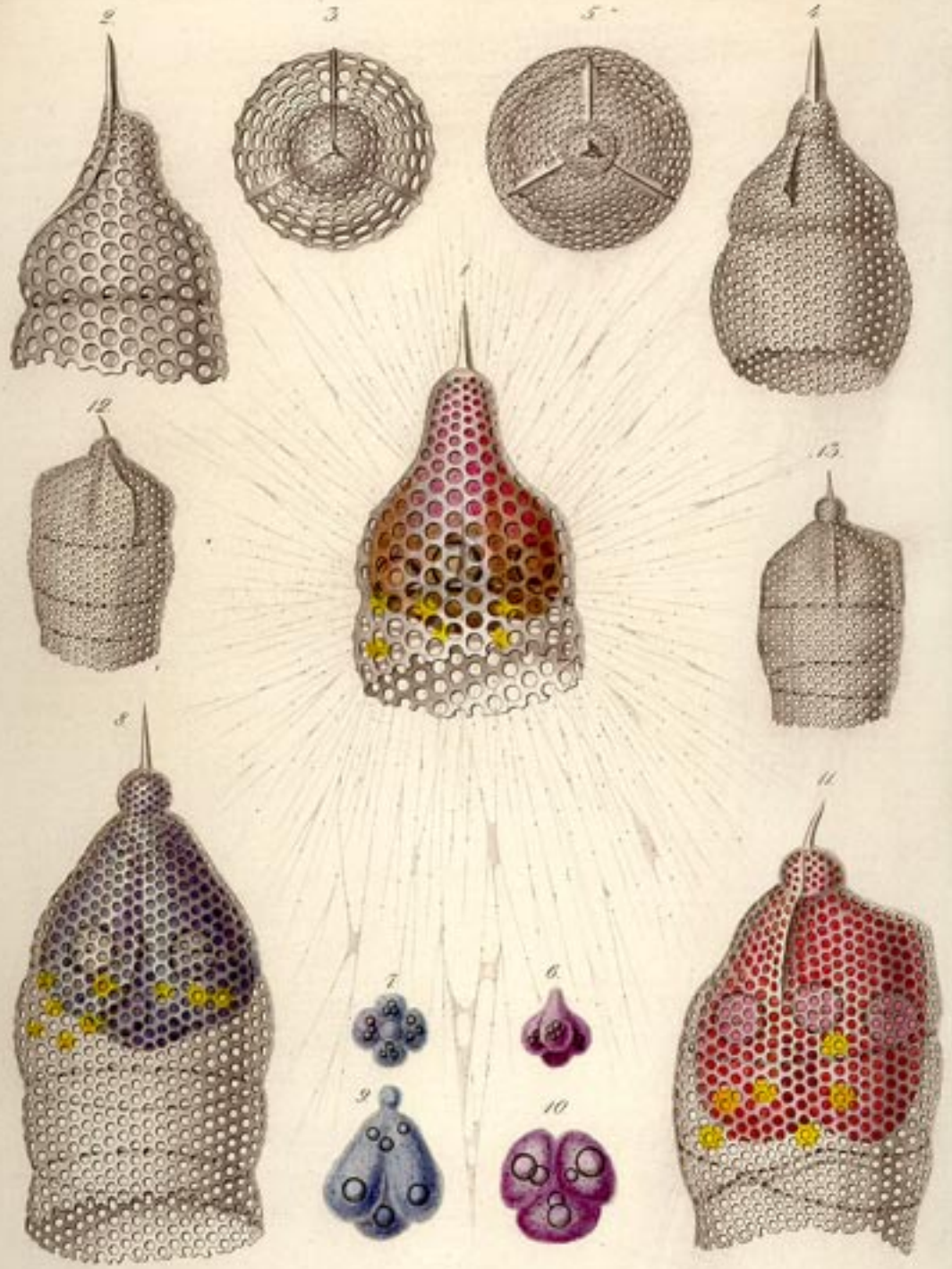
Le dôme de la basilique Saint-Pierre à Rome (1590).
http://www.stpetersbasilica.org/



Galilée : Phases de la lune (1616).
http://galileo.rice.edu/galileo.html



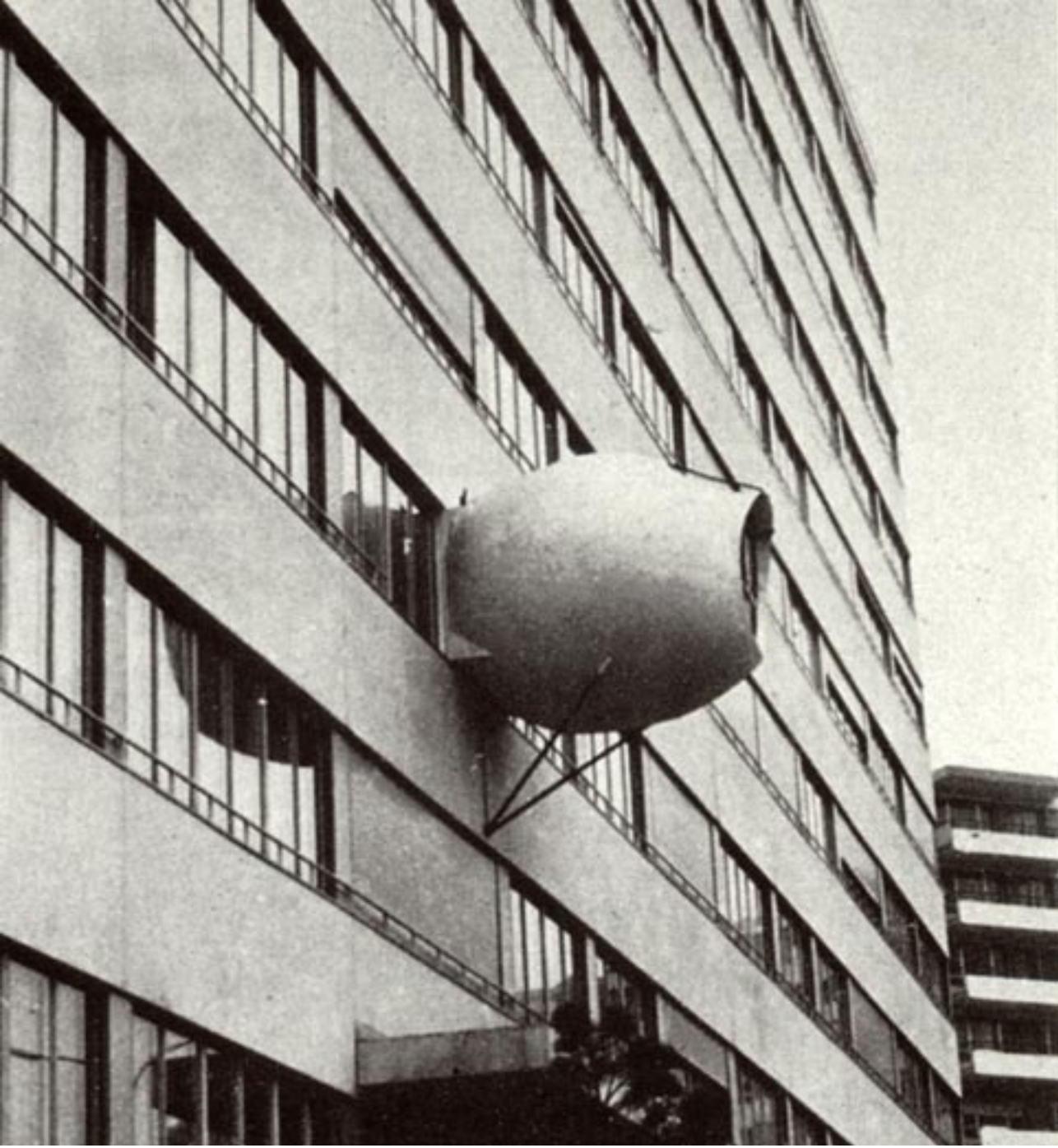
1–13. Collosphaera. 1–11. C. Huxleyi, Müller. 12 13. C. spinosa, Hkl.



1–13. Eucyrtidium. 1–3. E. cranoides, Hkl. 4–7. E. carinatum, Hkl.
8–10. E. Galea, Hkl. 11–13. E. anomalum, Hkl.

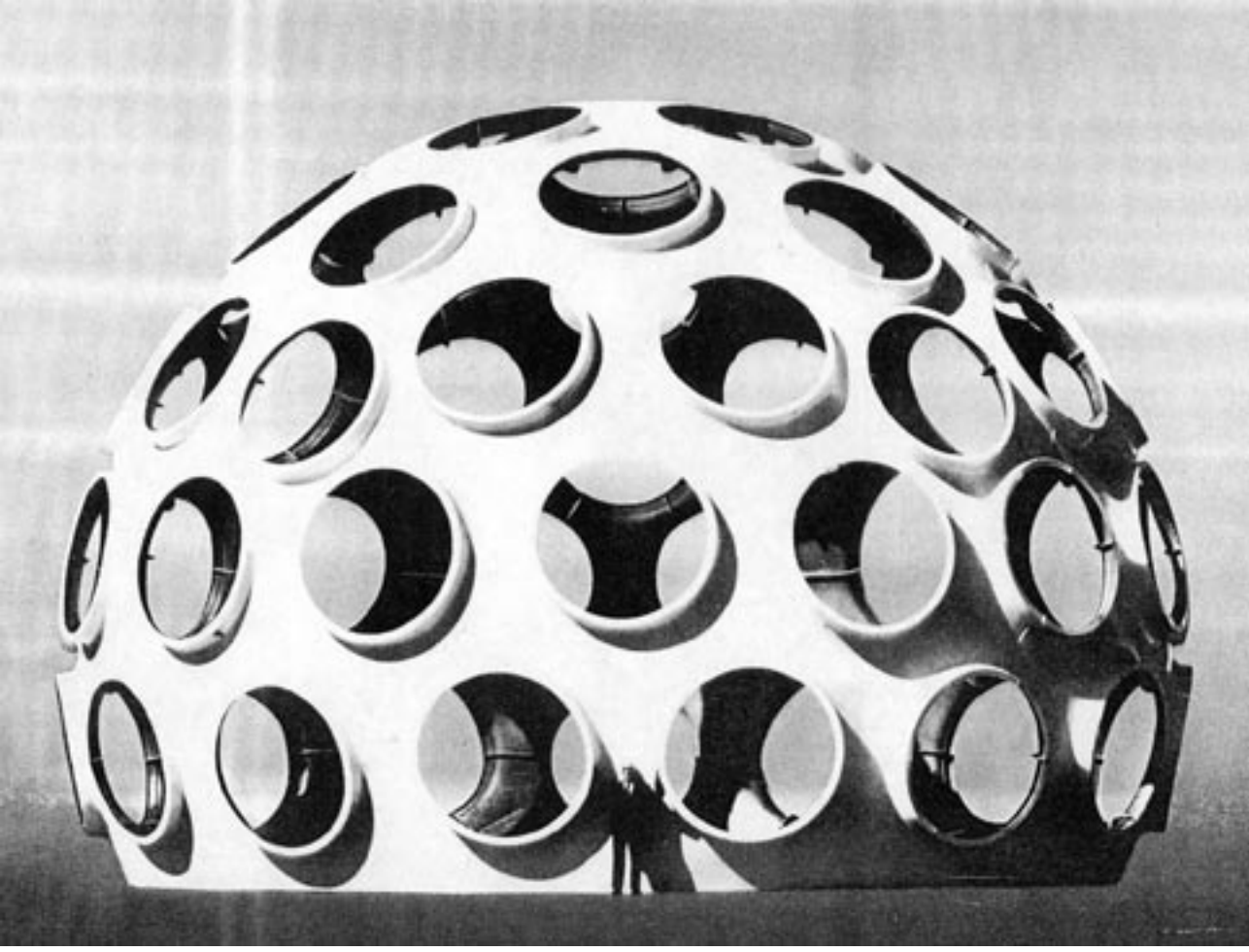
Piet Mondrian : Pier and Ocean (1914).
http://en.wikipedia.org/wiki/Piet_Mondrian

László Moholy-Nagy : Nuclear II (1946).
http://www.moholy-nagy.org/



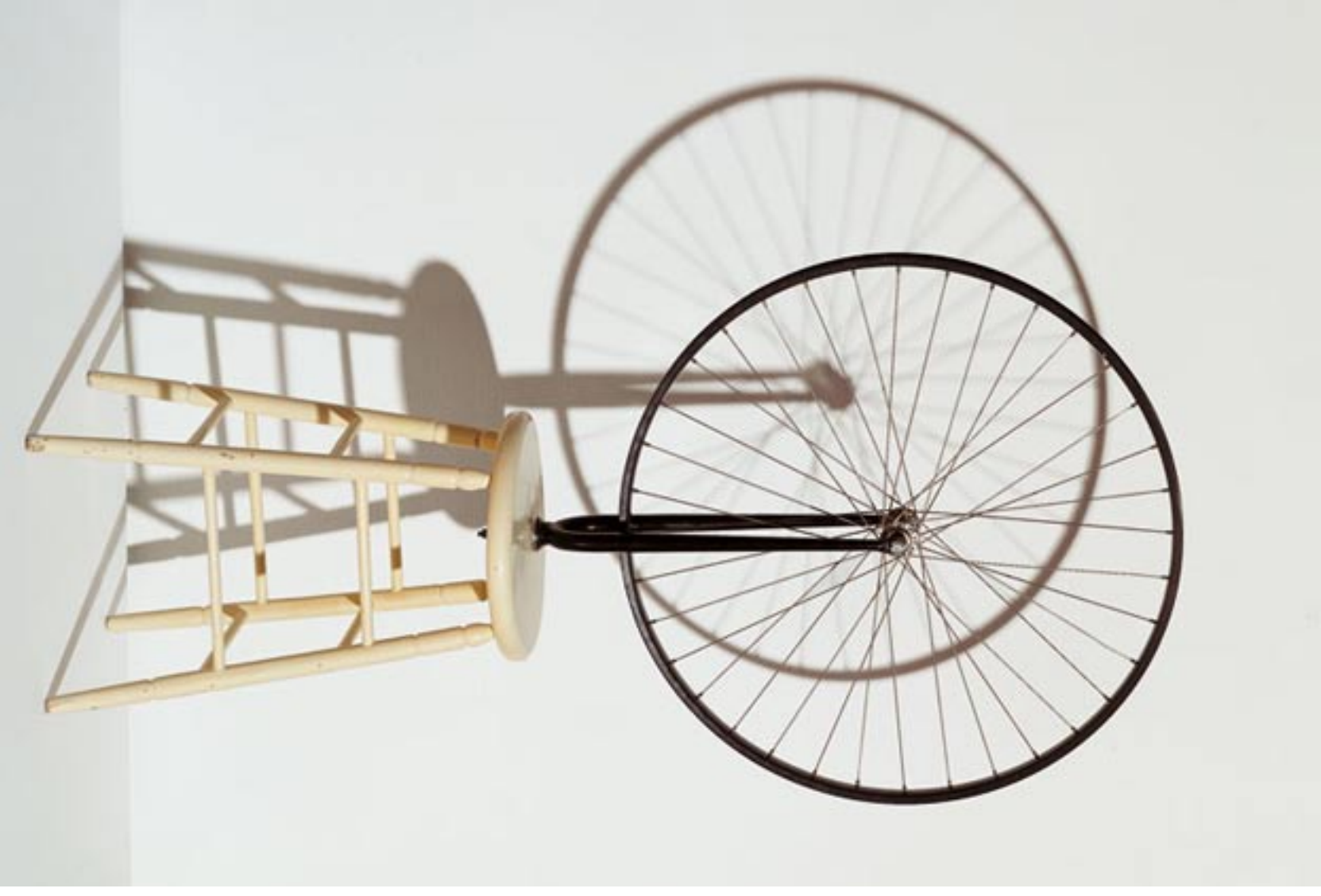
Jean-Louis Rey, dit Chanéac : Cellule parasite (1966).

Richard Buckminster Fuller : Dome over NYC (1968).
http://www.bfi.org/

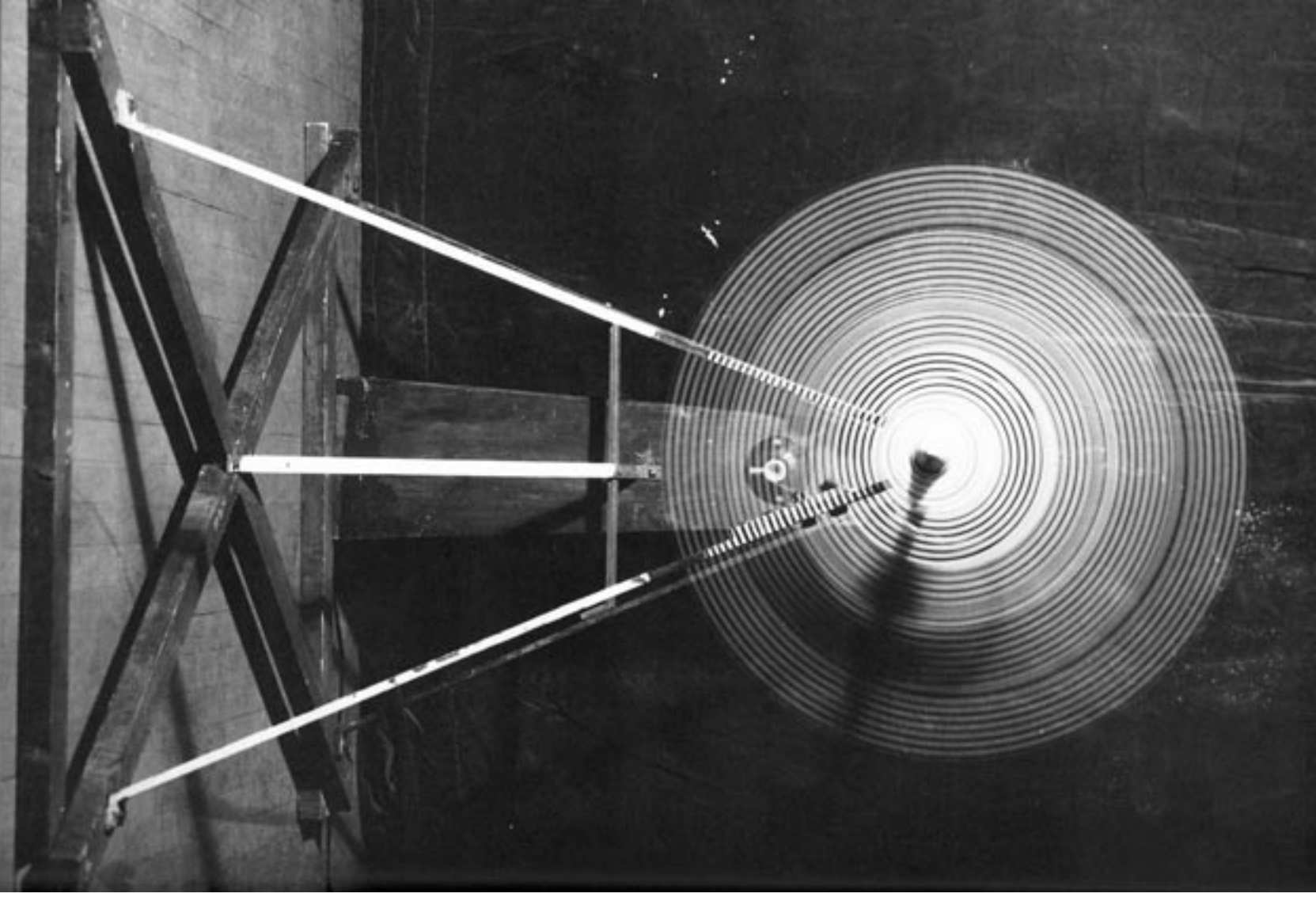


Richard Buckminster Fuller : Fly’s eye dome (1977).
http://www.bfi.org/

Richard Buckminster Fuller : Pavillon des USA,
exposition internationale de Montréal (1967).
http://fr.wikipedia.org/wiki/Richard_Buckminster_Fuller

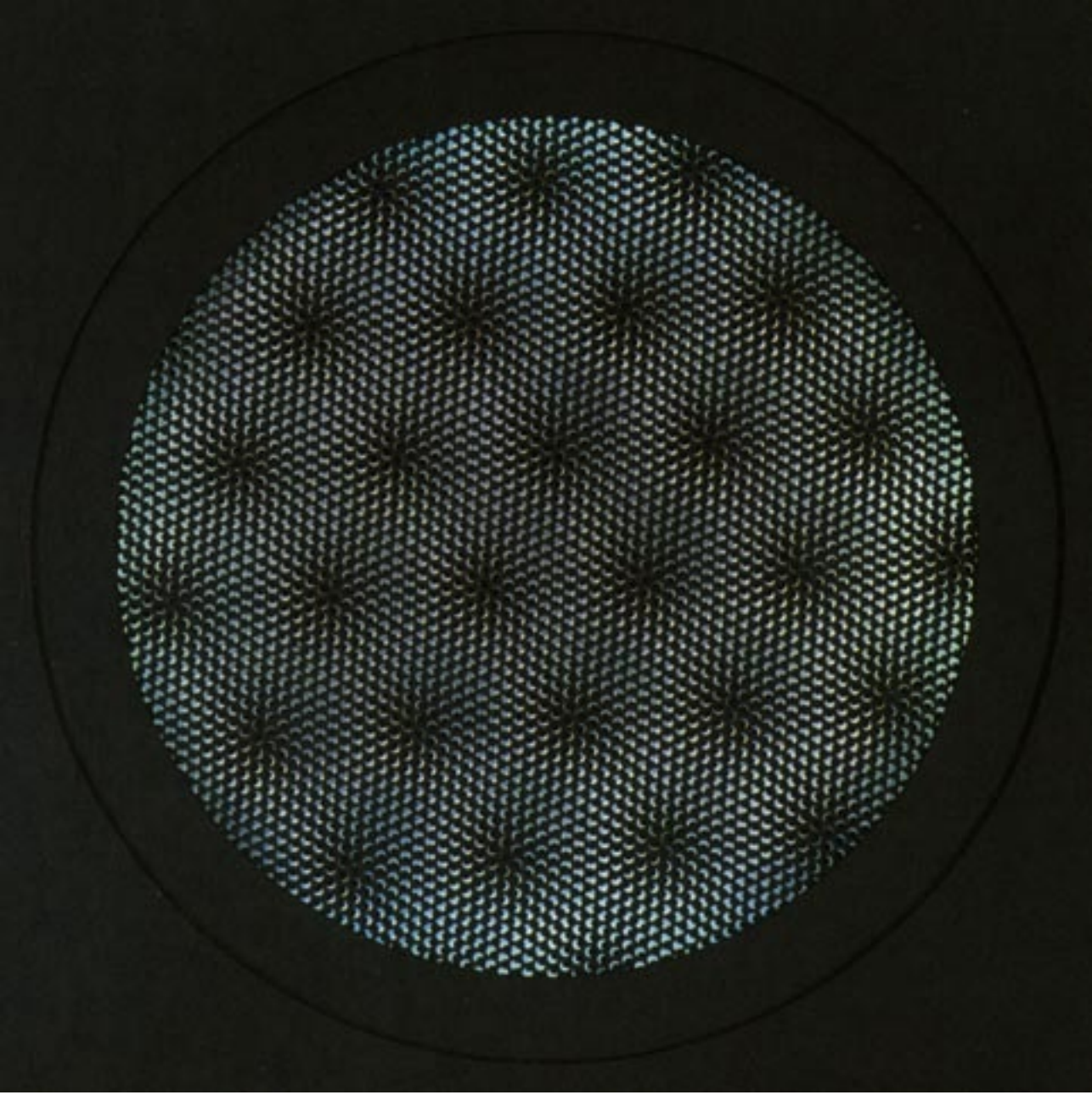


Marcel Duchamp : Roue de bicyclette. Original perdu (Paris, 1913).
Réplique sous la direction de Marcel Duchamp (1964).
http://www.marcelduchamp.net/



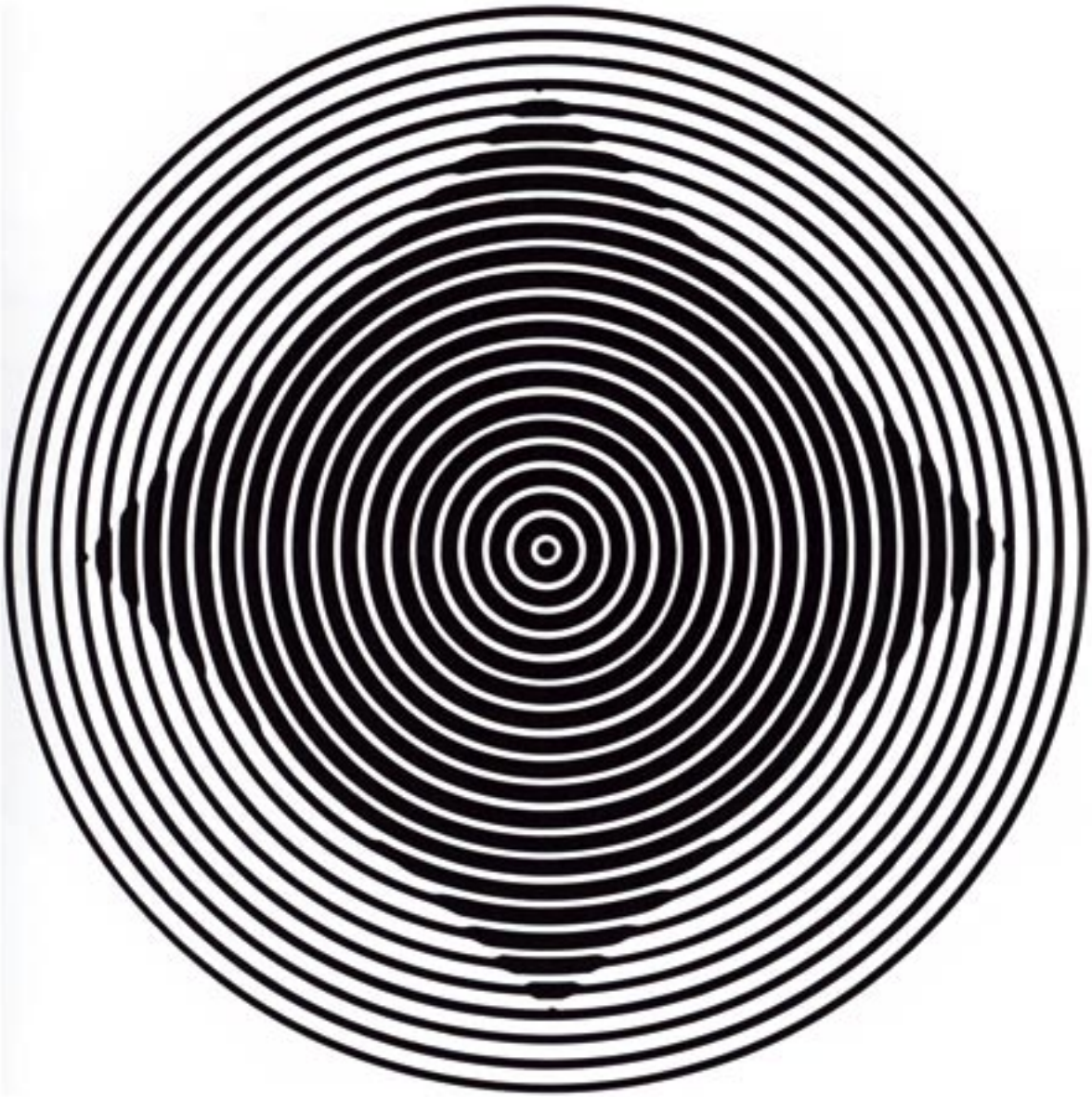
Marcel Duchamp : Rotative plaques verre (1920)
http://en.wikipedia.org/wiki/Duchamp

Marcel Duchamp : Rotorelief (1935).
http://www.marcelduchamp.net/



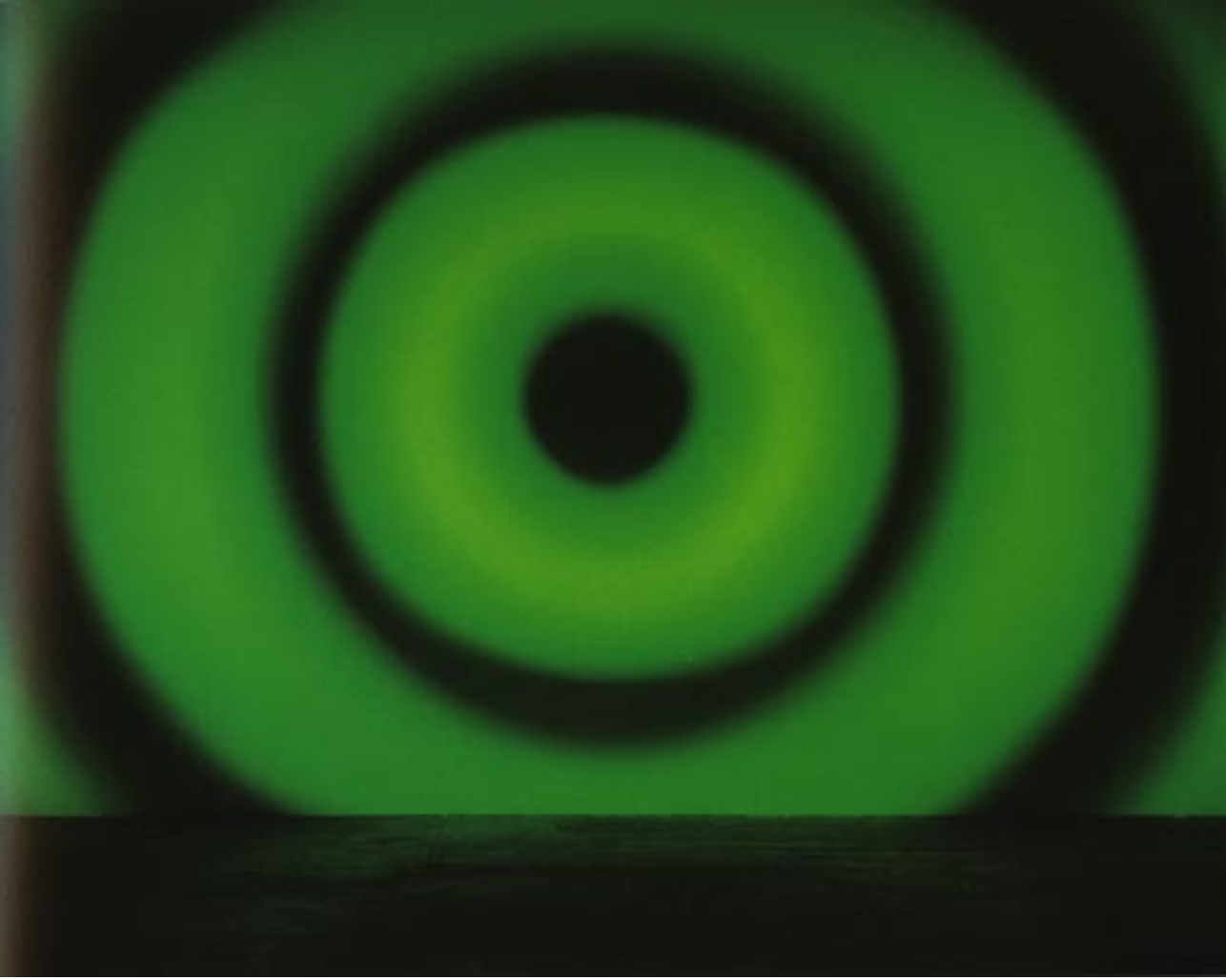
Joël Stein : Acceleration III mouvement optique (1964).

Marina Apollonio : Dinamica circolare (1966)



Victor Vasarely : Gamma (1958-1965).
http://www.fondationvasarely.fr/

Ugo Rondinone :
No. 337-ACHTUNDZWANZIGSTERMAIZWEITAUSENDUNDVIER (2004)
Acrylic on canvas (2006).
http://www.whitechapel.org/content.php?page_id=2212



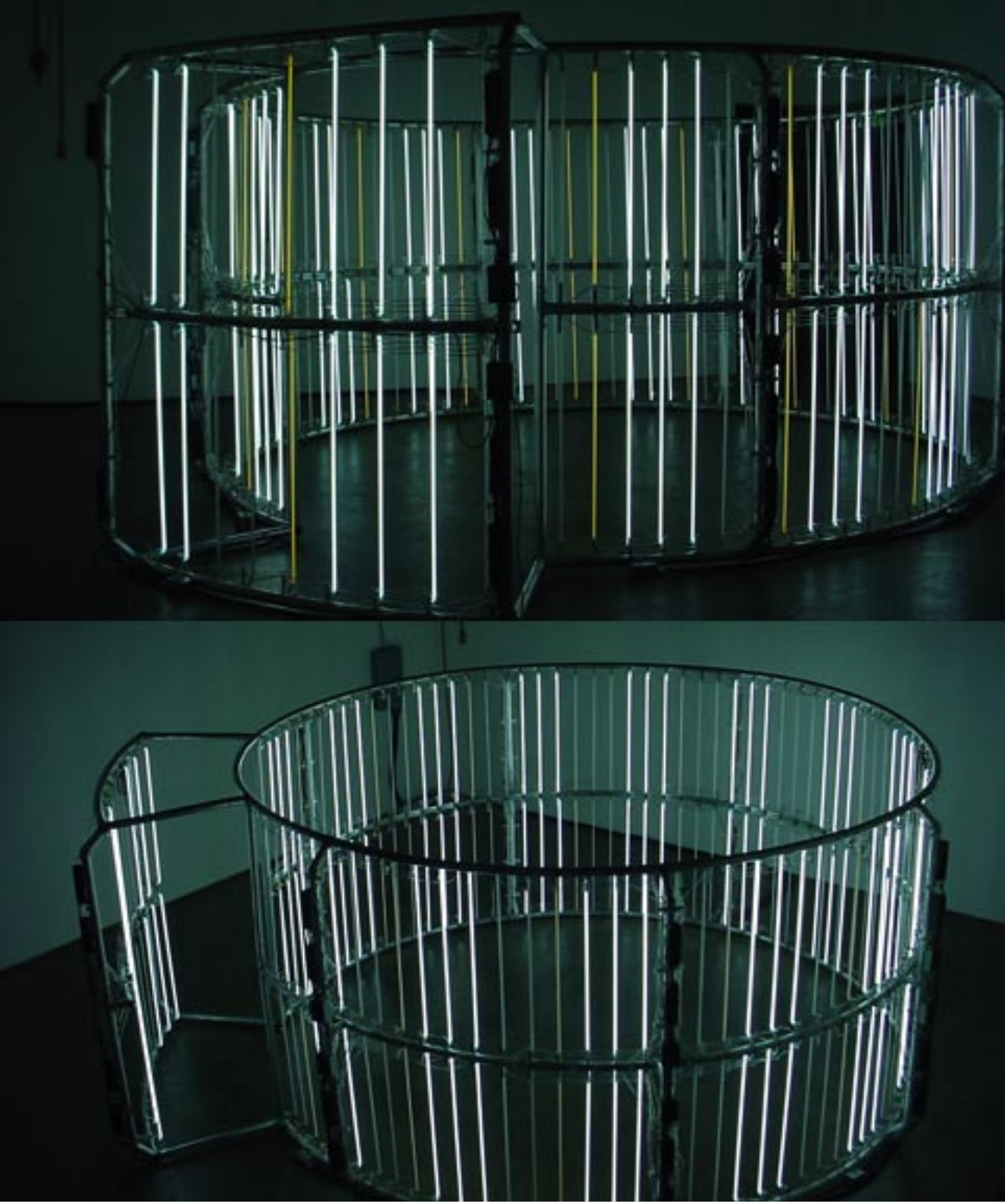
Ann Veronica Janssens : Donuts (2006).
http://www.gms.be/

Philippe Decrauzat : Fight disc (2002).
http://www.praz-delavallade.com/

Philippe Decrauzat : Fight disc (2002).
http://www.praz-delavallade.com/

Philippe Decrauzat : Sans titre (2005).
http://www.praz-delavallade.com/

Jean-Pierre Yvaral : Sculpture cynétique pénétrable (1966).



Carsten Höller : Neon circle (2001).
http://www.airdeparis.com/

Olafur Eliasson : The weather project (2003).
http://www.olafureliasson.net/

Ann Veronica Janssens : Untitled (2003).
http://www.gms.be/

Meg Cranston : The Complete Works of Jane Austen (1991),
globe de vinyl couleur crème de 457 cm de diamètre,
rempli de 100.000 litres d'air, volume dont une personne est censée
avoir besoin pour lire les oeuvres complètes de Jane Austen.
http://www.galeriemichaeljanssen.de/
http://www.thehappylion.com/

Klaus Pinter : Rebonds (2002),
Oeuvre éphémère pour le Panthéon.
Structure gonflable. 20m X 26m X 15m.
http://www.klauspinter.net/

Yayoi Kusama : Dots Obsession (2000),
11 ballons, points de vinyl, installation.
http://www.yayoi-kusama.jp/

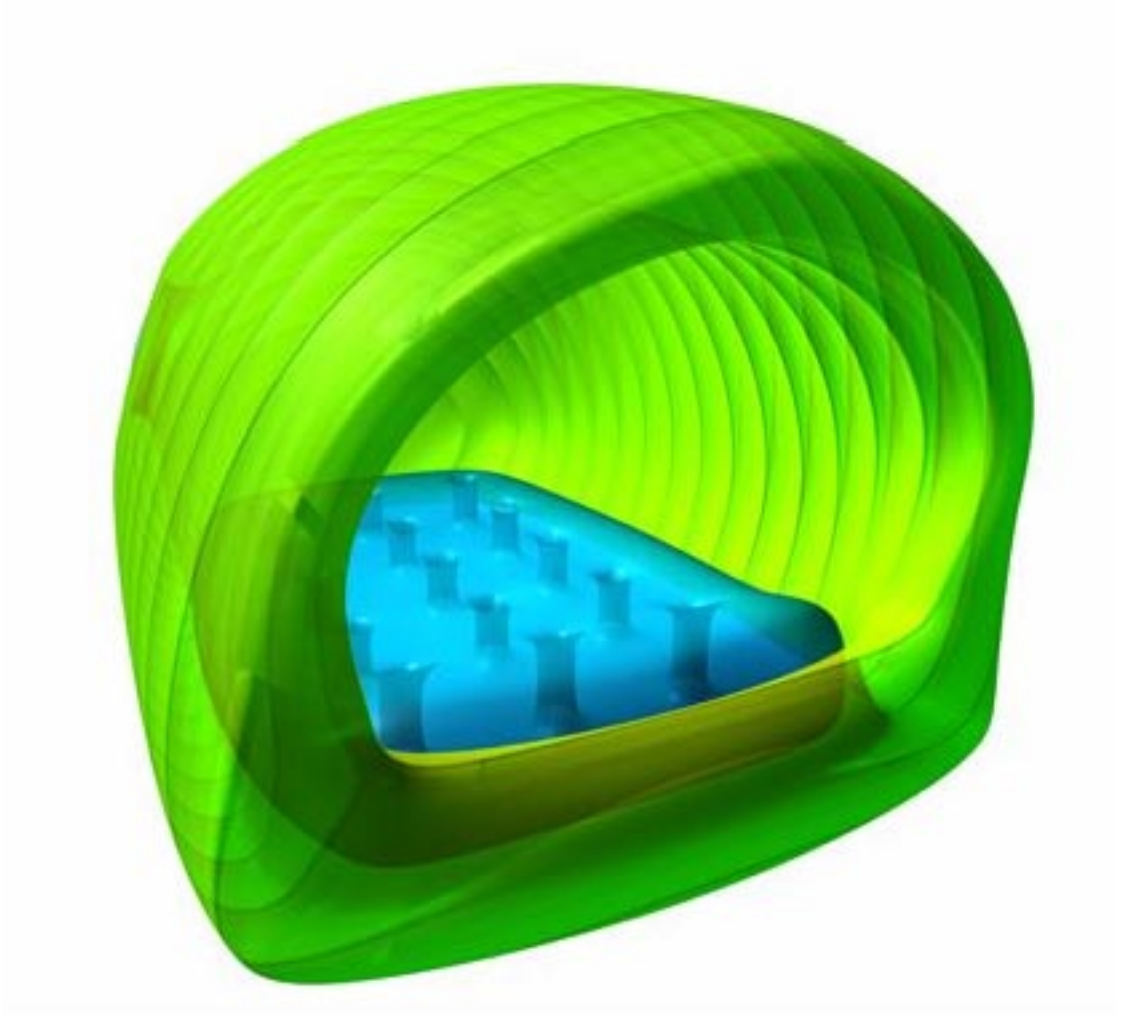
Michel François : Boule élastique (1989).
http://www.artnet.com/artist/20573/michel-franois.html

Michel François : Éponge (1989).
http://www.artnet.com/artist/20573/michel-franois.html

Nils Udo: The Nest (1978).
http://greenmuseum.org/content/artist_index/artist_id-36.html

Andy Goldsworthy : Stone Houses (2004).
http://www.metmuseum.org/

Michael Rakowitz : ParaSITE, NYC (1997).
http://www.michaelrakowitz.com/
http://www.lombard-freid.com/

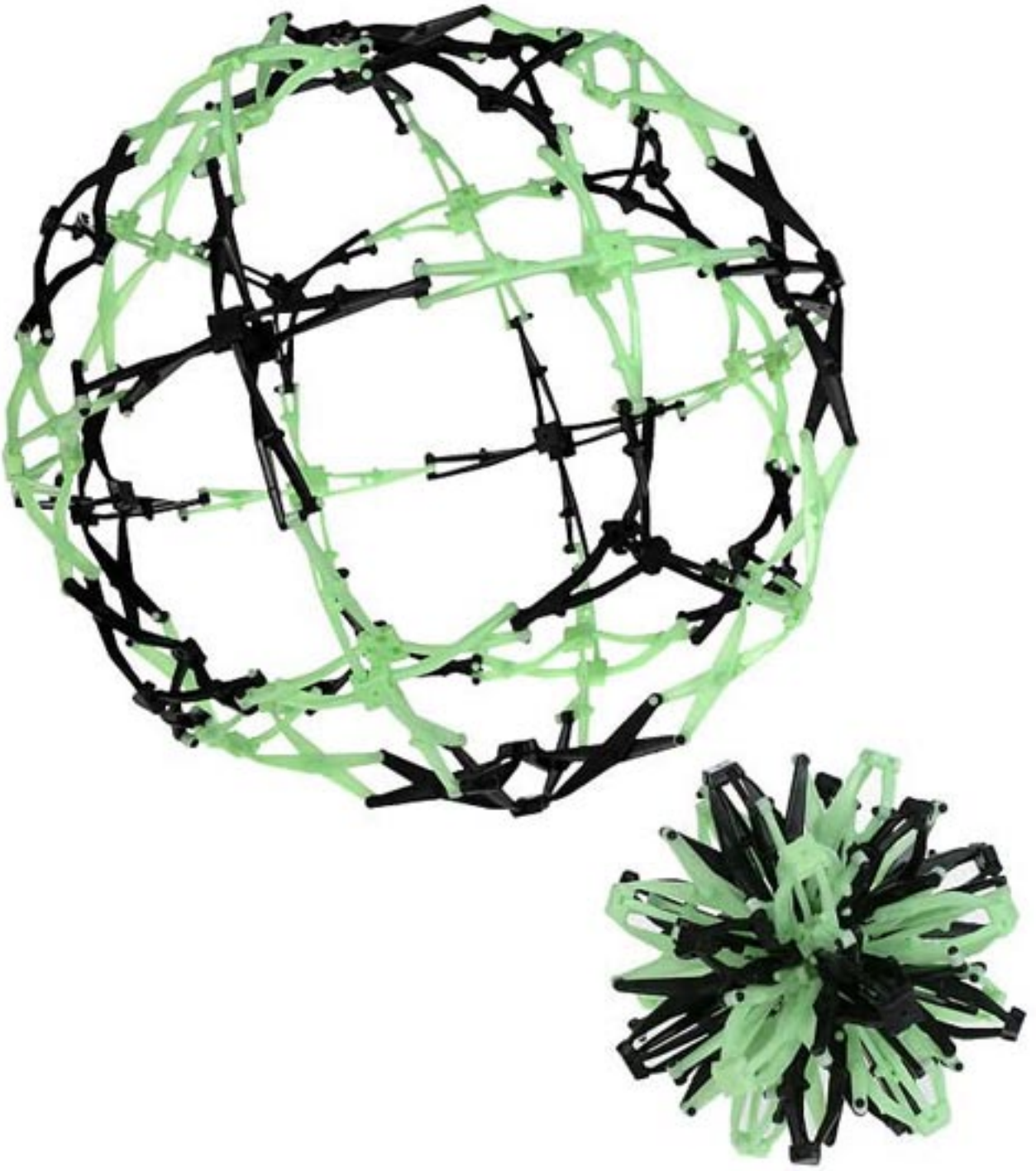


Cameron McNall and Damon Seeley : Urban Nomad Shelter (2005).
http://electroland.net/

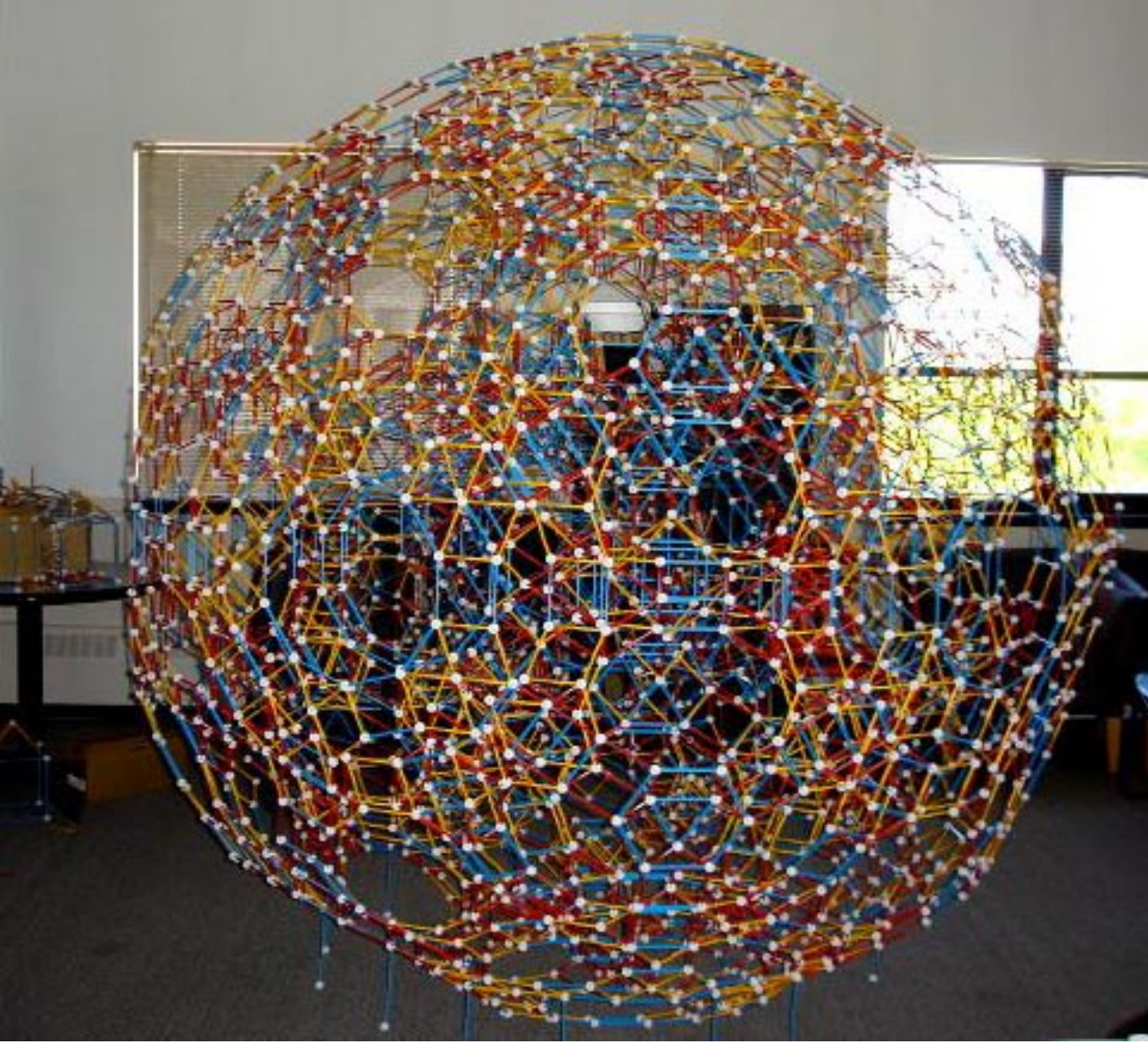
Cameron McNall and Damon Seeley : Urban Nomad Shelter (2005).
http://electroland.net/

Peter Brewin & Will Crawford : Concrete canvas (2004).
http://www.concretecanvas.org.uk/

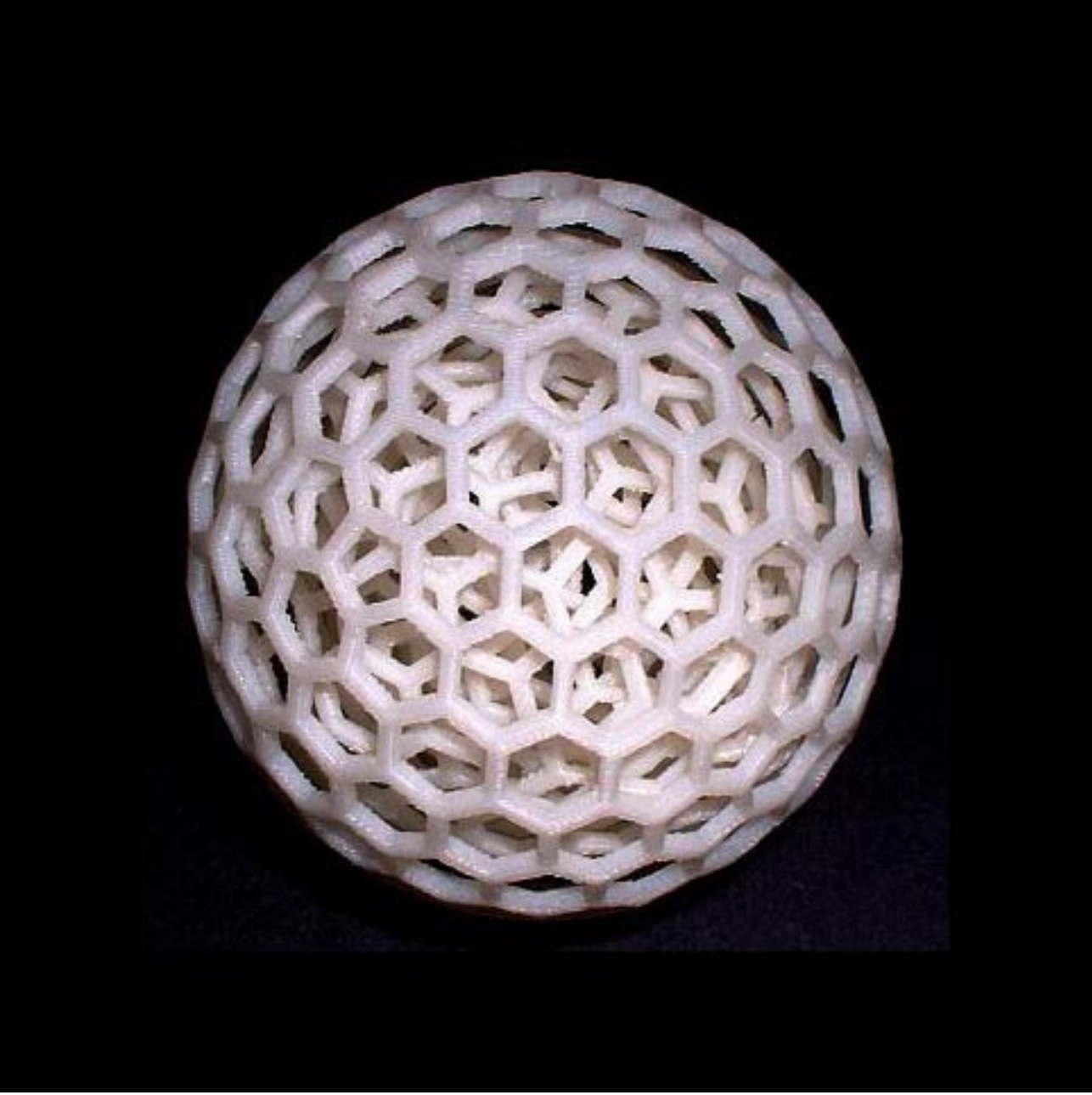
Chuck Hoberman : Hoberman Arch,
Olympic Medals Plaza, Salt Lake City (2002).
http://www.hoberman.com/fold/Associates/associates.htm



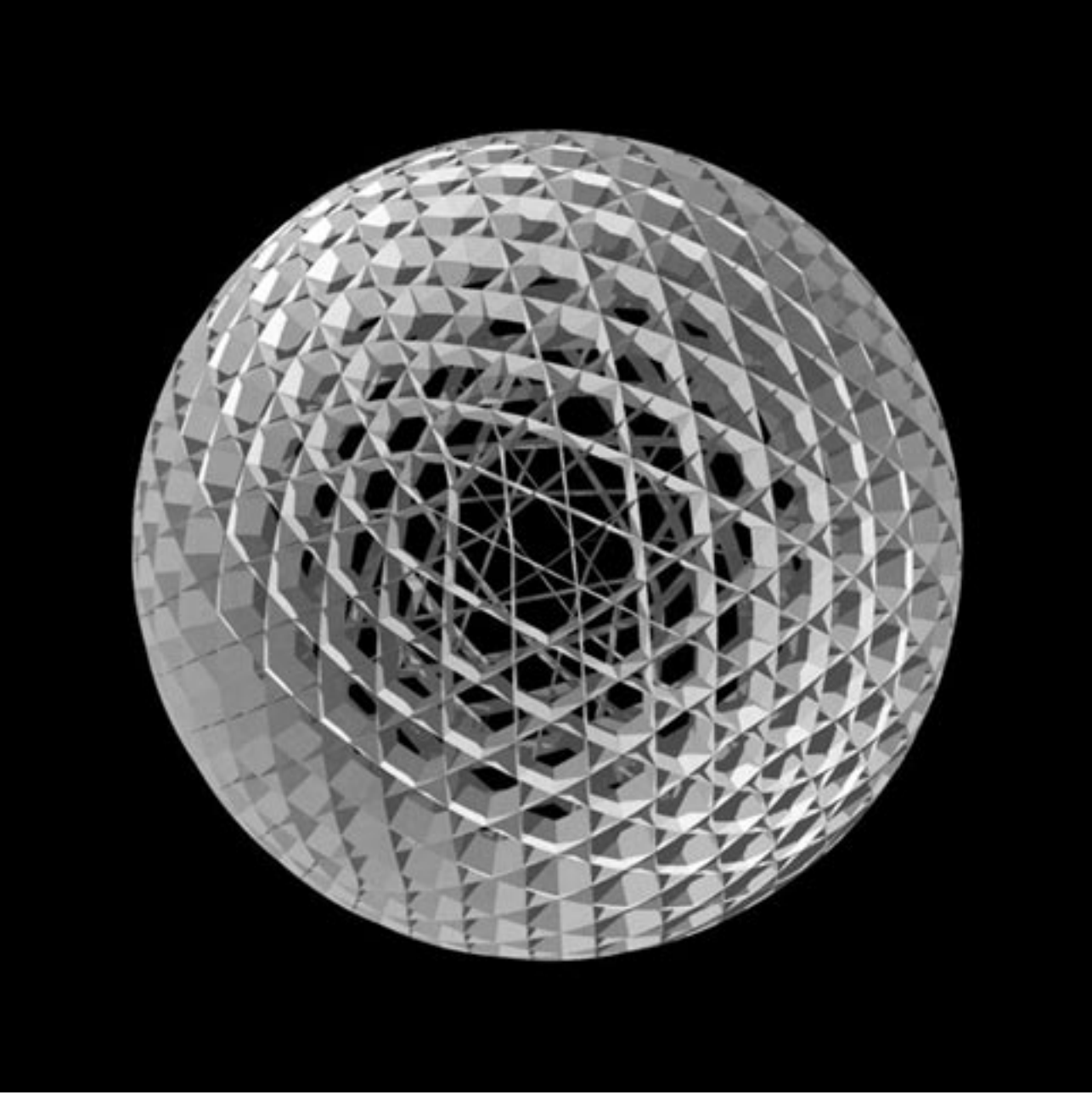
Chuck Hoberman : Hoberman Sphere.
http://www.hoberman.com/fold/Sphere/sphere.htm



George W. Hart : Zometool Polyhedra.
http://www.georgehart.com/
http://www.zometool.com/



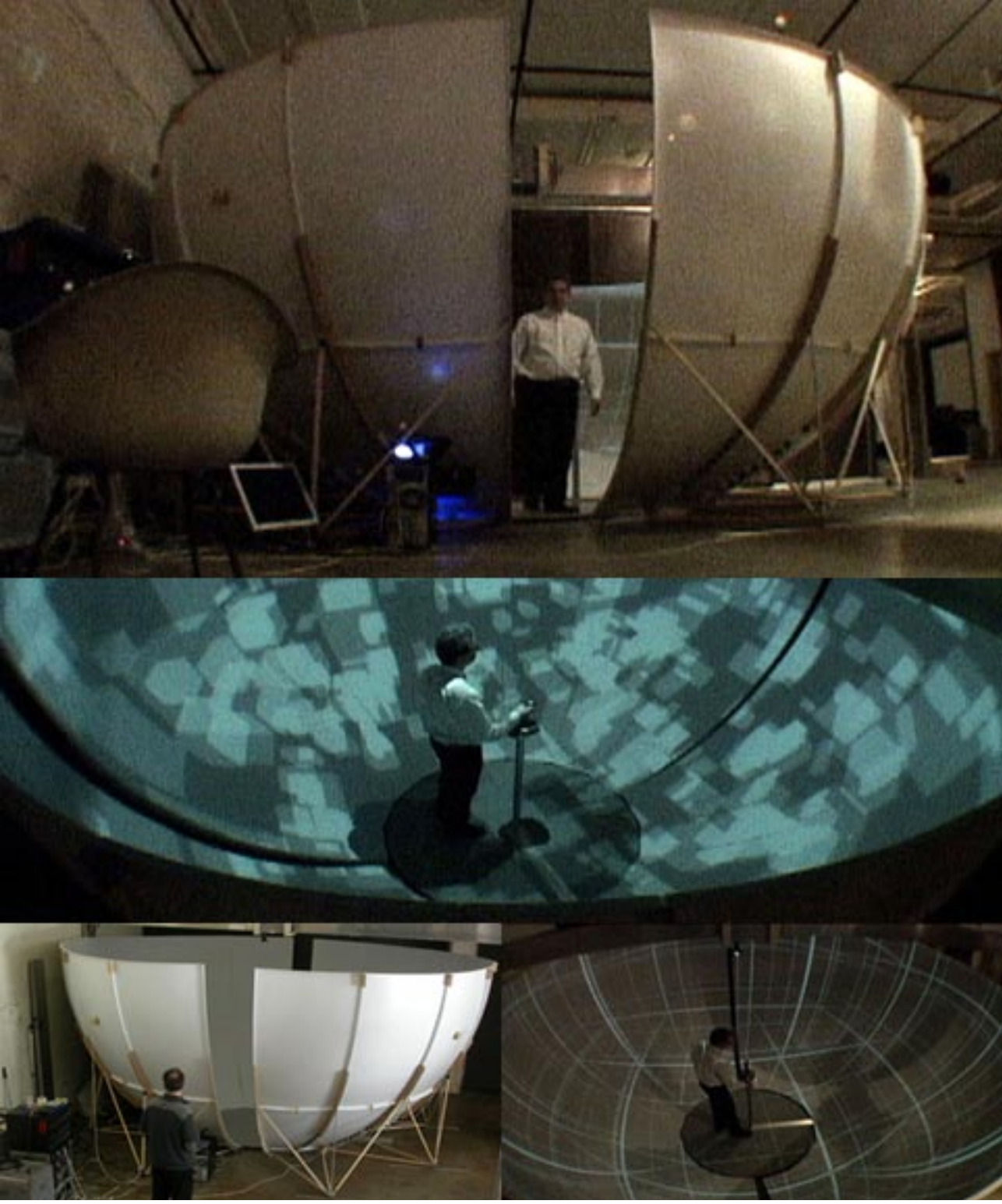
George W. Hart : Seven nested spheres.
http://www.georgehart.com/



George W. Hart : Paper Ball.
http://www.georgehart.com/Paperbll.html

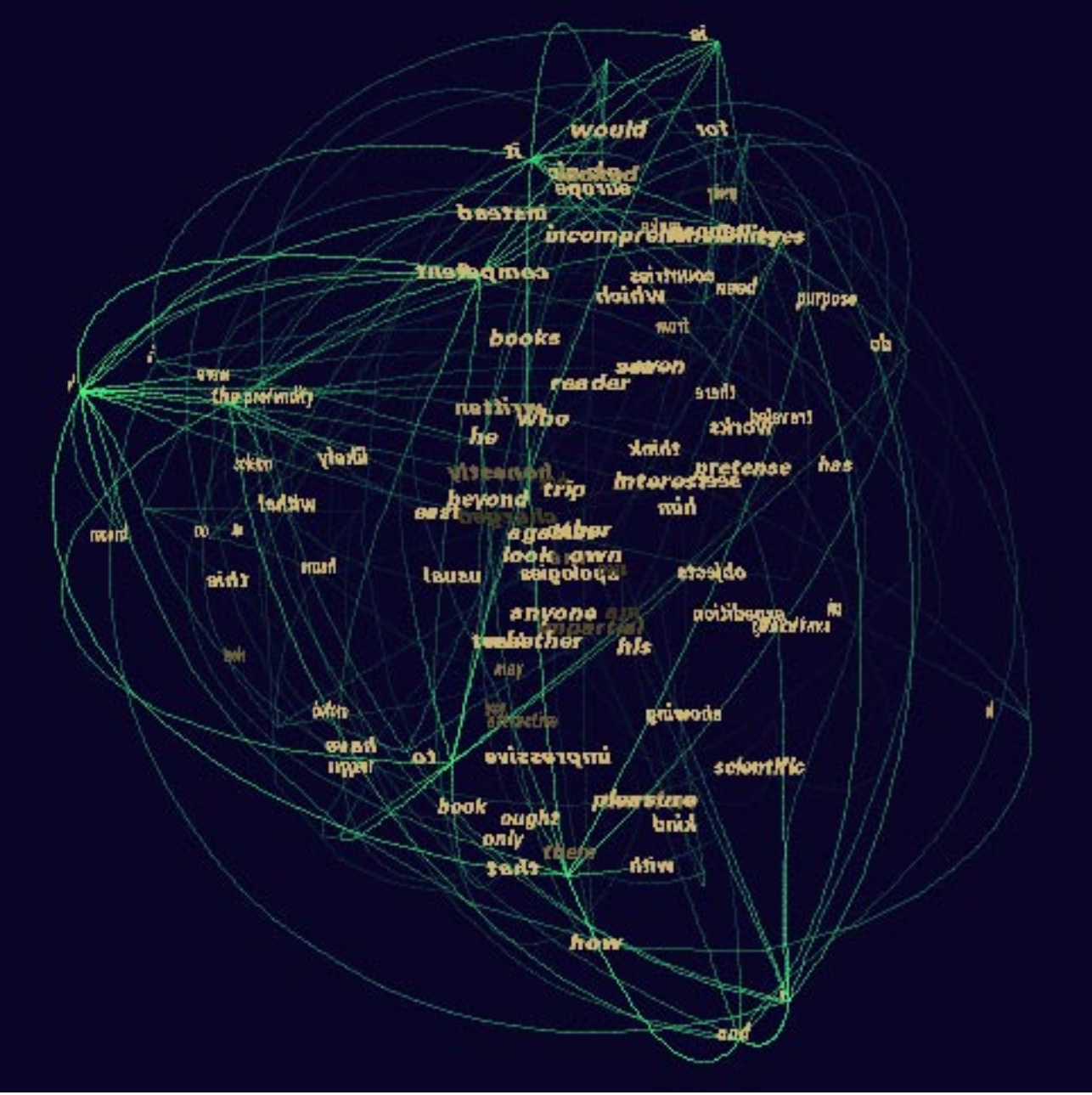
Une des sphères les plus parfaites jamais créées par l'homme (ici, reflétant l'image d'Einstein) : un gyroscope de quartz fondu réalisé pour l'expérience «Gravity Probe B».
Cette sphère est parfaite à 40 atomes d'épaisseur près.
http://en.wikipedia.org/wiki/Sphere

Luc Courchesne : The Visitor - Living by Numbers (2001).
http://www.din.umontreal.ca/courchesne/

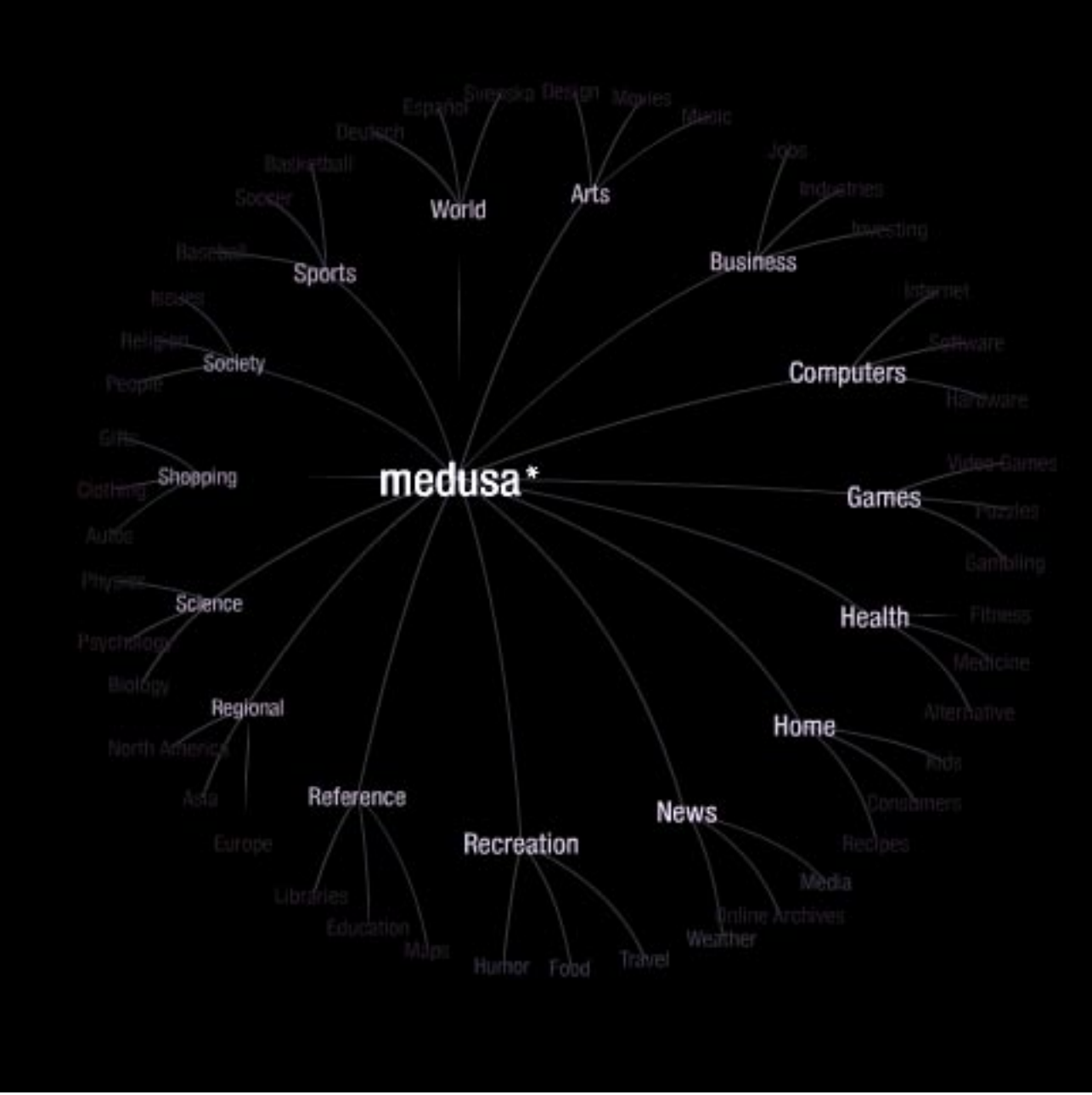


Luc Courchesne : Panoscope 360° (2005).
http://www.panoscope360.com/

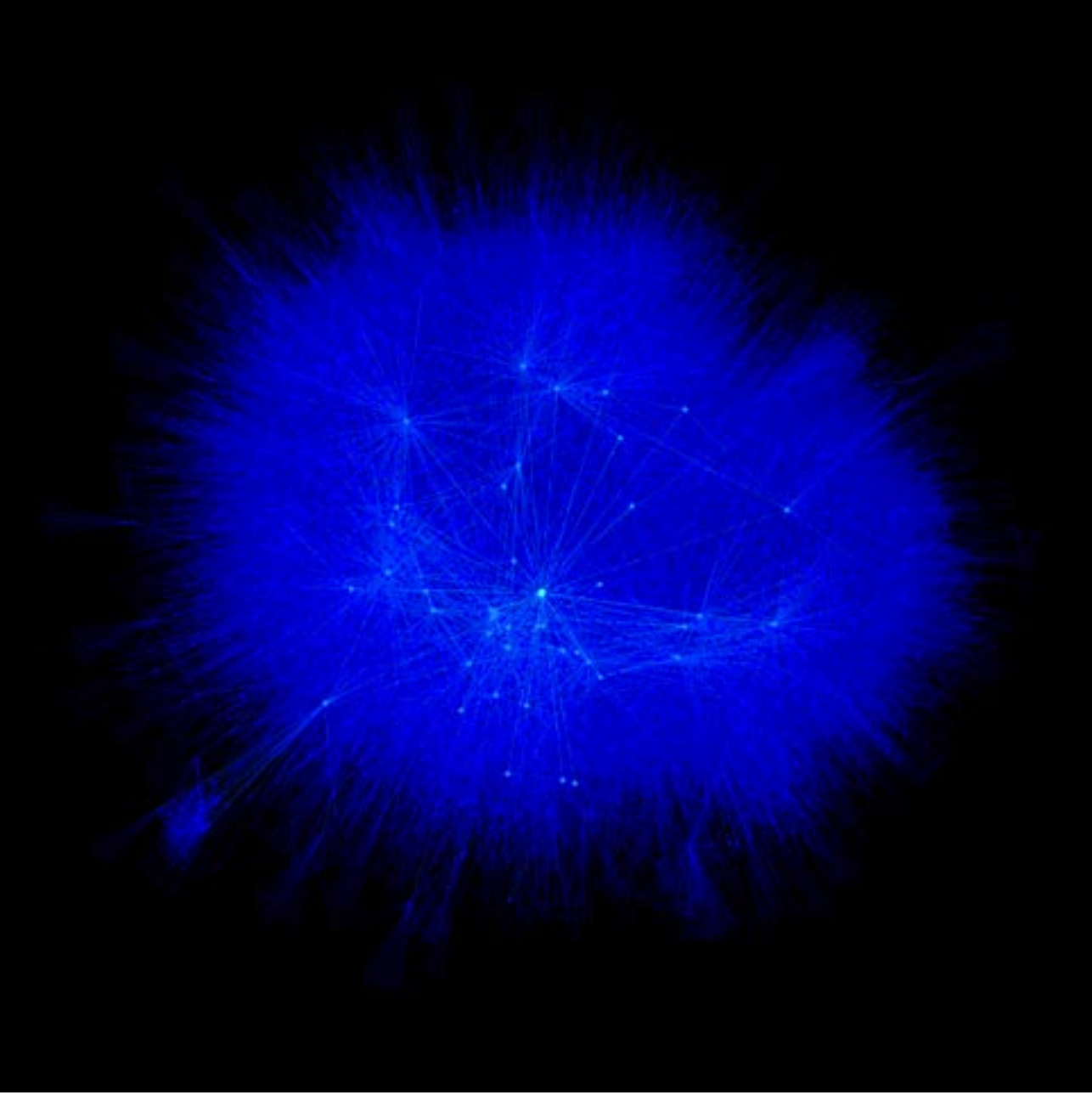
Hiroo Iwata : Floating Eye (2003).
http://www.aec.at/



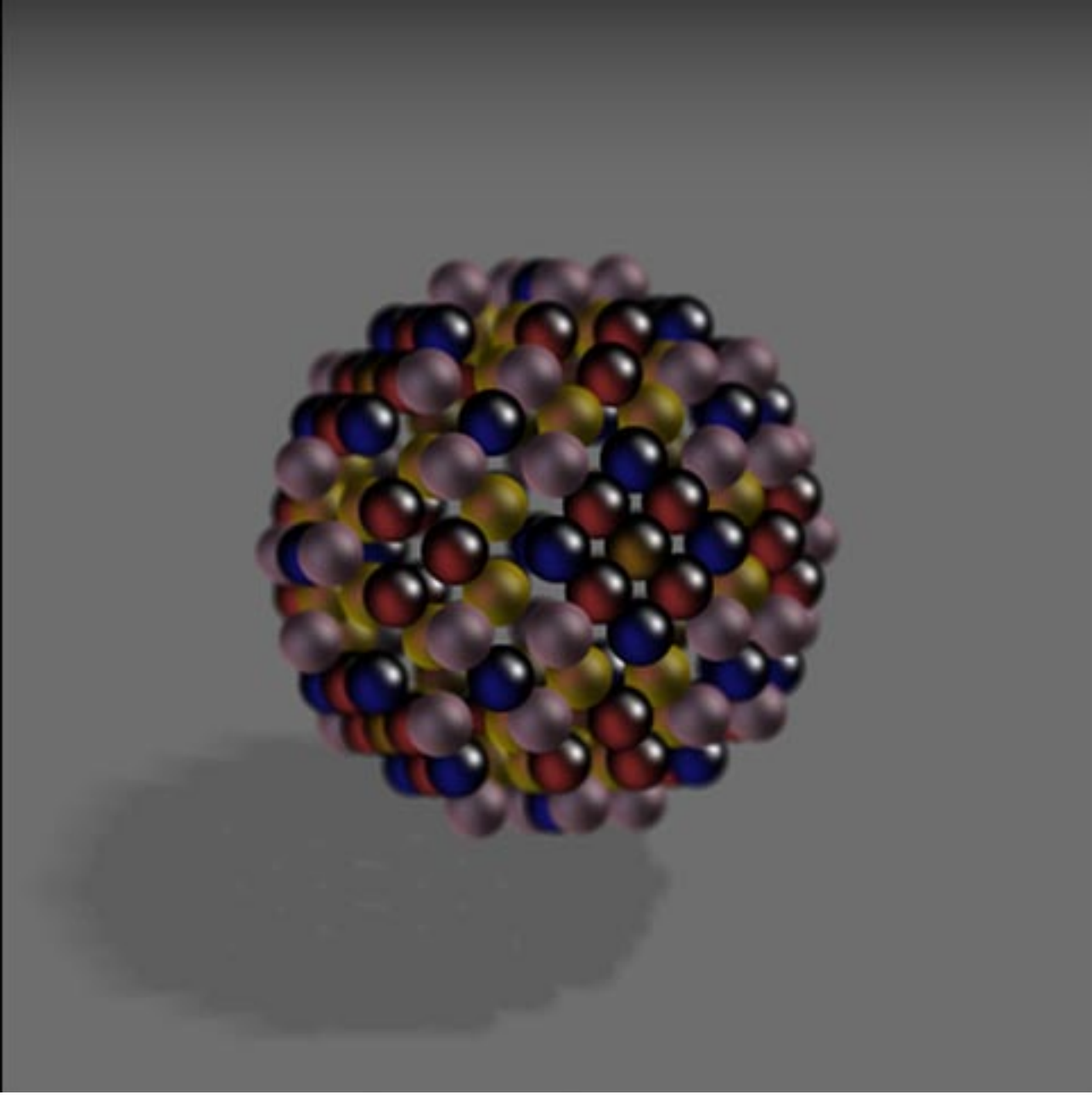
Benjamin Fry : Valence (2001),
représentation explorant les structures et les relations
au sein de grands ensembles d'informations.
http://acg.media.mit.edu/people/fry/valence/



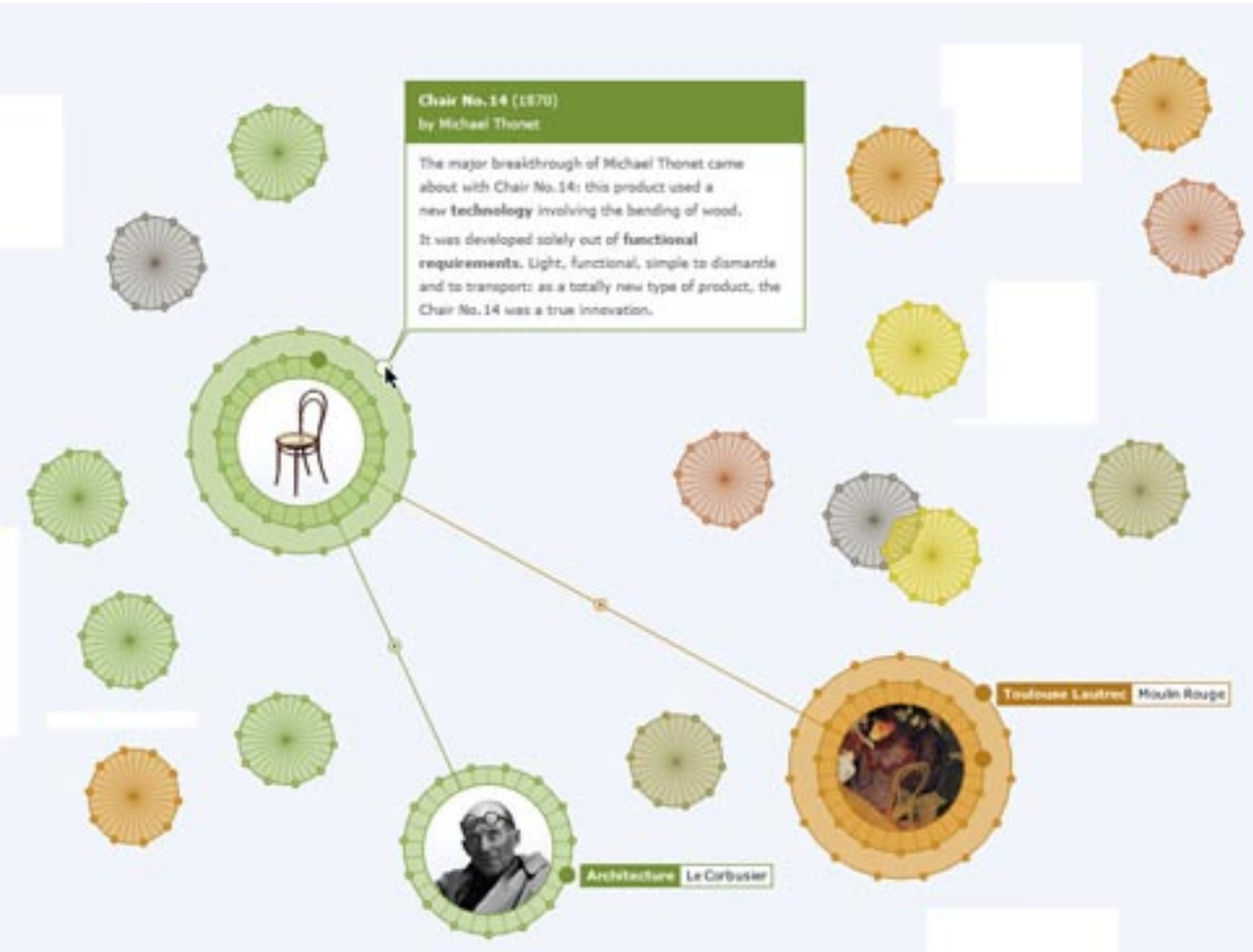
Chris Pollak, Colin Hughes et Stefan Brandys : Medusa (2000), plateforme communautaire expérimentale de visualisation de données.
http://www.chrispollak.com/medusa/medusa_demo/medusa.html



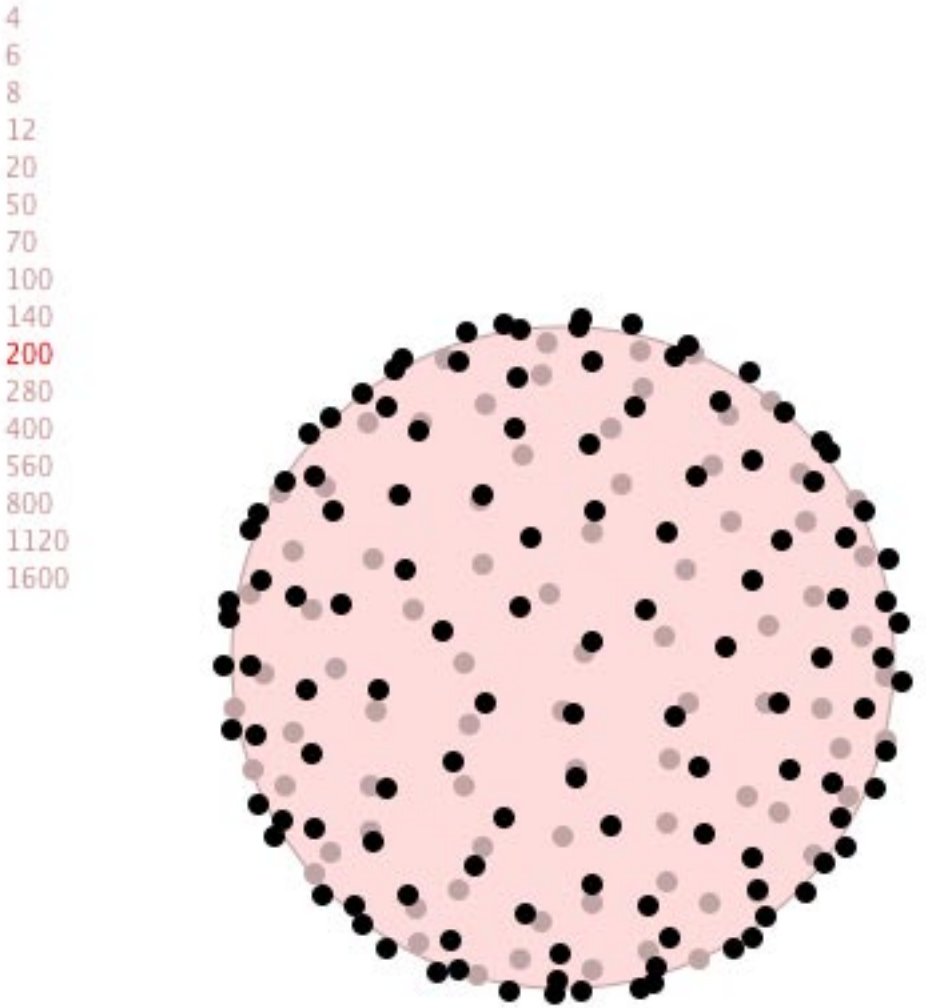
Jeffrey Heer : Prefuse (2005),
outil libre de visualisation de données.
http://prefuse.sourceforge.net/
http://www.cs.berkeley.edu/%7Ejheer/socialnet/



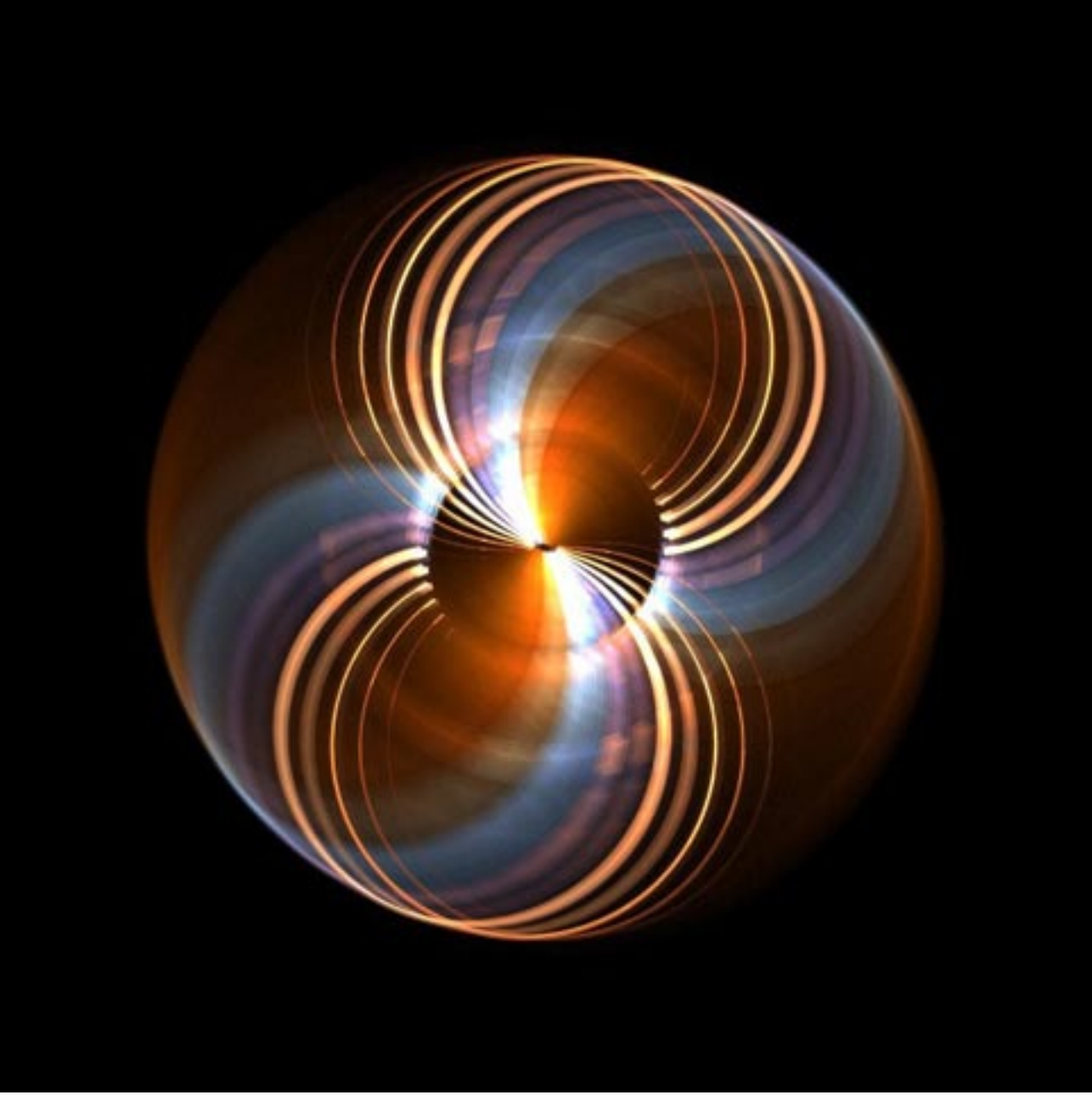
Ken Perlin : Buckyballet (2001),
en hommage à Buckminster Fuller.
http://mrl.nyu.edu/~perlin/experiments/bucky/



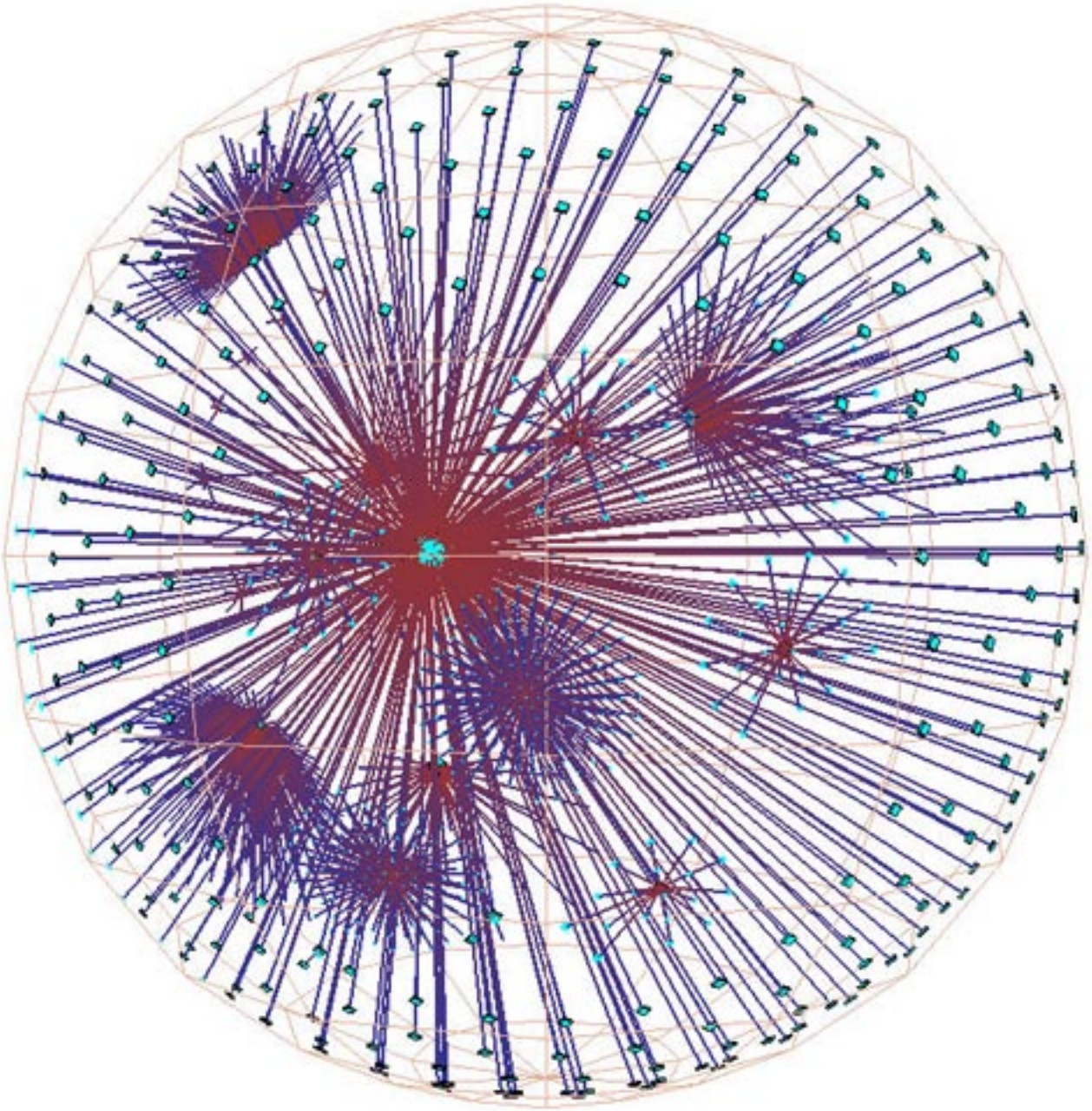
Carolin Horn & Florian Jenett : Jellyfish (2005),
visualisation d'une encyclopédie.
http://www.carohorn.de/jellyfish/



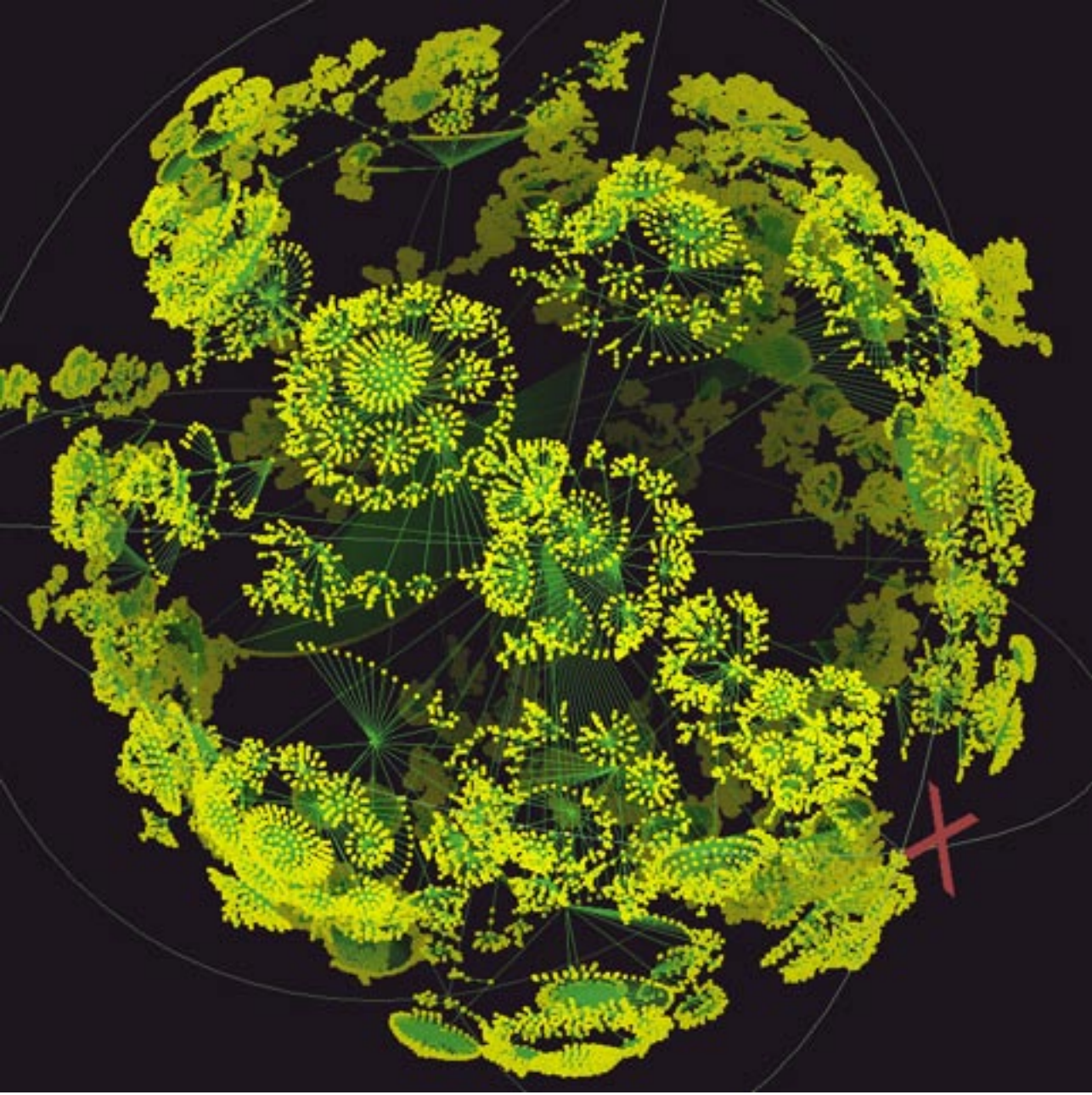
Ken Perlin : Little creatures in a little world (2001).
http://mrl.nyu.edu/~perlin/experiments/repel/



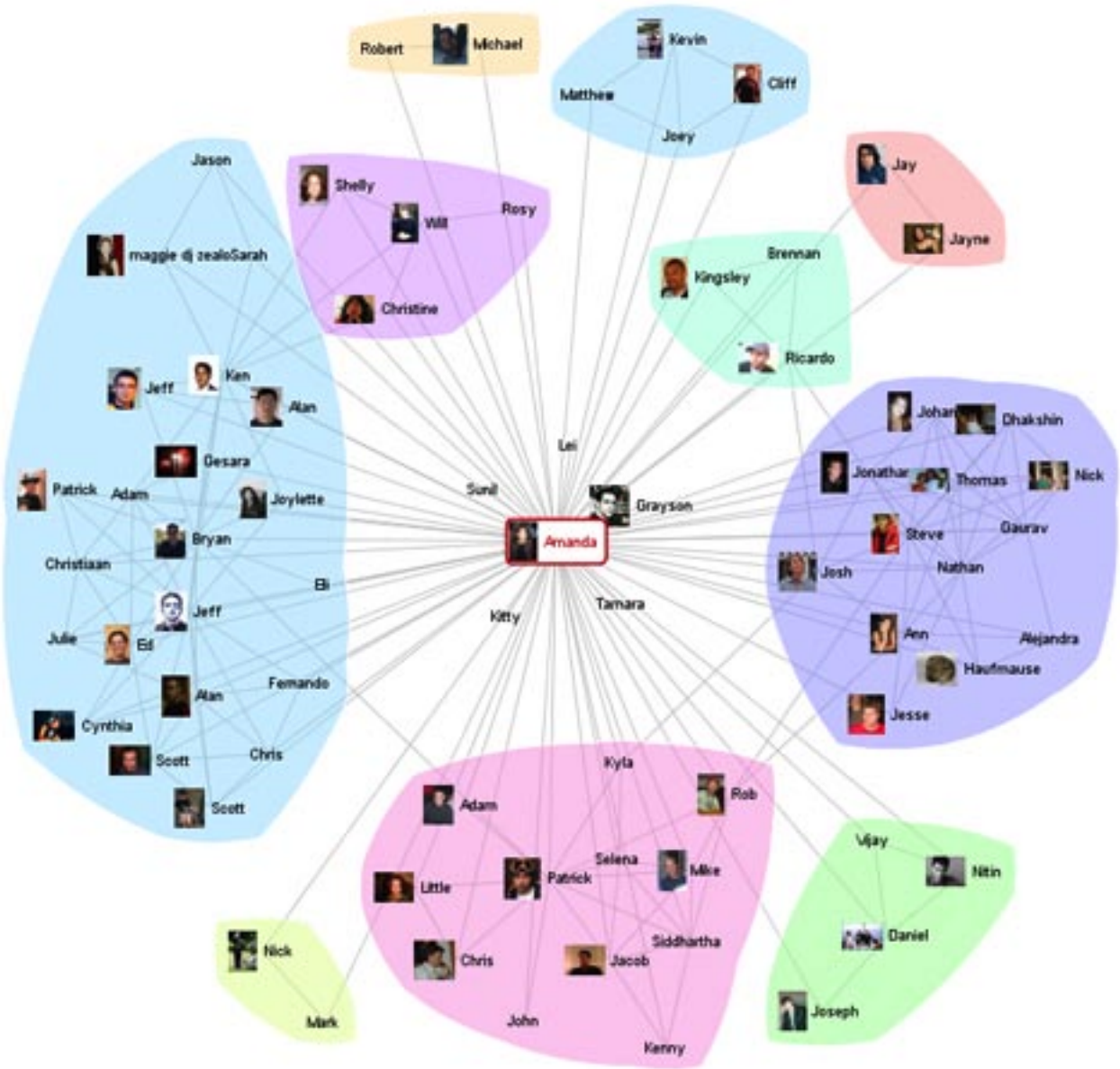
Scott Draves : Electric Sheep (2001),
écran de veille collaboratif et open-source.
http://electricsheep.org/
http://draves.org/



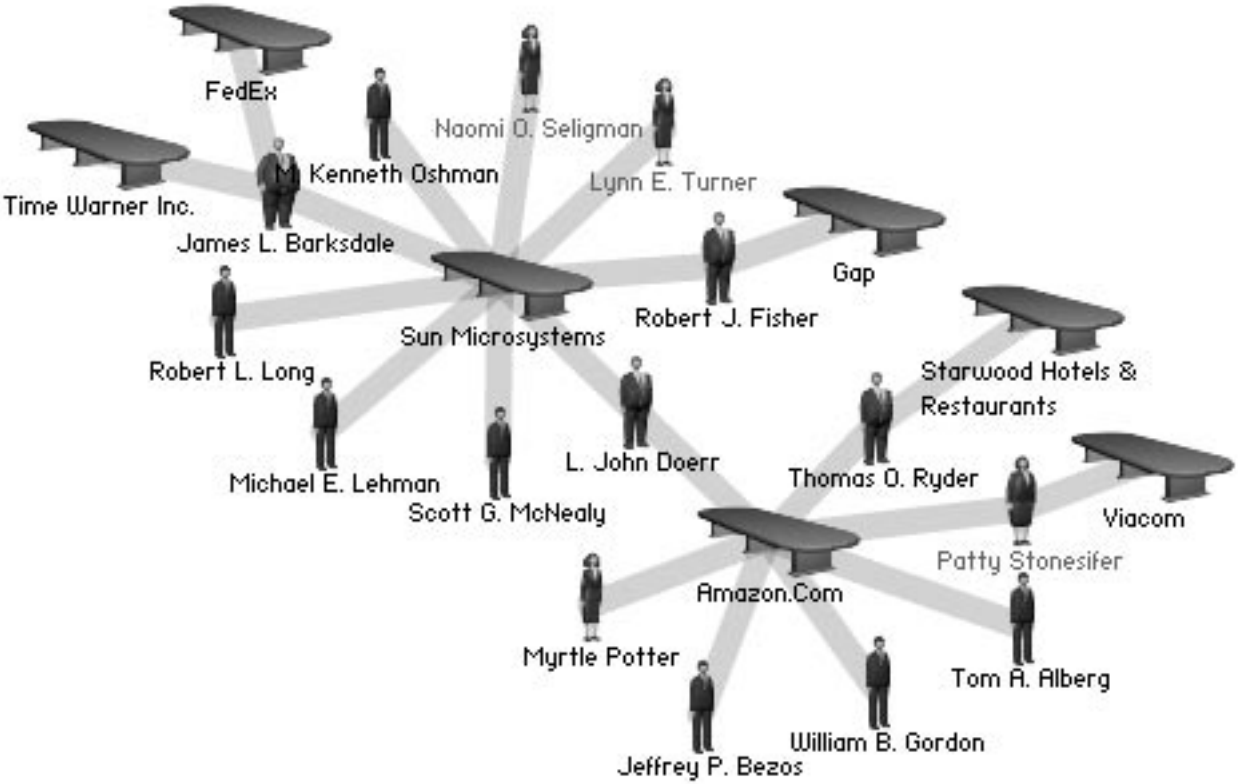
Daniel W. McRobb (CAIDA) : H3 (1999),
outil 3D de visualisation de données (C++ et OpenGL).
http://graphics.stanford.edu/papers/munzner_thesis/html/node8.html



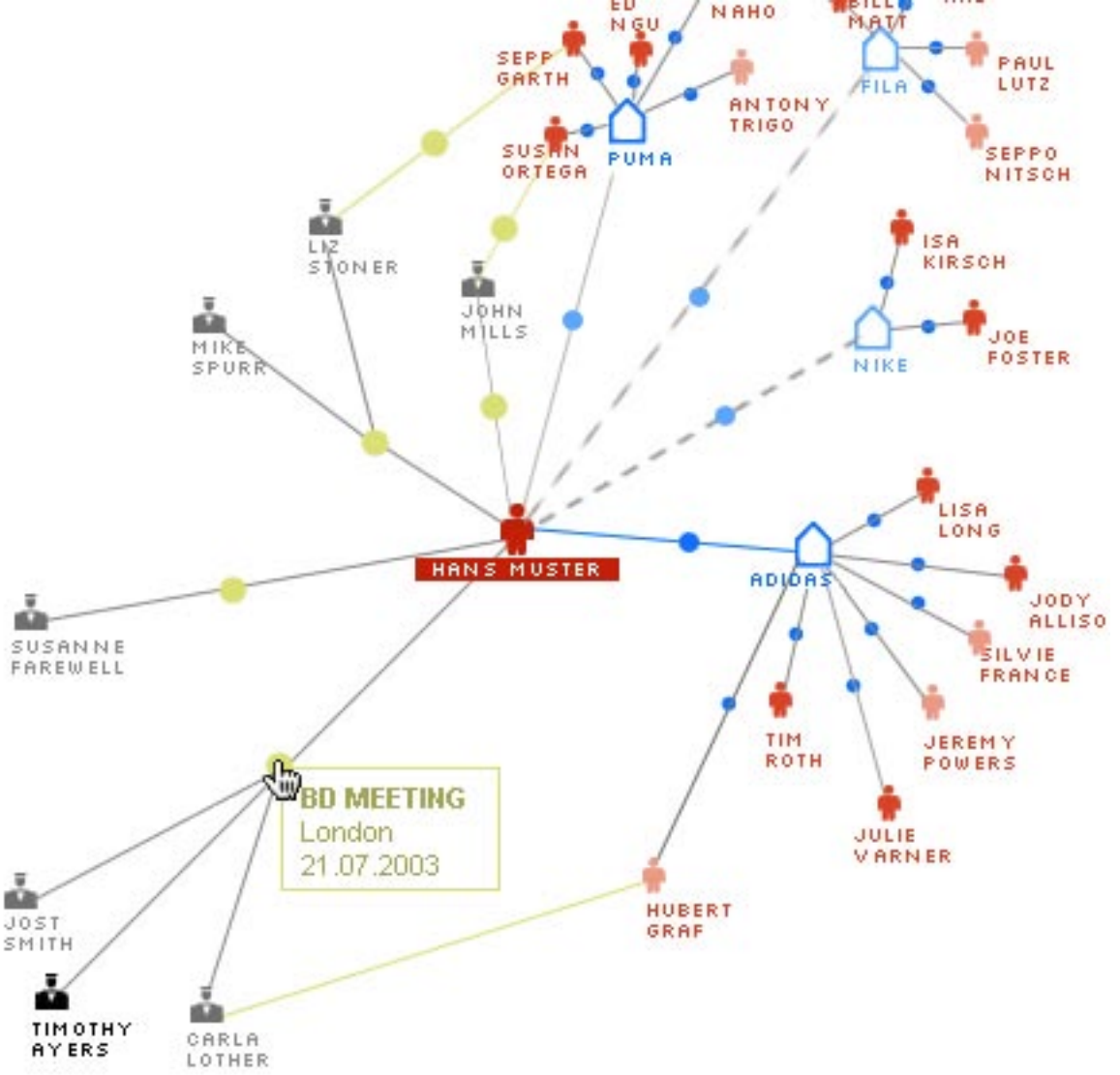
Young Hyun (CAIDA) : Walrus (2001),
outils de visualisation de données.
http://www.caida.org/tools/visualization/walrus/gallery1/



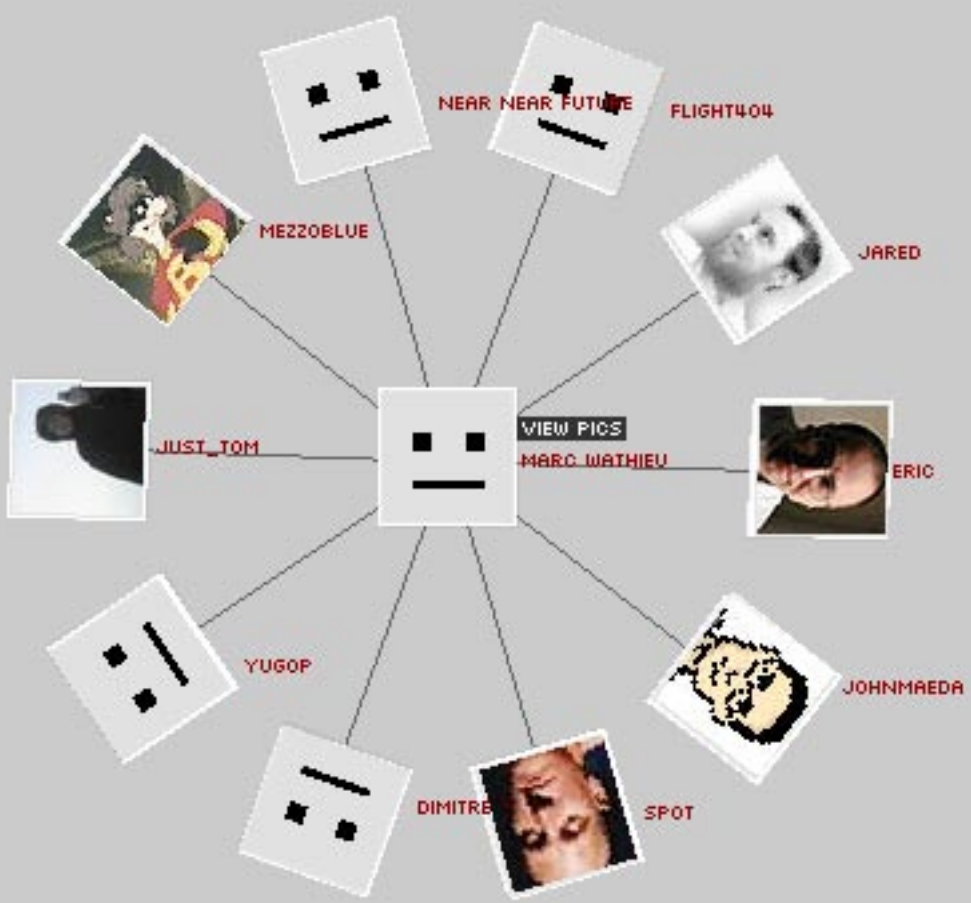
Jeffrey Heer & Danah Boyd : Vizster,
visualisation de réseaux.
http://jheer.org/vizster/



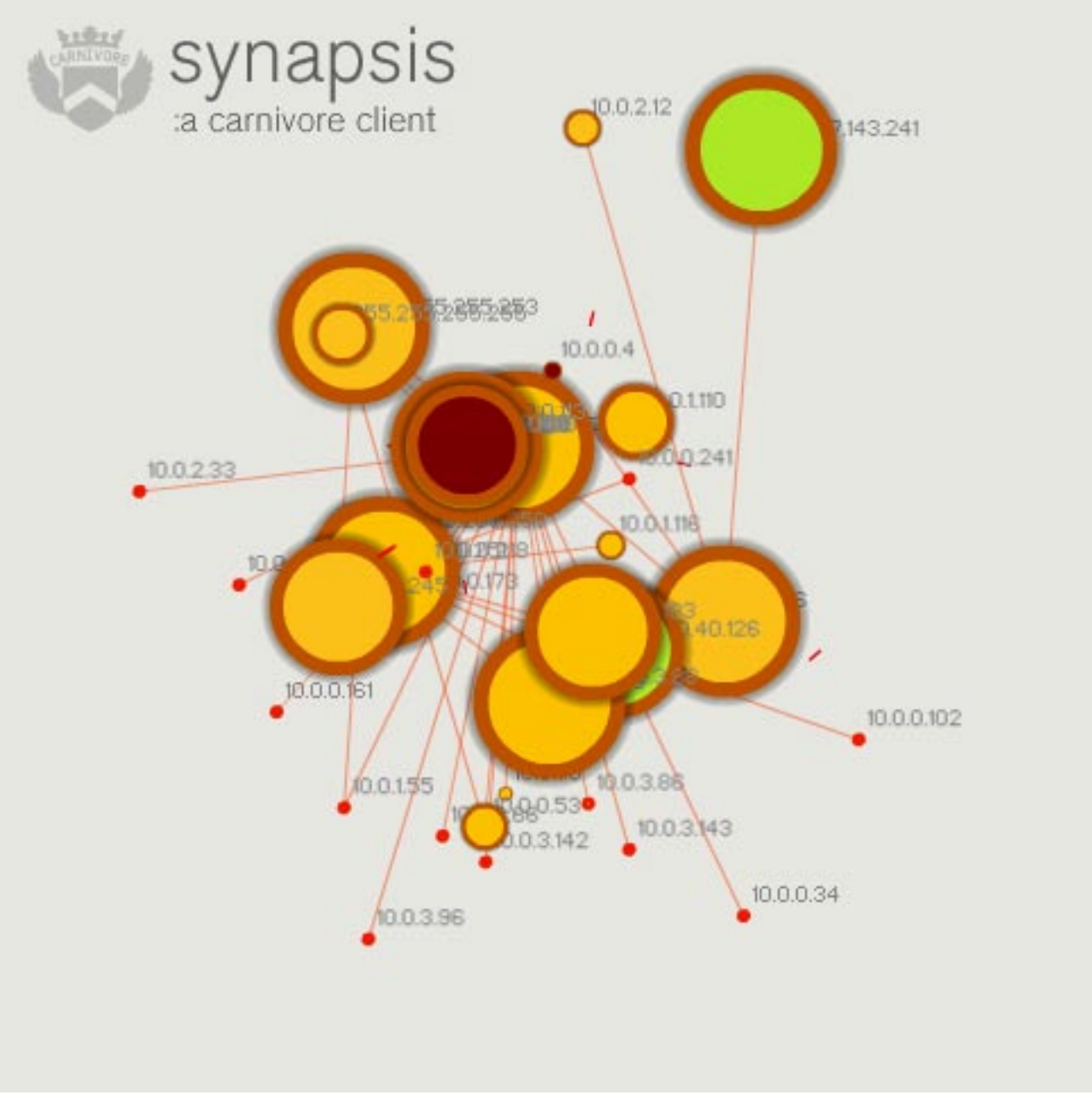
Josh On : They rule (2001),
moteur de recherche sur les dirigeants de grandes entreprises.
http://www.theyrule.net/



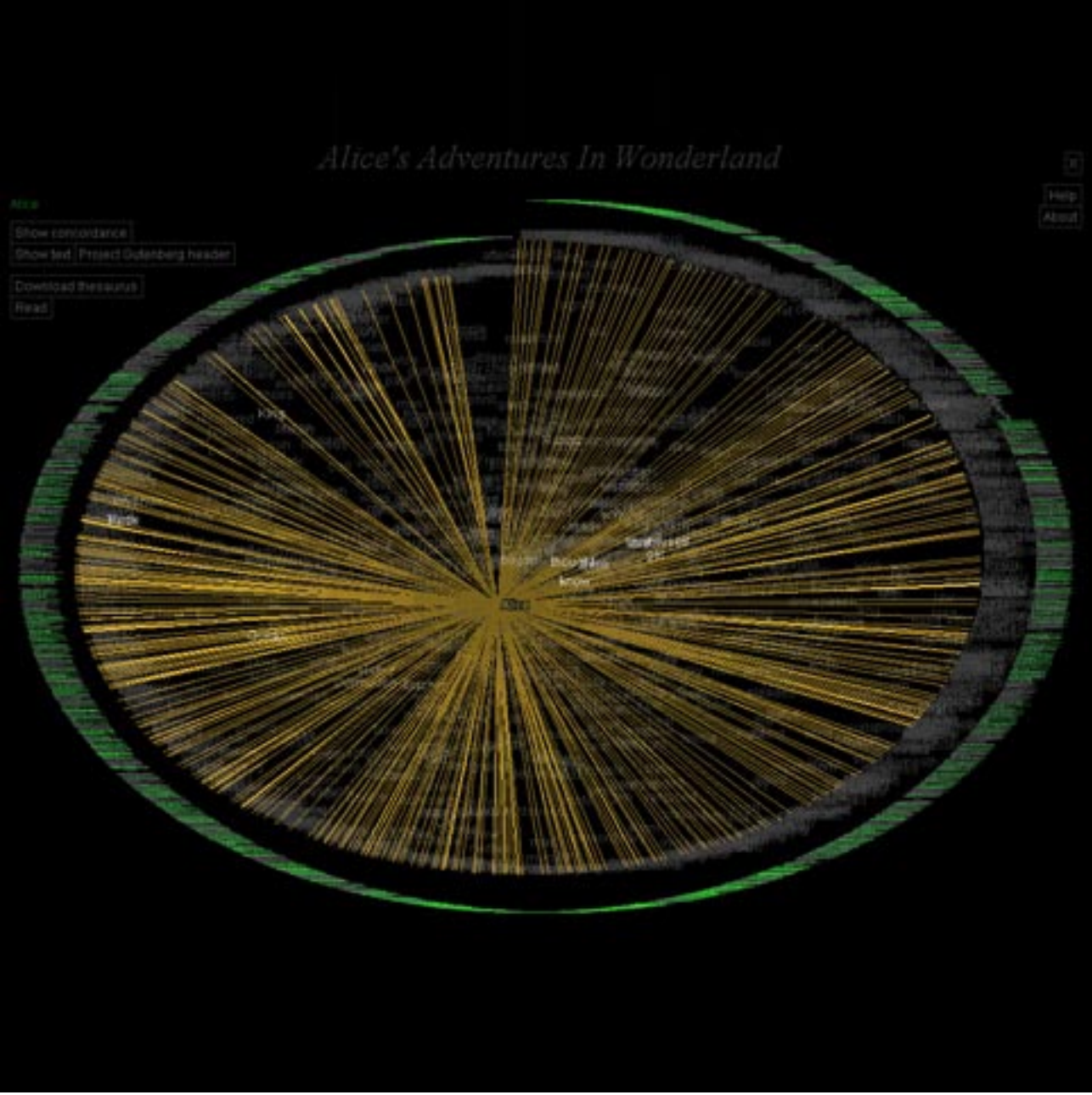
ThinkMap, Inc : ThinkMap SDK (Software Development Kit), plateforme configurable de visualisation de données complexes.
http://www.thinkmap.com/



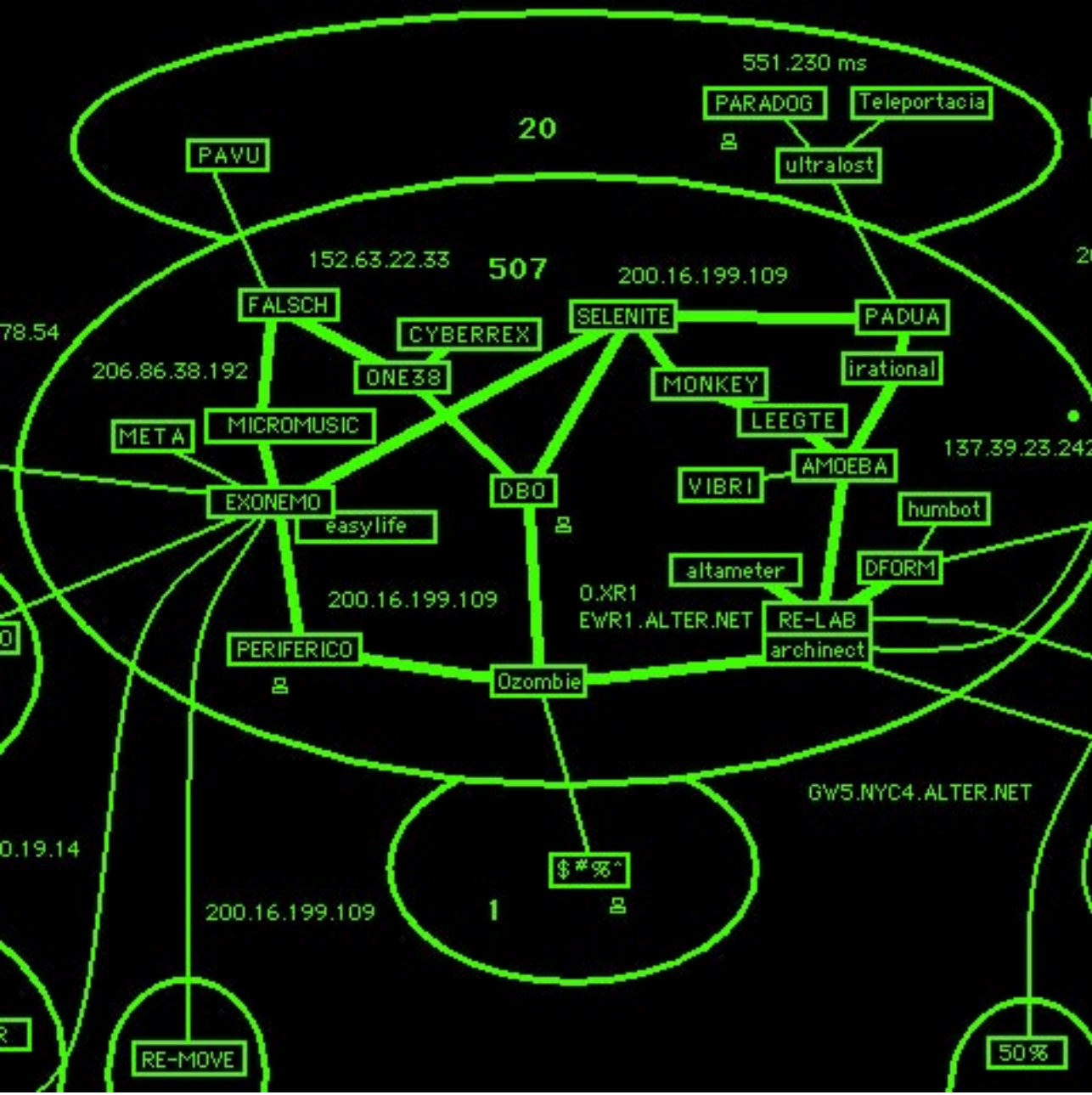
Marcos Weskamp : Flickr Graph (2005),
visualisation de communautés Flickr.
http://www.marumushi.com/apps/flickrgraph/



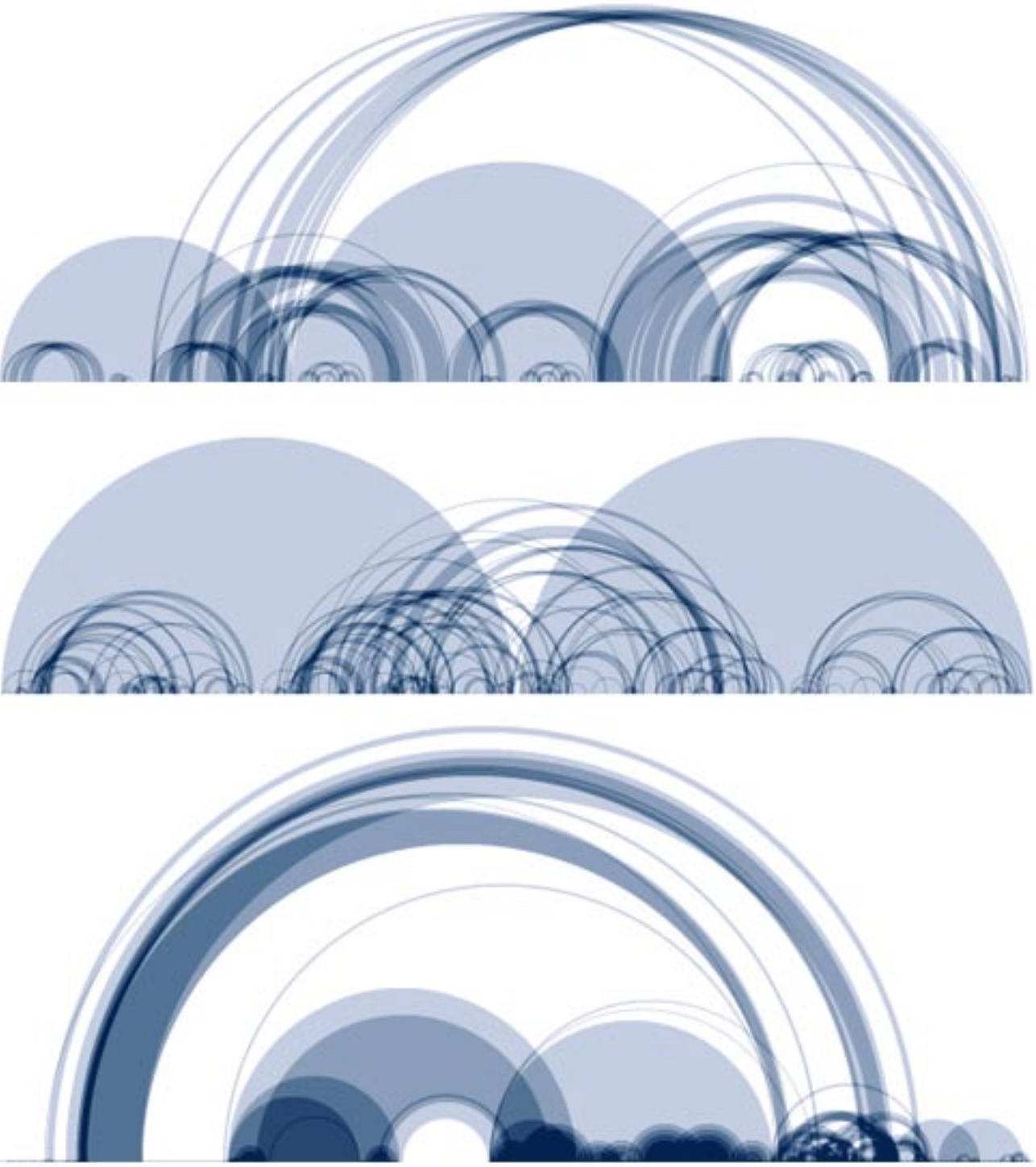
Marcos Weskamp : Synapsis (2003),
visualisation de certains trafics sur le réseau.
http://www.marumushi.com/



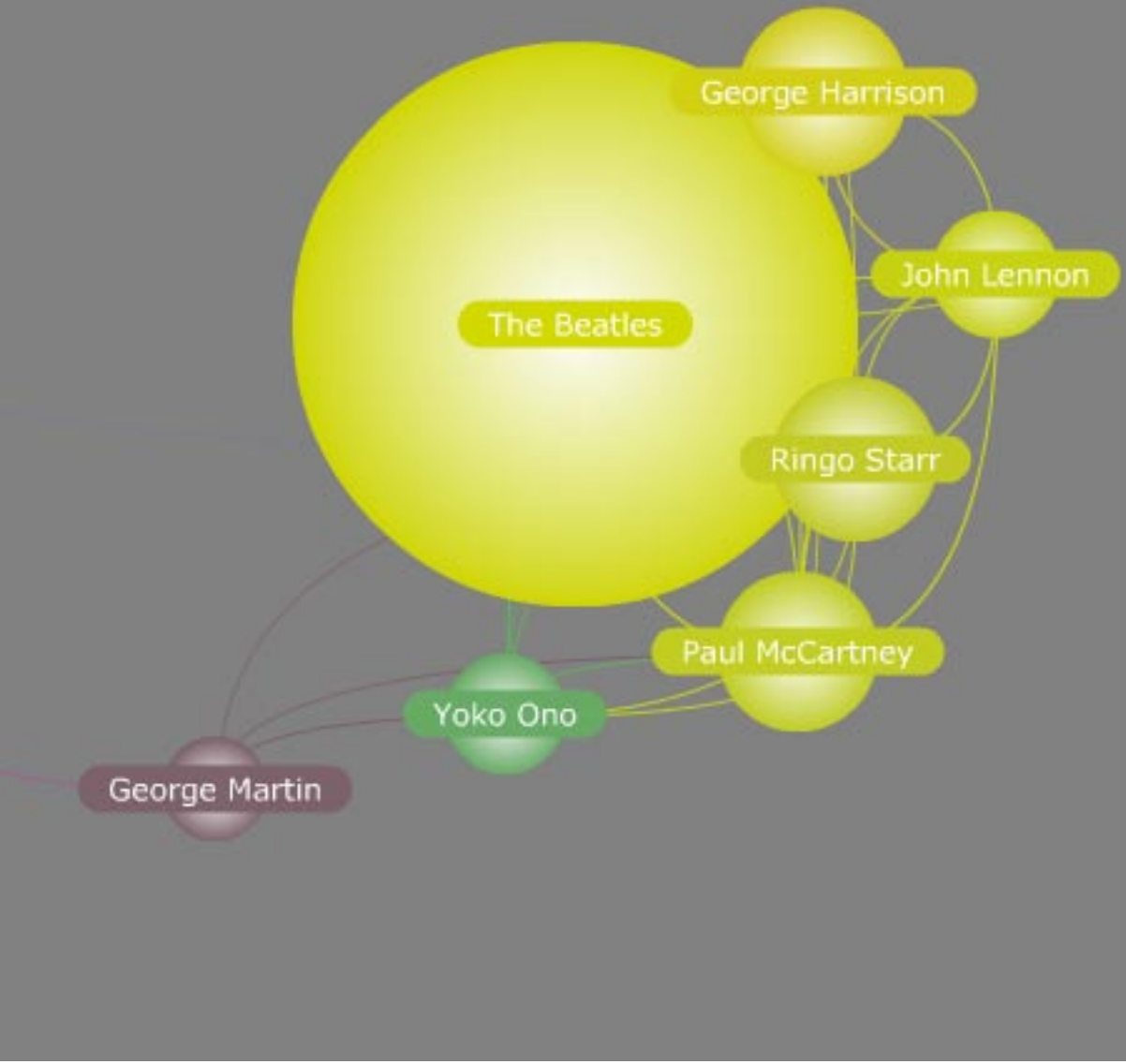
W. Bradford Paley : TextArc (2002),
représentation visuelle du contenu d'un texte.
http://www.textarc.org/



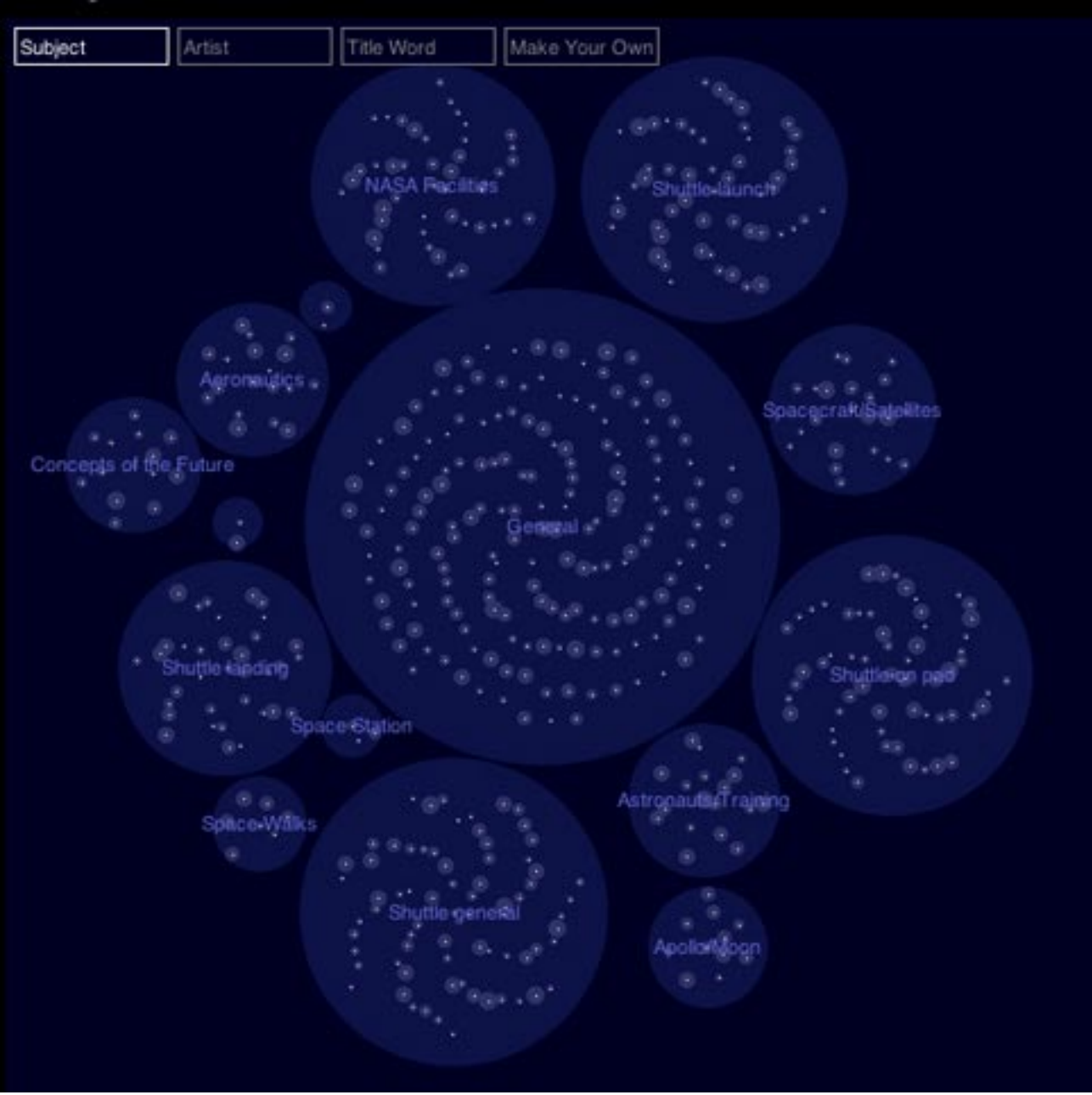
Jodi : map.jodi.org (1997),
visualisation d'un réseau.
http://map.jodi.org/



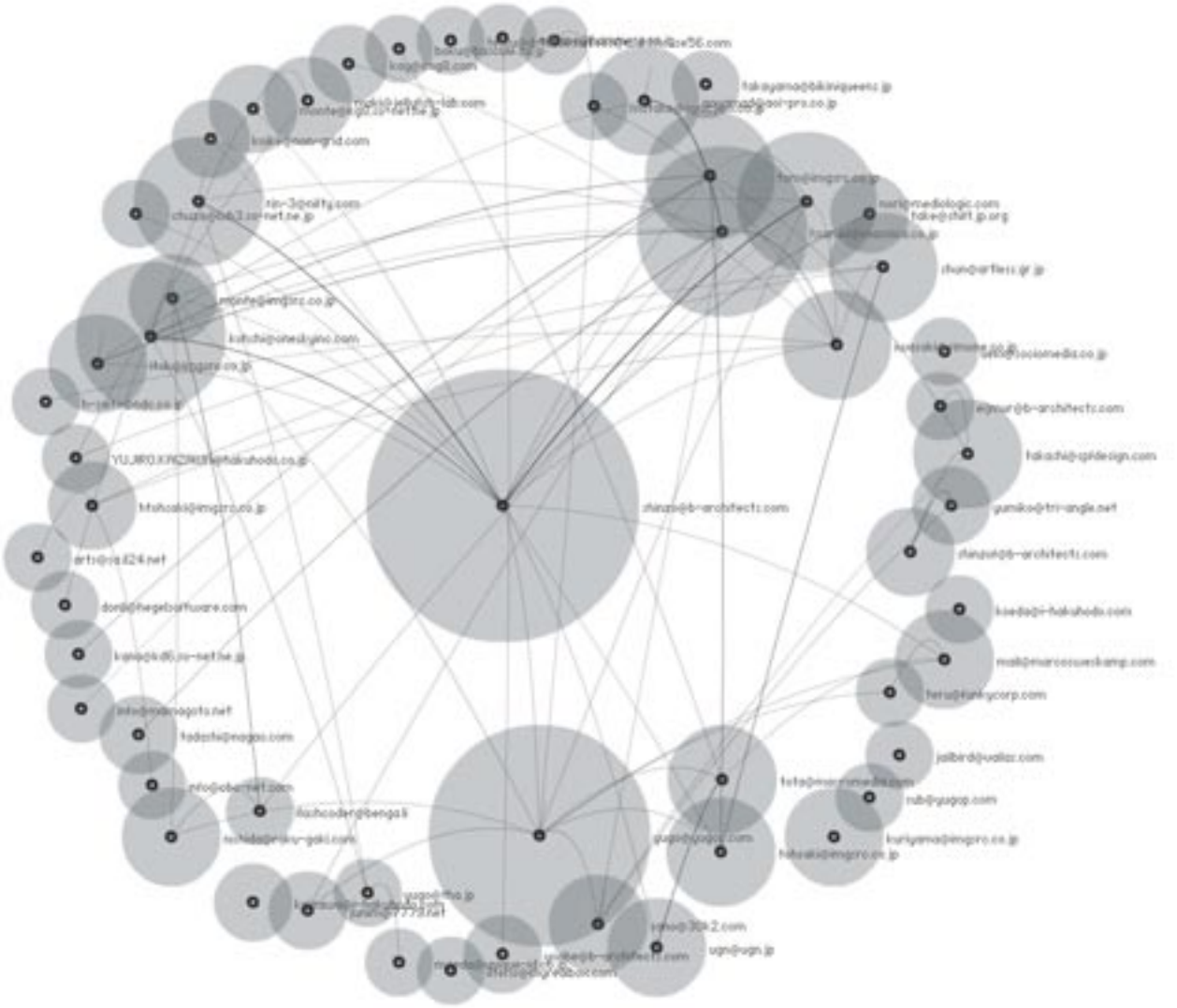
Martin Wattenberg : Shape of Song (2001),
visualisation de structures et motifs au sein d'oeuvres musicales.
http://www.turbulence.org/Works/song/



Frederic Vavrille : LivePlasma,
moteur de recherche musicale.
http://www.liveplasma.com/

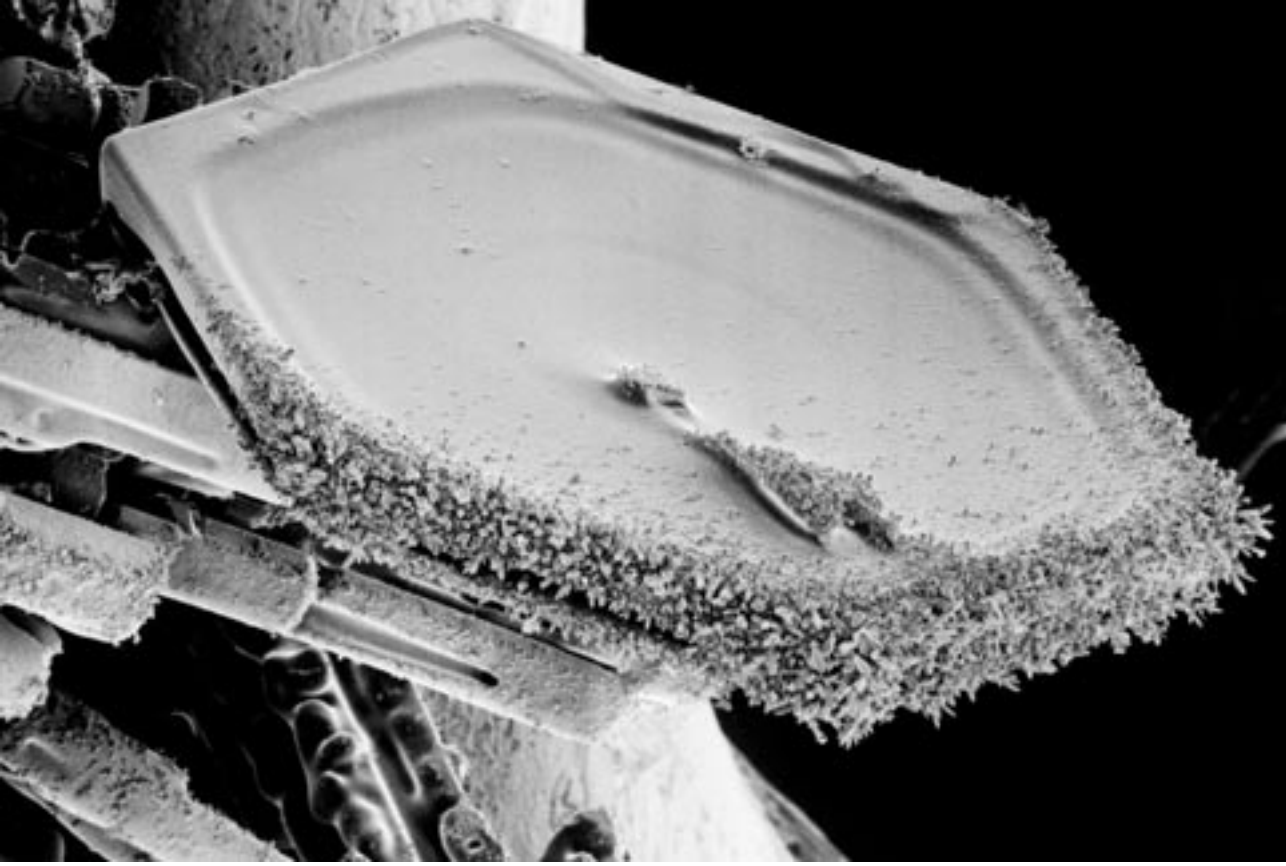


Martin Wattenberg : Copernica (2002),
moteur de recherche de l'iconothèque de la NASA.
http://www.hq.nasa.gov/copernica/

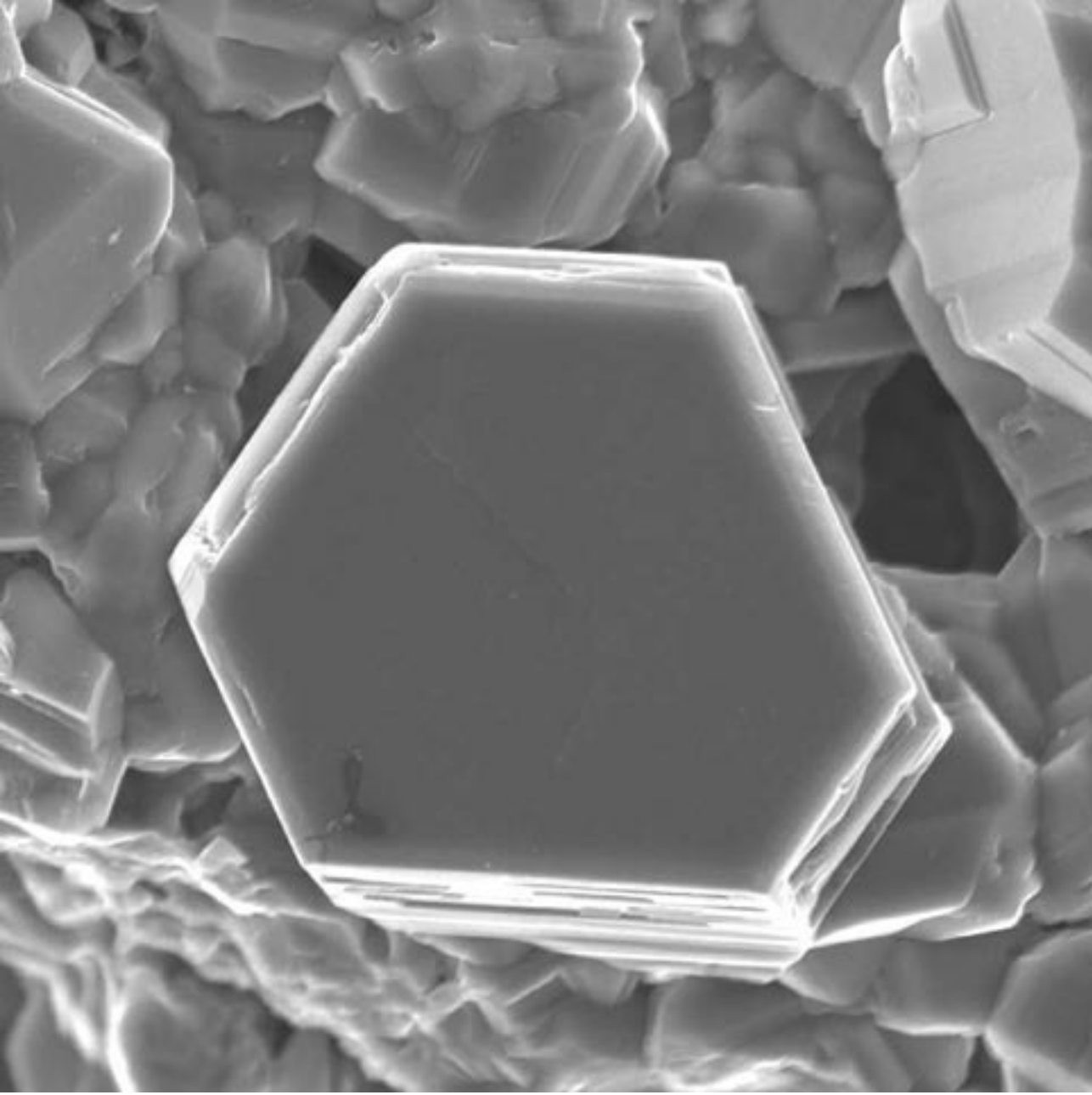


Marcos Weskamp : SocialCircles,
visualisation de réseaux de mailing lists.
http://marumushi.com/apps/socialcircles/

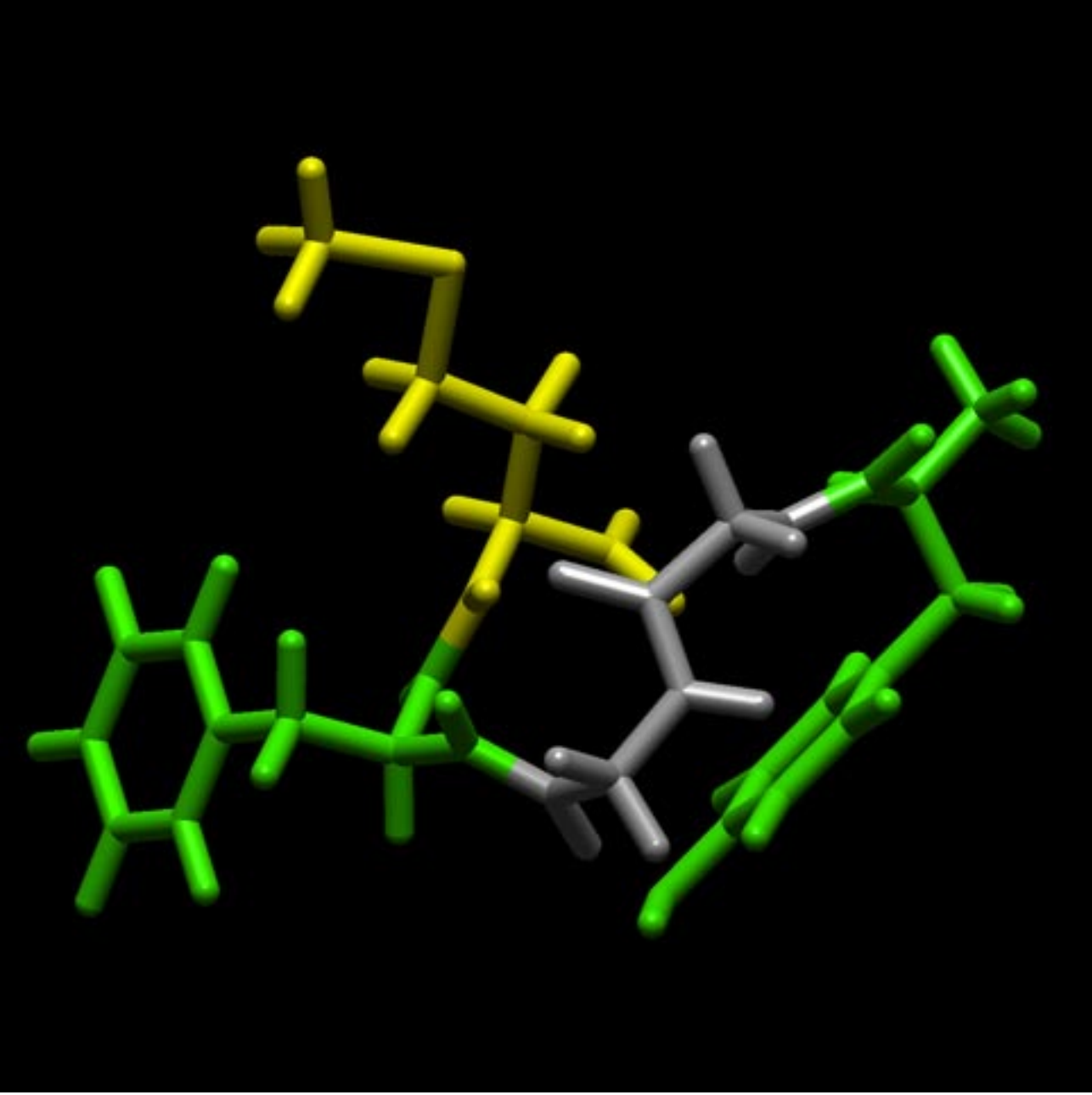
Polygones, polyèdres.



Flocon de neige.
http://www.lpsi.barc.usda.gov/emusnow/Selected/Select1.htm
http://www.snowcrystals.com/
http://en.wikipedia.org/wiki/Snow



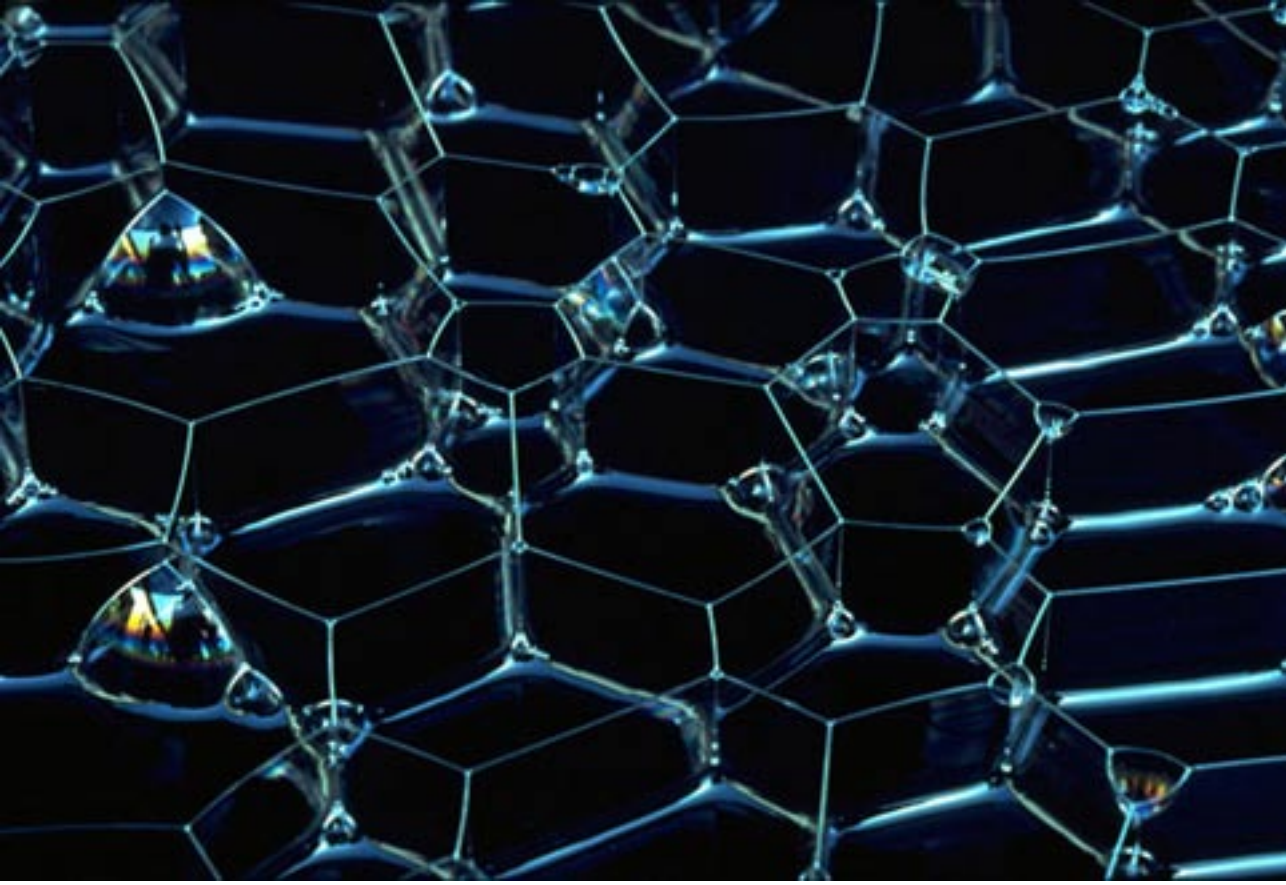
Nano-cristaux.
http://fb6www.upb.de/ag/ag-greulich/sic-porous.html



Enkephaline.
http://dasher.wustl.edu/ffe/pages/gallery.html

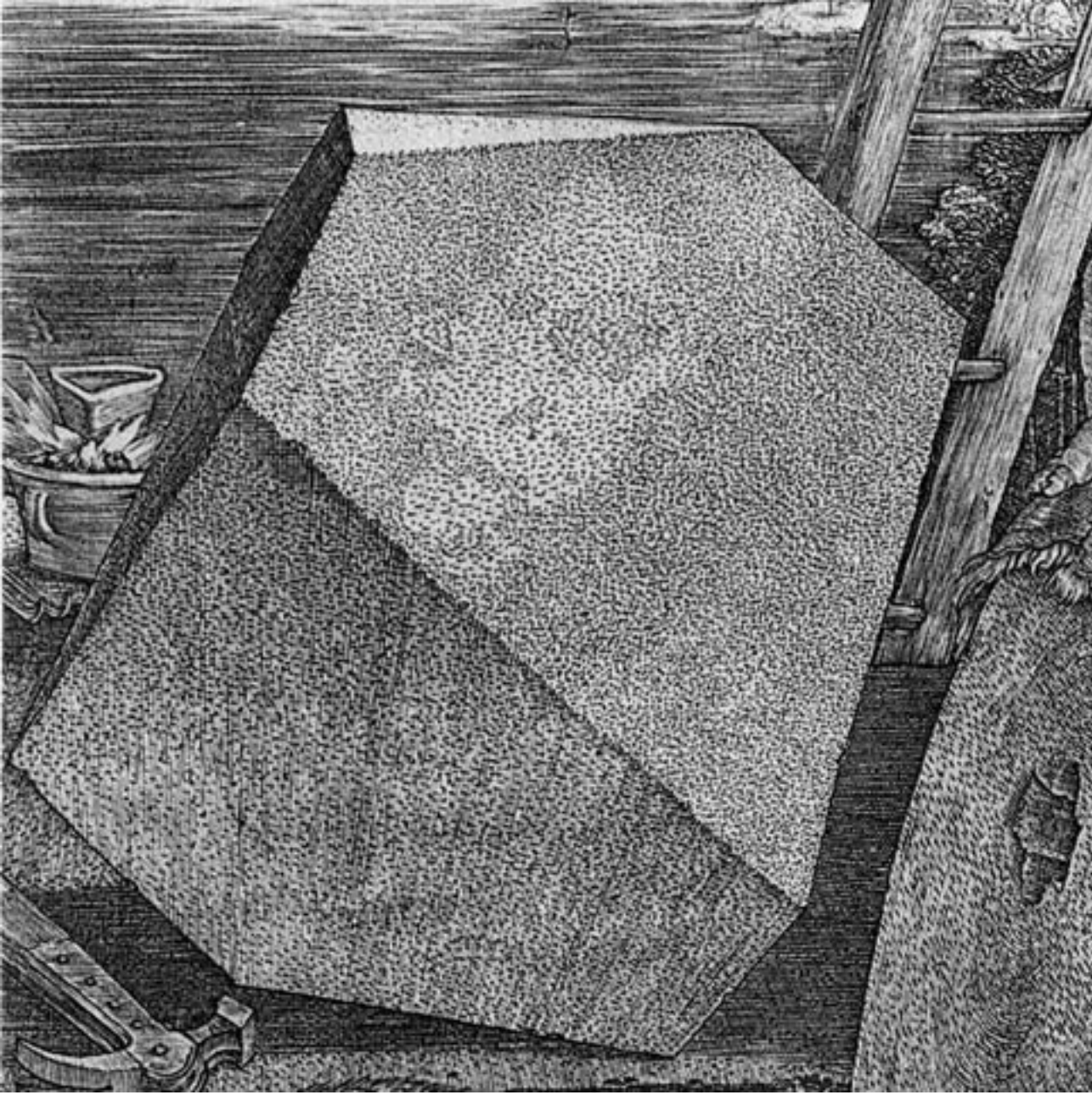
http://fr.wikipedia.org/wiki/Alvéole_d'abeille
http://hypo.ge-dip.etat-ge.ch/www/math/html/node18.html

http://en.wikipedia.org/wiki/Dragon_fly



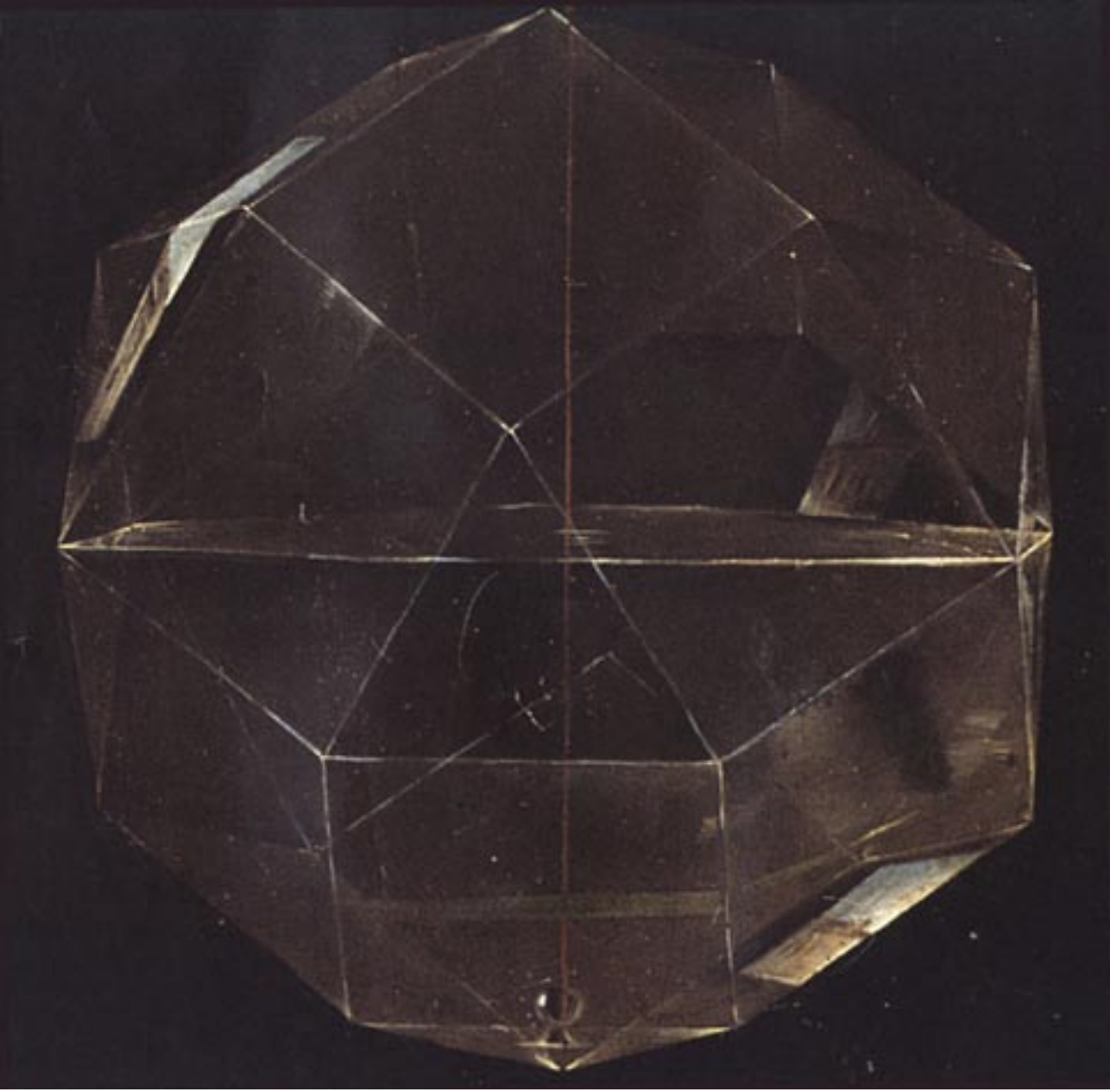
http://en.wikipedia.org/wiki/Soap_bubble


MELENCOLIA I

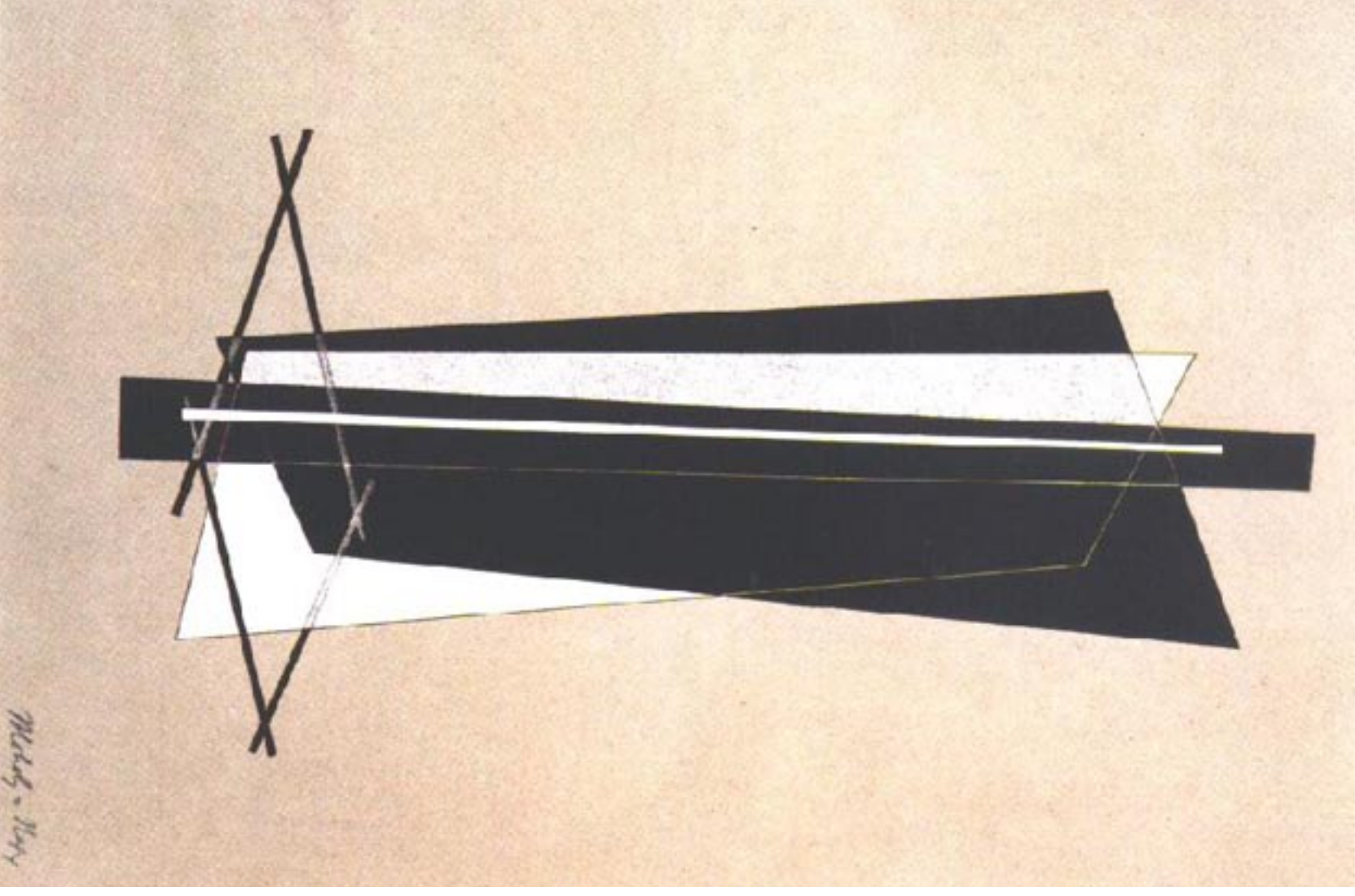


Albrecht Dürer : Melancolia, détail (1514).
http://en.wikipedia.org/wiki/Albrecht_Dürer
http://www.georgehart.com/virtual-polyhedra/durer.html

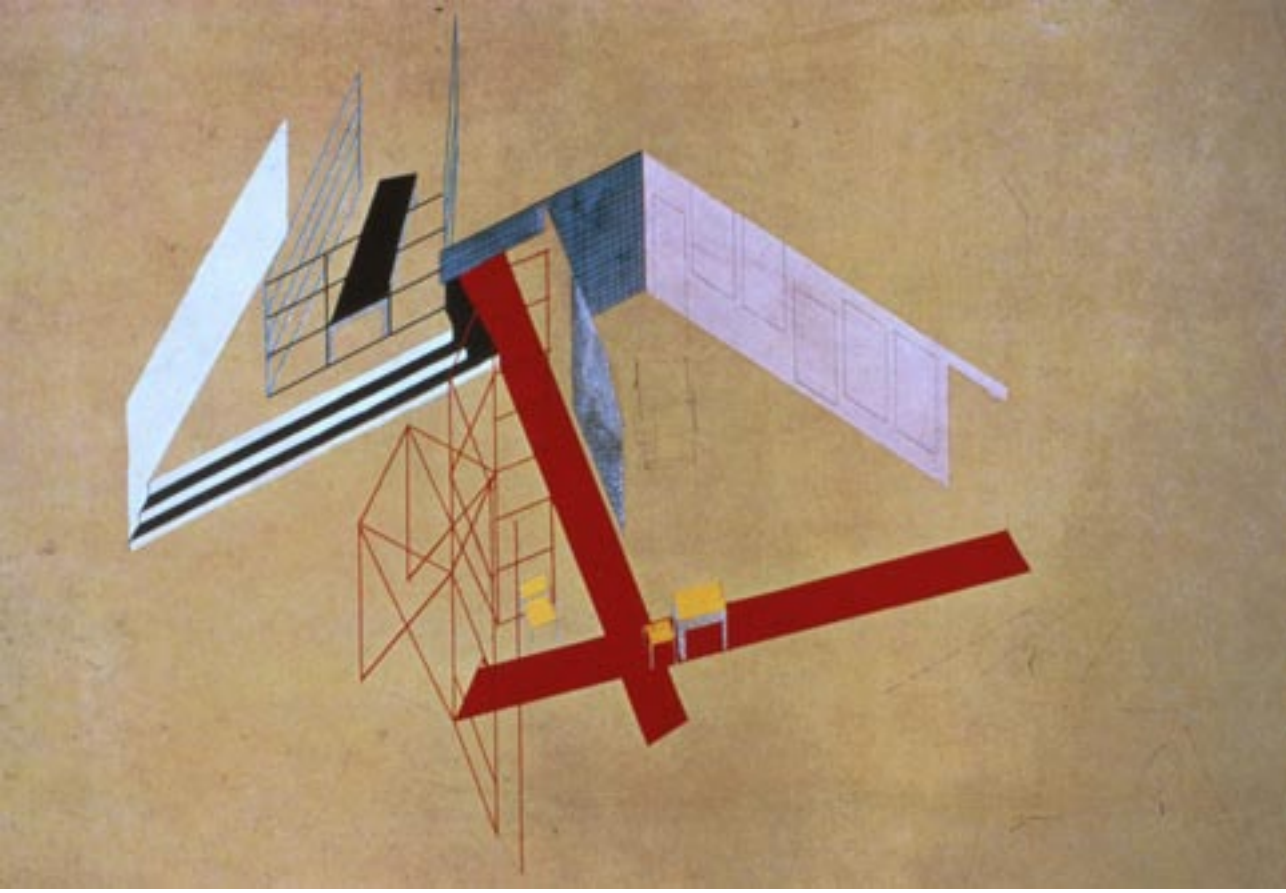
«Ritratto di Frà Luca Pacioli»
attribué à Jacopo de Barbari (1495)
http://cage.rug.ac.be/~hs/pacioli/pacioli.html
http://en.wikipedia.org/wiki/Luca_Pacioli



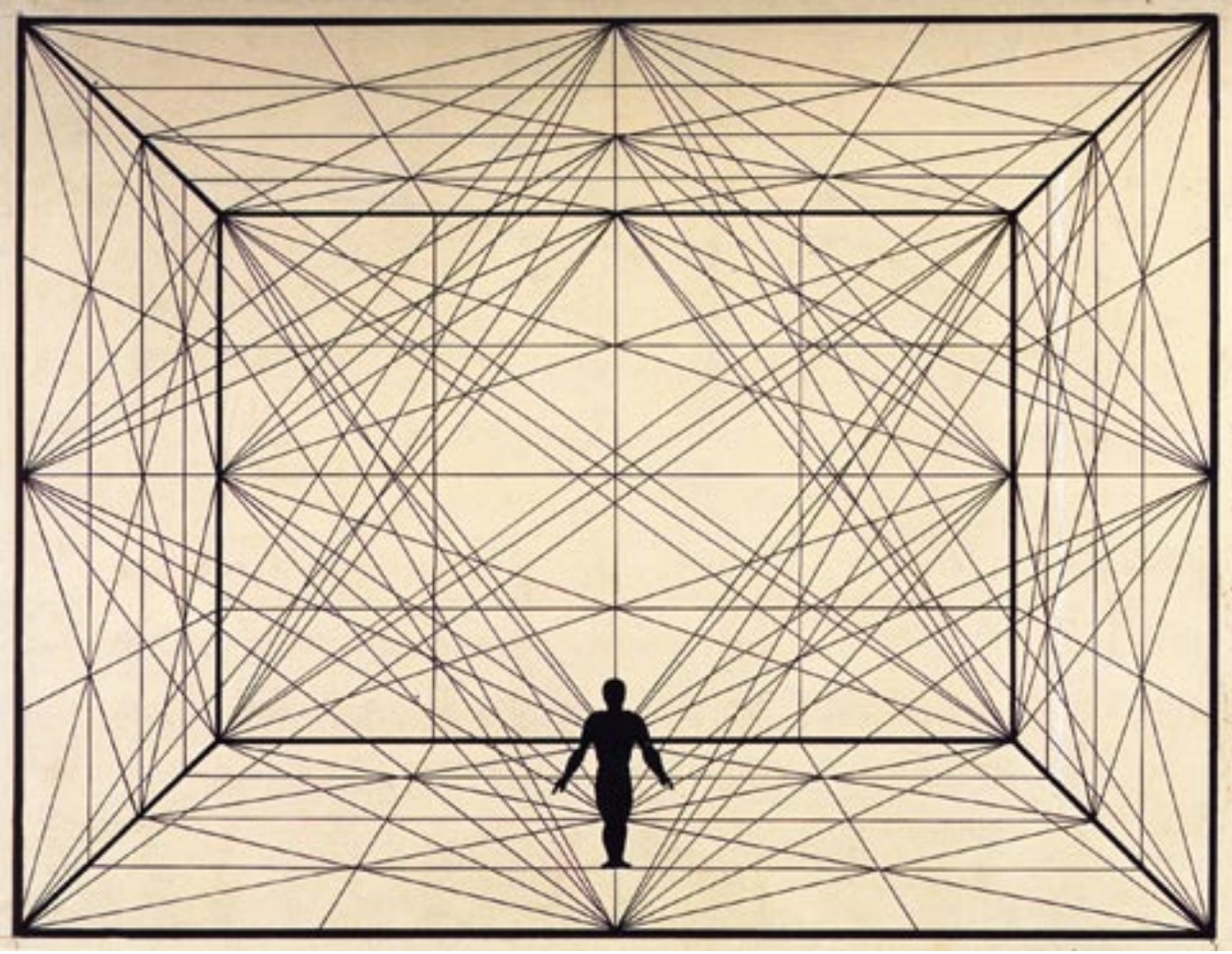
«Ritratto di Frà Luca Pacioli» (détail)
attribué à Jacopo de Barbari (1495)
http://cage.rug.ac.be/~hs/pacioli/pacioli.html
http://en.wikipedia.org/wiki/Luca_Pacioli



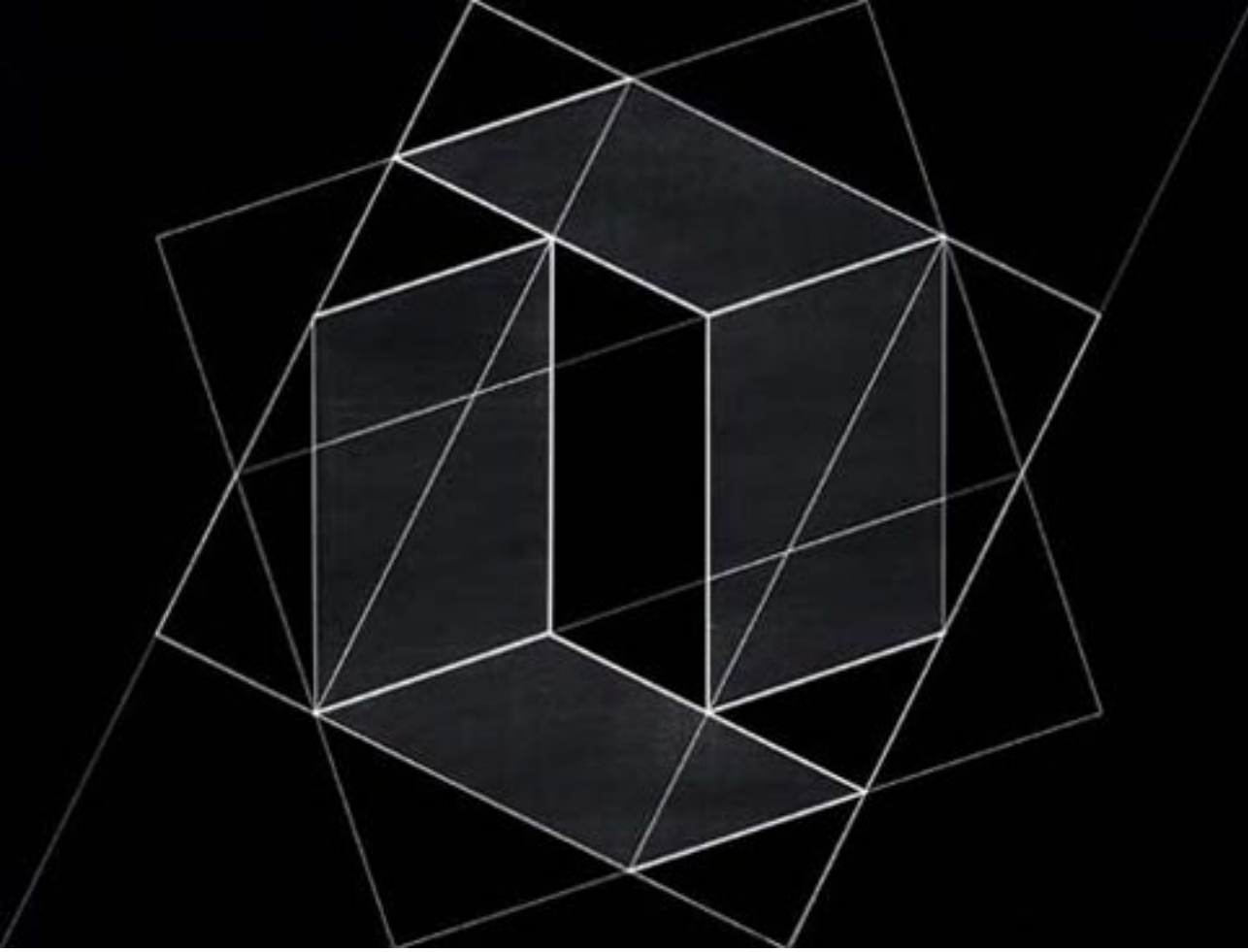
László Moholy-Nagy : Konstrukció (1923).
http://www.moholy-nagy.org/



László Moholy-Nagy : Construction (1923).
http://www.moholy-nagy.org/

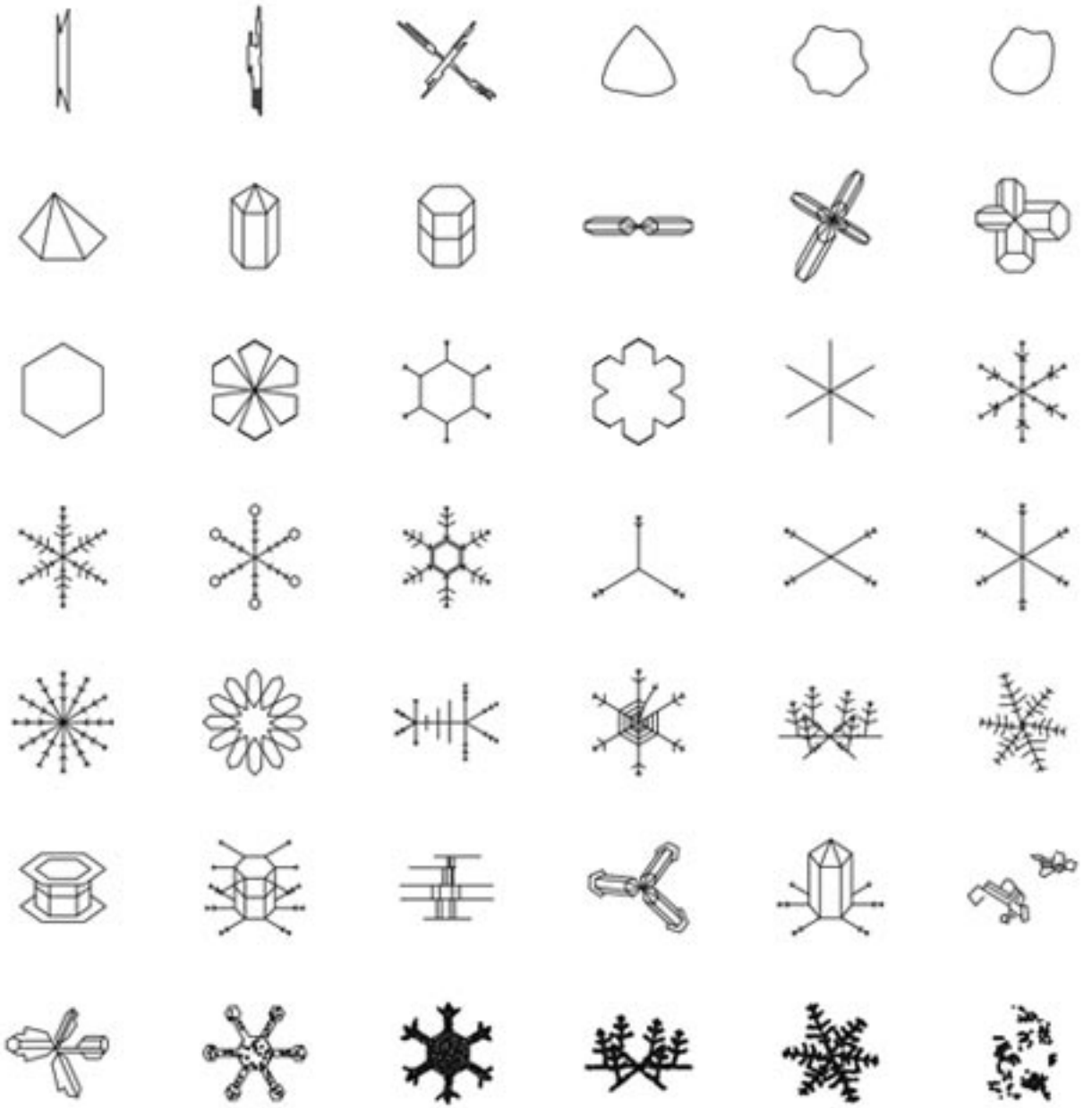


Oskar Schlemmer : Figur und raumlineatur (1924).



Josef Albers :
Structural Constellation, Transformation of a Scheme No.12 (1950).
http://www.tate.org.uk/modern/exhibitions/albersmoholy/

André Waterkeyn : l'Atomium (1958).
http://fr.wikipedia.org/wiki/Atomium



Carsten Nicolai : Snow noise (2002).
http://www.eigen-art.com/

Carsten Nicolai : Snow noise (2002).
http://www.eigen-art.com/

Olafur Eliasson : The inverted shadow (2003).
http://www.olafureliasson.net/

Olafur Eliasson : Quasi brick wall (2003).
http://www.olafureliasson.net/

Olafur Eliasson : Color Spectrum Kaleidoscope (2003).
http://www.olafureliasson.net/

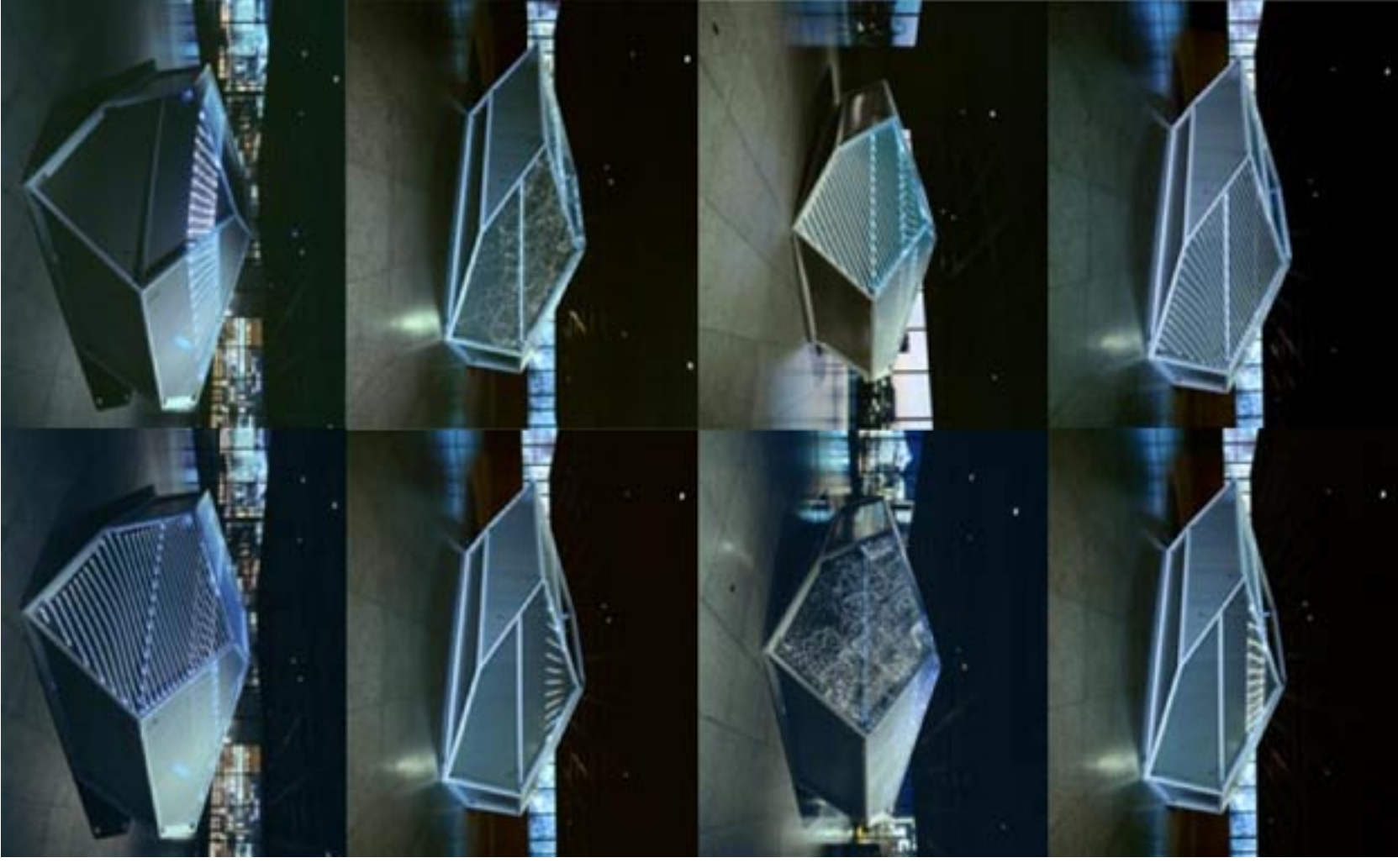
Olafur Eliasson : The antispe (2003).
http://www.olafureliasson.net/

Carsten Nicolai : modular re.strukt (2003),
modules, porcelaine.
http://www.paolocurti.com/nicolai2/nicolai.htm

Carsten Nicolai : modular re.strukt (2003),
modules, porcelaine.
http://www.paolocurti.com/nicolai2/nicolai.htm

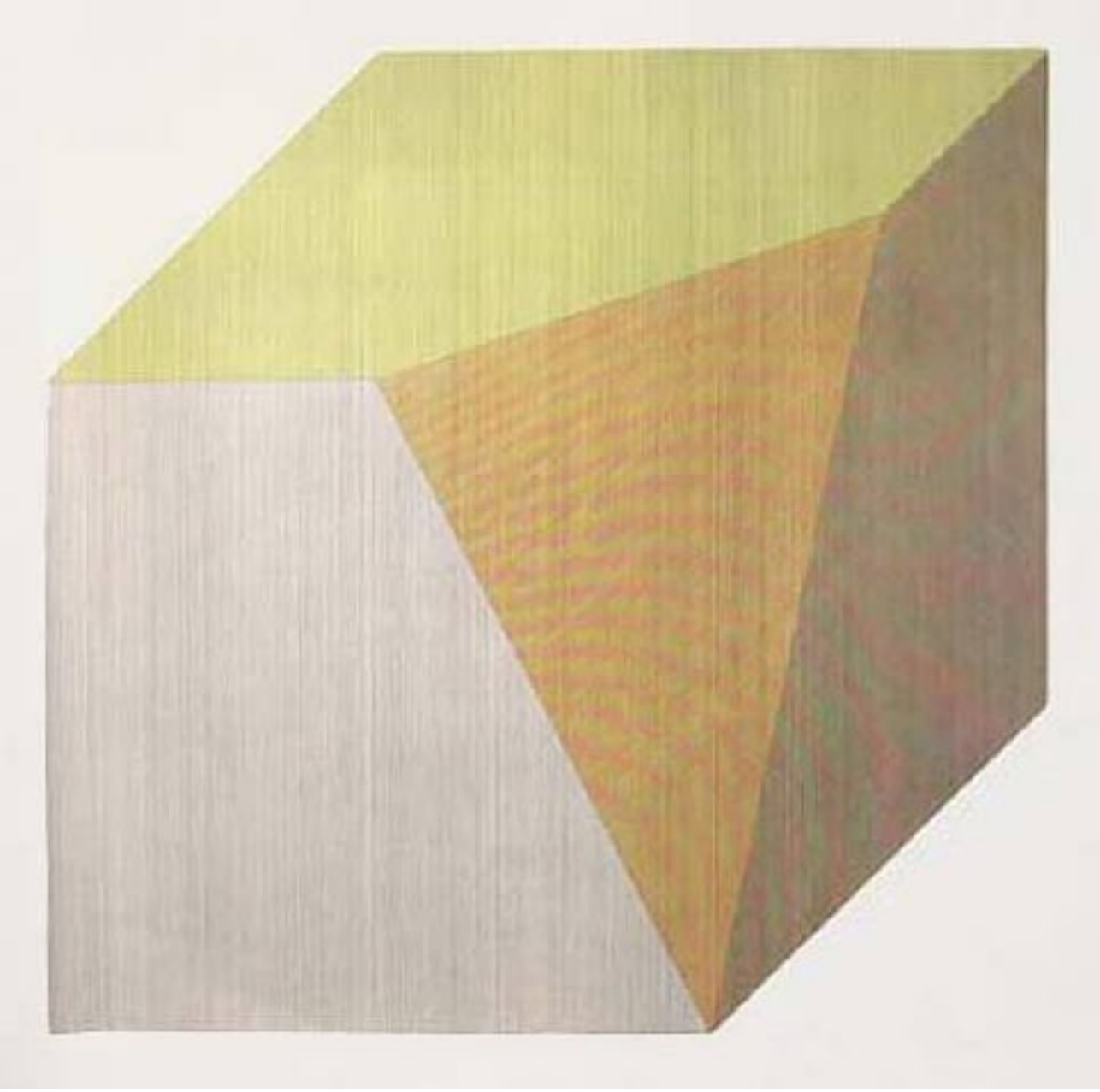
Carsten Nicolai : anti (2005).
http://www.eigen-art.com/
http://www.antireflex.de/

Carsten Nicolai : Reflex (2005).
http://www.eigen-art.com/
http://www.antireflex.de/

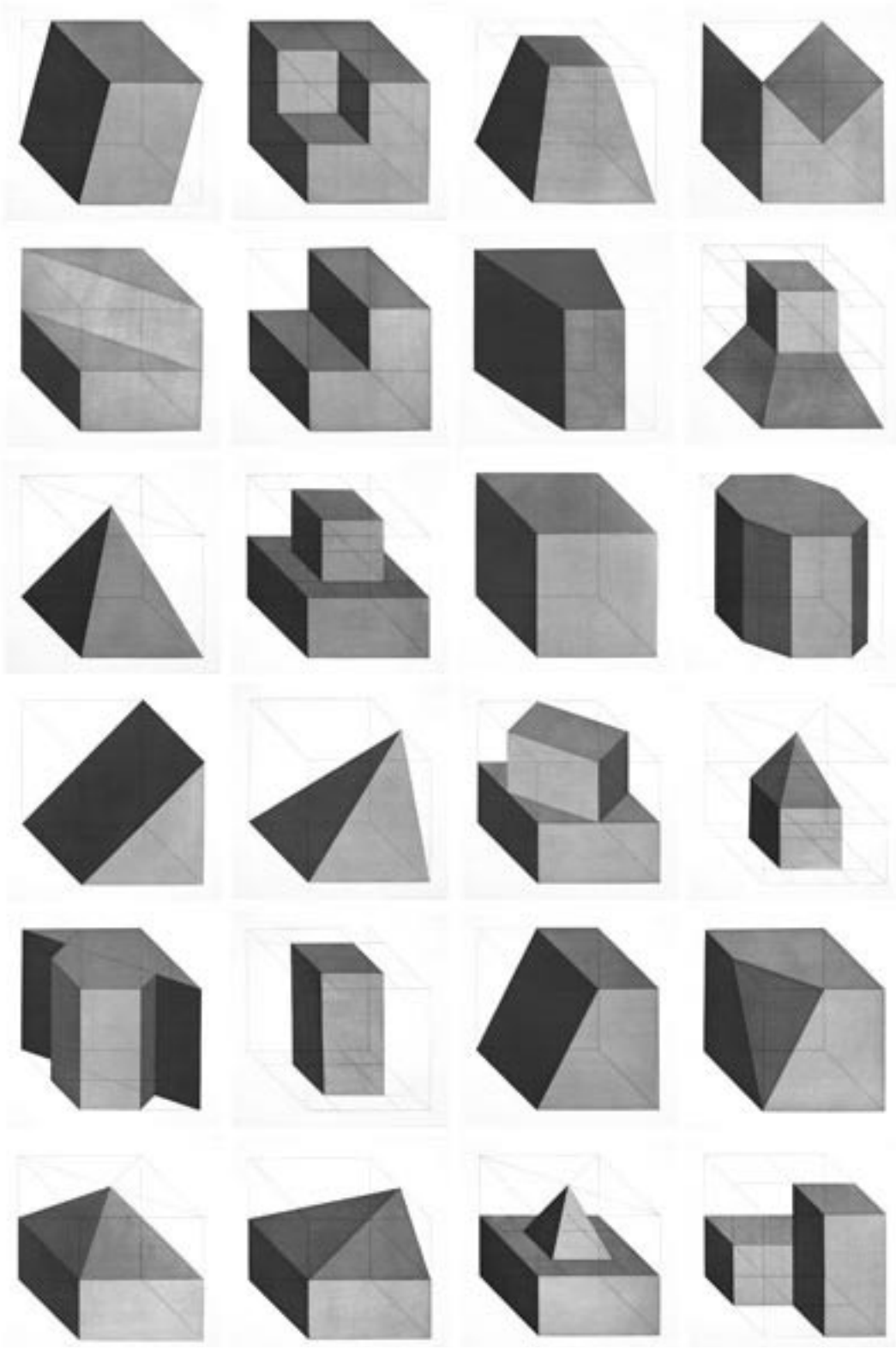


Carsten Nicolai : Syn chron (2005).
Neue Nationalgalerie, Berlin
http://www.eigen-art.com/

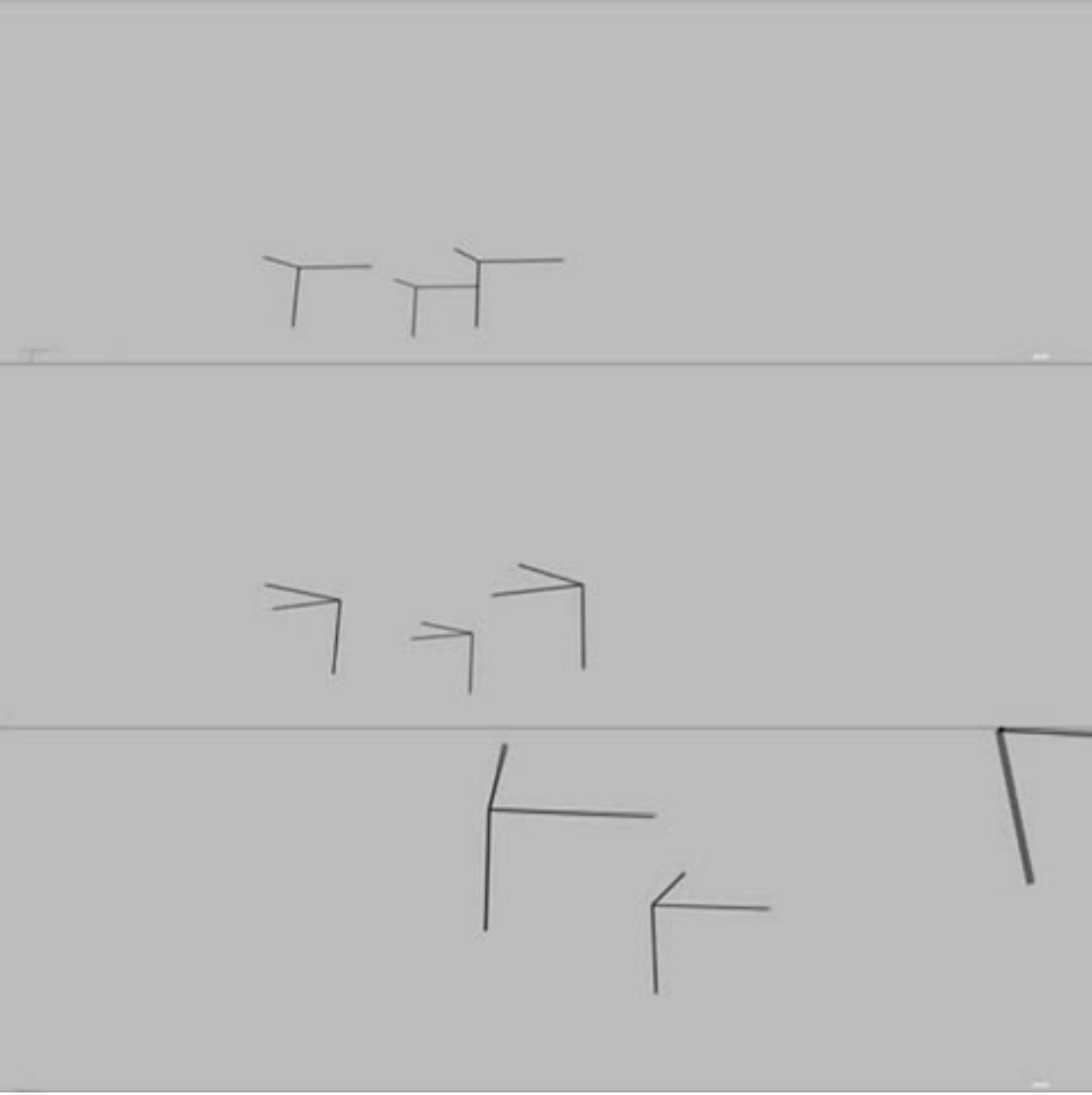
Elizabeth Murray : Red Shoe (1996).
http://stuartcollection.ucsd.edu/StuartCollection/Murray.htm



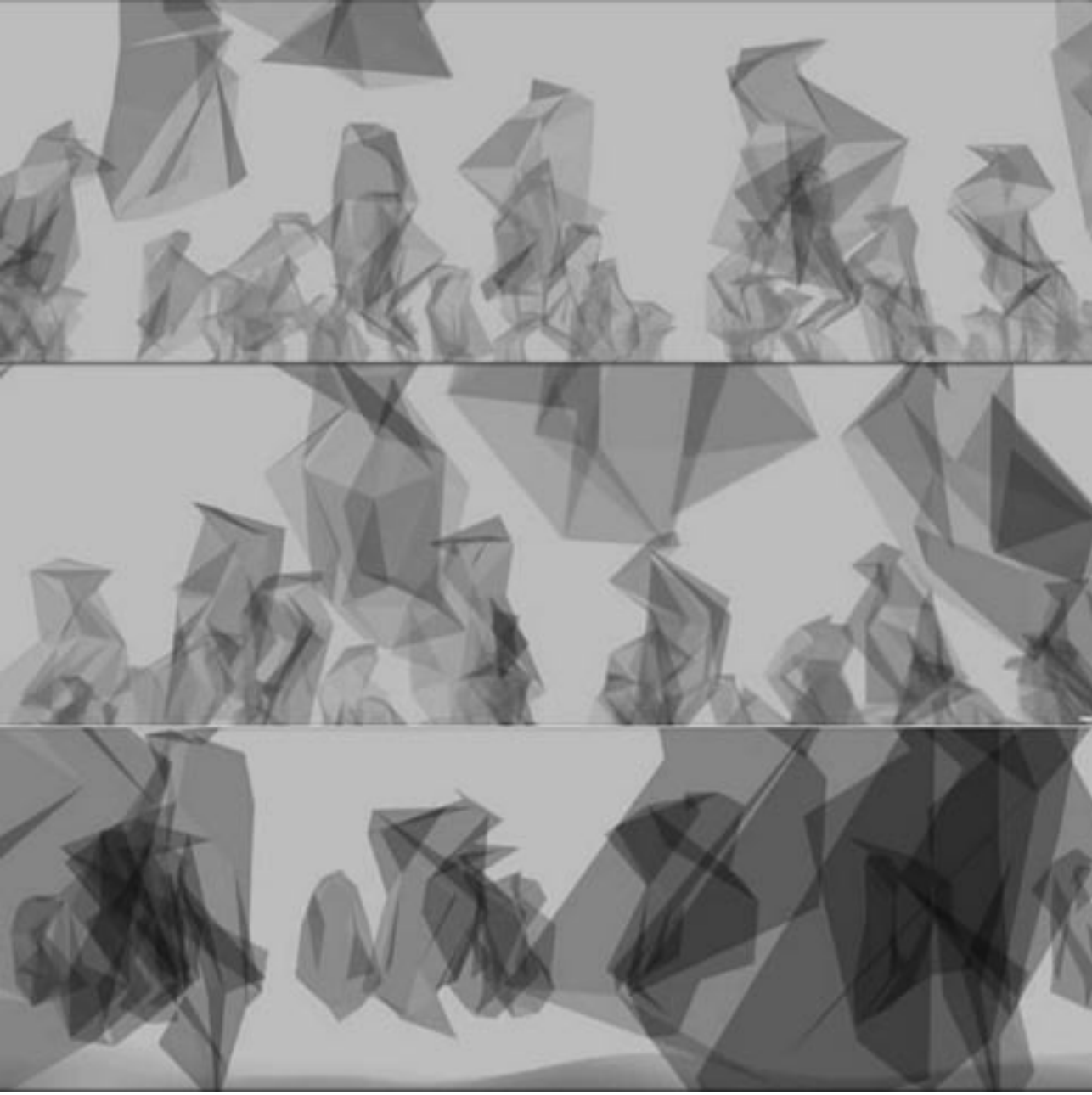
Sol Lewitt : Isometric Form
with Lines in Four Directions and Four Colors (1983).



Sol Lewitt : Forms Derived from a Cube (1982).



Anouk de Clercq : Portal (2002),
DVD.
http://www.portapak.be/



Anouk de Clercq : Portal (2002),
DVD.
http://www.portapak.be/

Marcel Biefer-Beat Zgraggen : God (1998).
http://www.biefer.com/kunst/

Cameron McNall & Damon Seeley : Terra Metallum, Artpark, NYC (1991).
http://electroland.net/

OMA (Office for Metropolitan Architecture) - Rem Koolhaas :
Seattle Central Library (2004).
http://www.oma.nl/
http://fr.wikipedia.org/wiki/Rem_Koolhaas

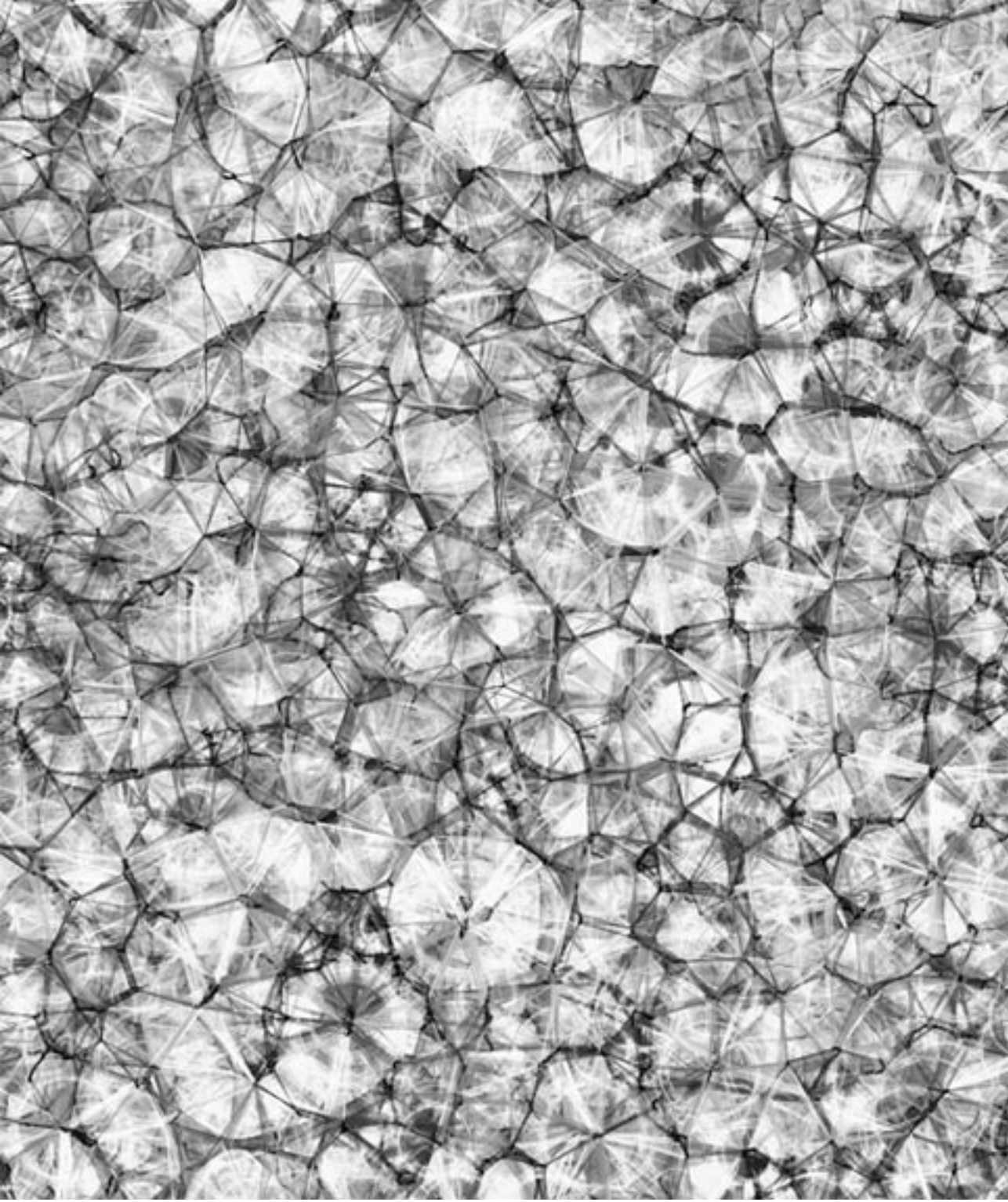
OMA (Office for Metropolitan Architecture) - Rem Koolhaas :
Seattle Central Library (2004).
http://www.oma.nl/
http://fr.wikipedia.org/wiki/Rem_Koolhaas

Morphosis (Thom Mayn) :
Diamond Ranch High School, Pomona, California (1999).
http://www.morphosis.net/

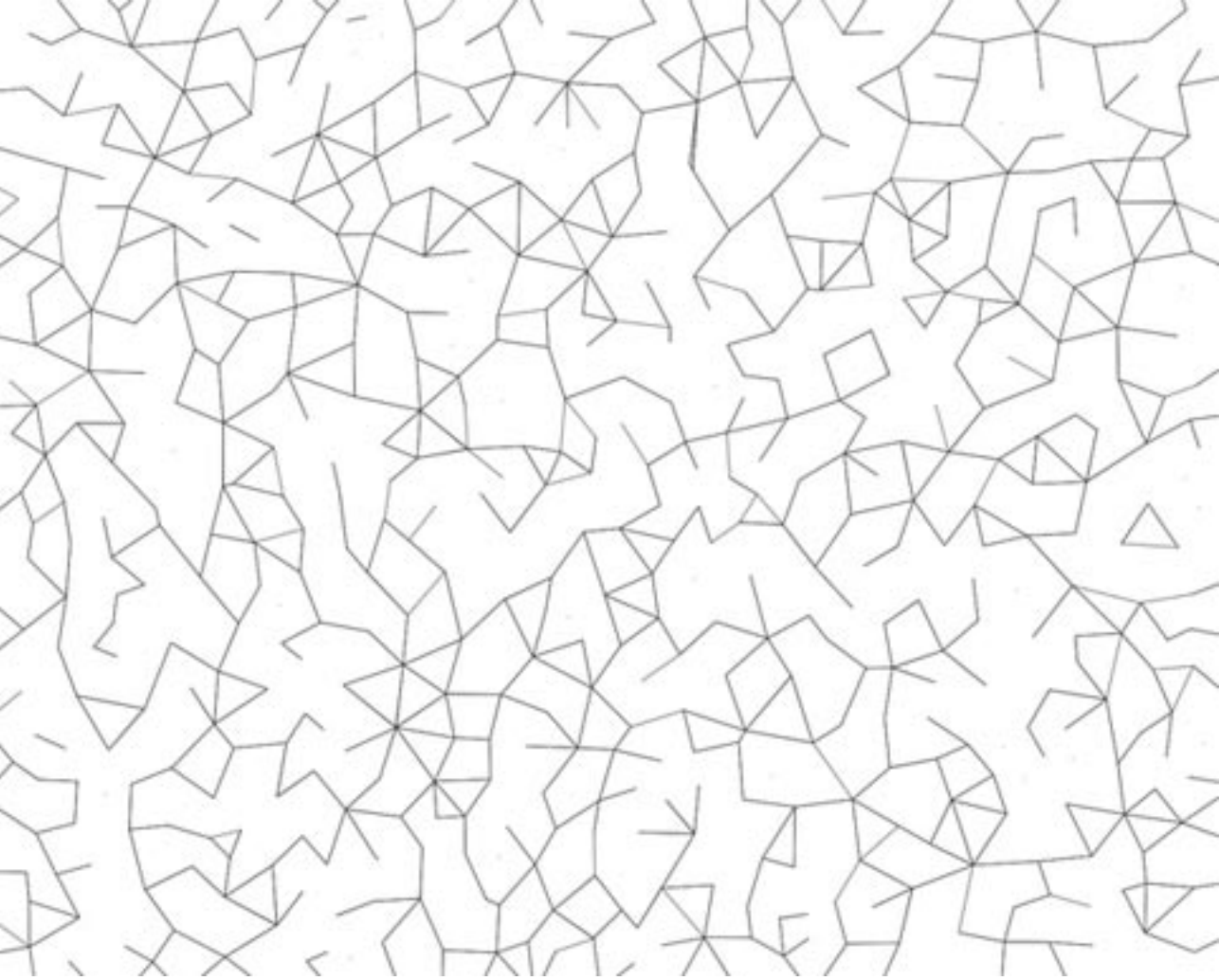
Morphosis (Thom Mayn) :
Diamond Ranch High School, Pomona, California (1999).
http://www.morphosis.net/

Zaha M. Hadid : Fire station,
Weil am Rhein, Germany (1992).
http://www.zaha-hadid.com/

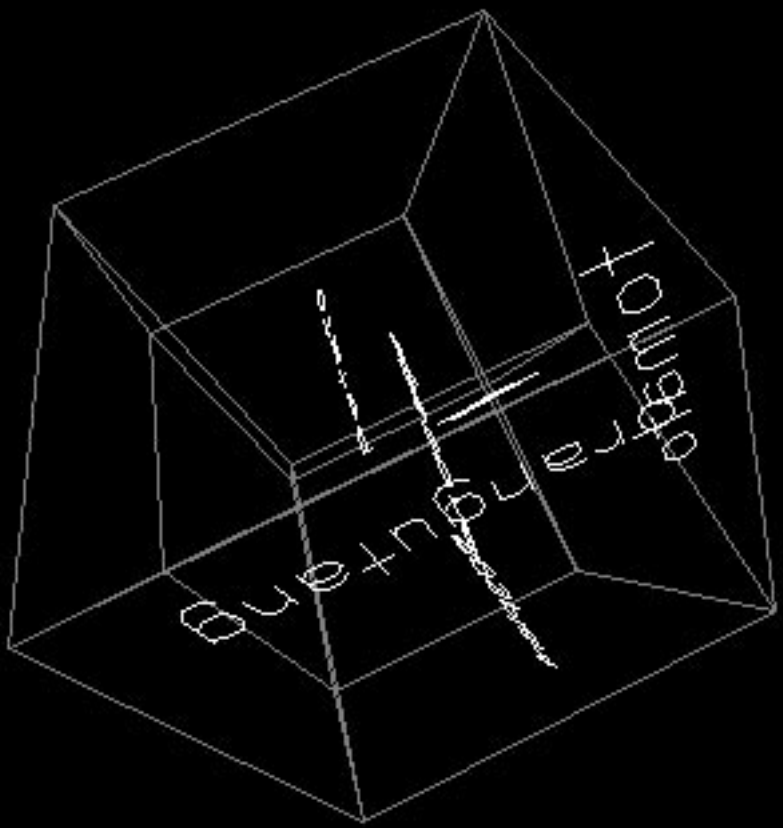
Zaha M. Hadid : Fire station,
Weil am Rhein, Germany (1992).
http://www.zaha-hadid.com/



Casey Reas : Process 4, Form 2 (2005).
http://reas.com/



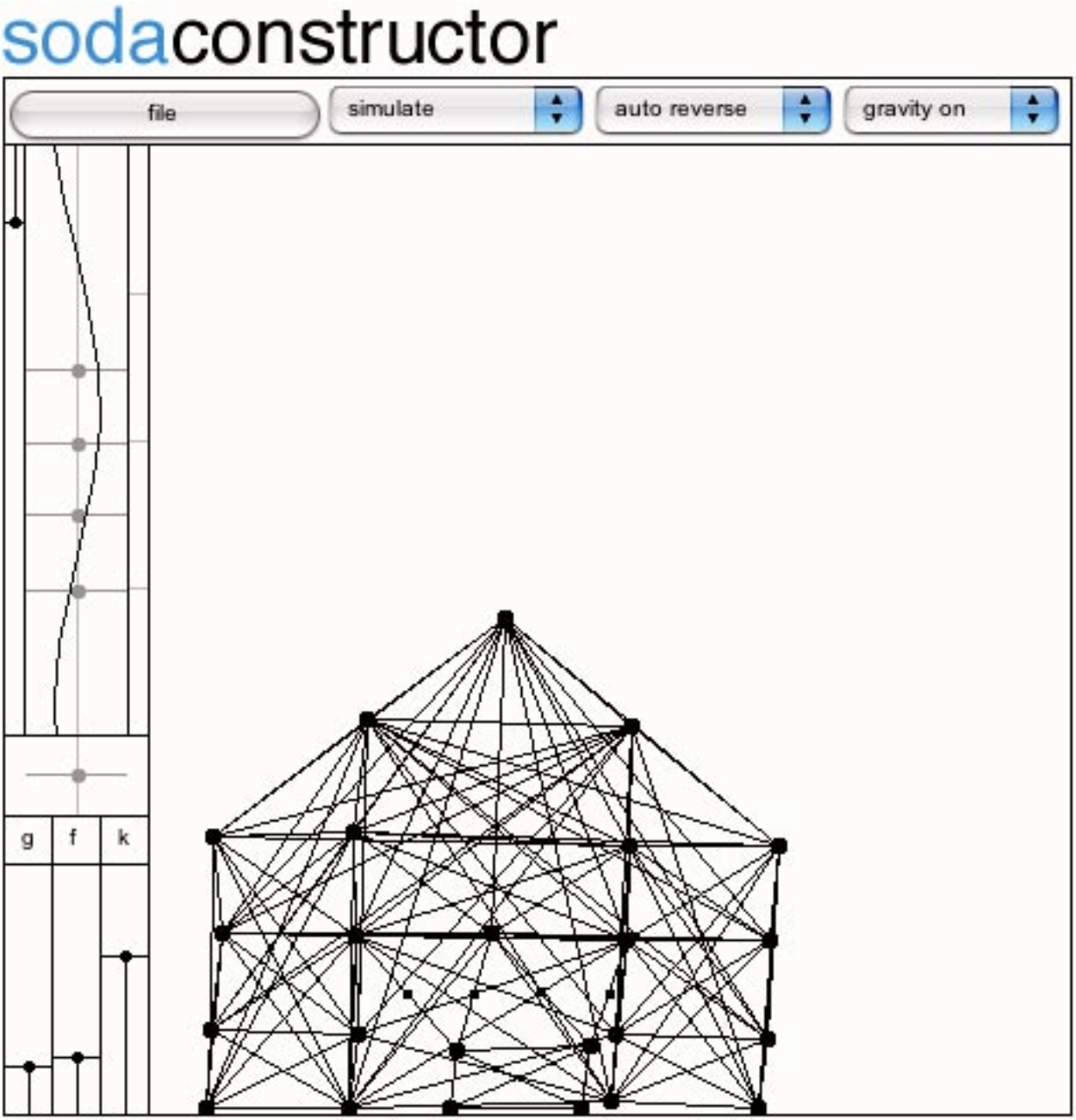
Casey Reas : Preparatory software for Process 4, 5, 6 (2005).
http://reas.com/



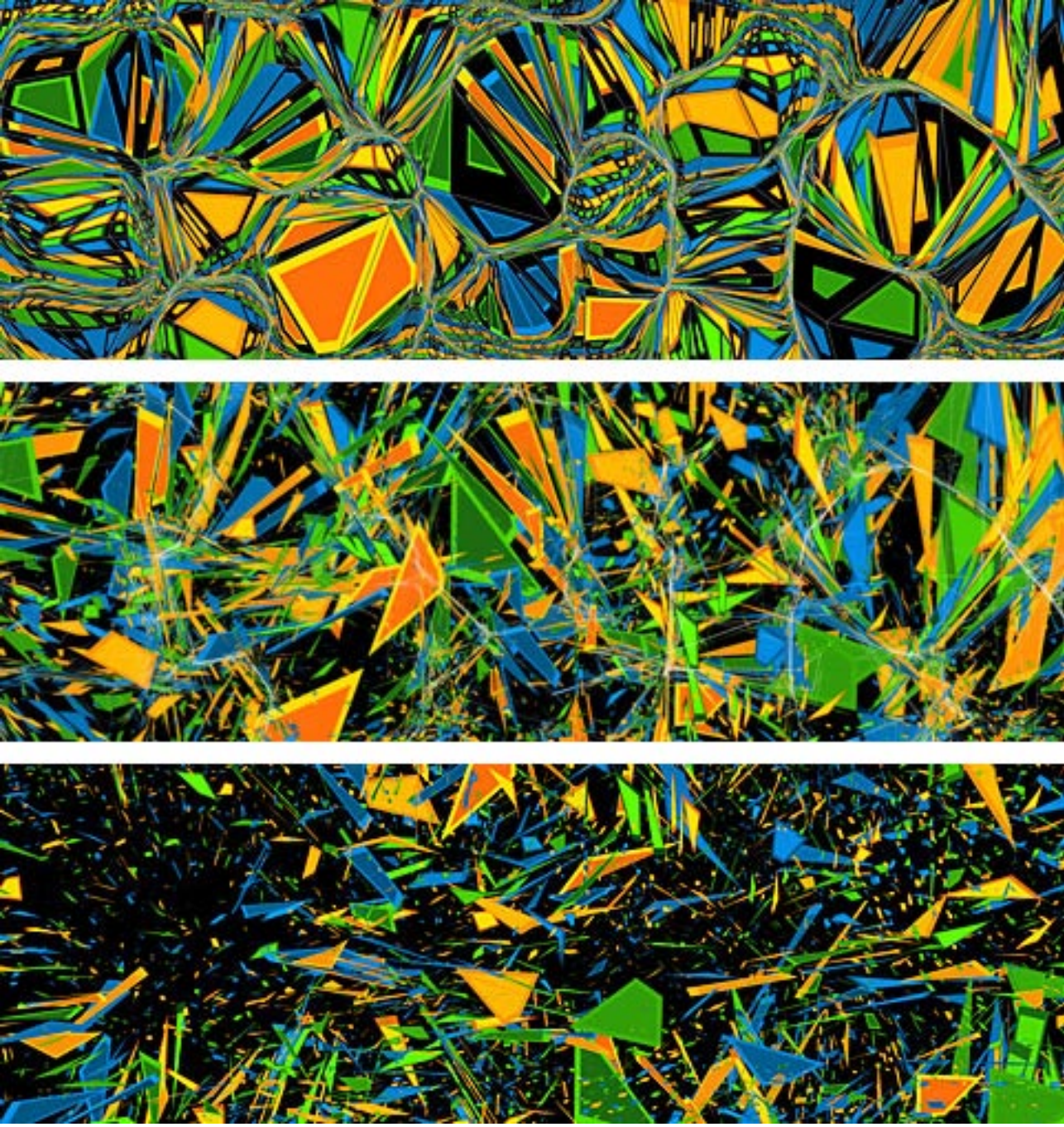
Ben Fry : Hypercube (2000/2005).
http://acg.media.mit.edu/people/fry/ndim/

Scott Snibbe : Boundary Functions (1998).
http://www.snibbe.com/

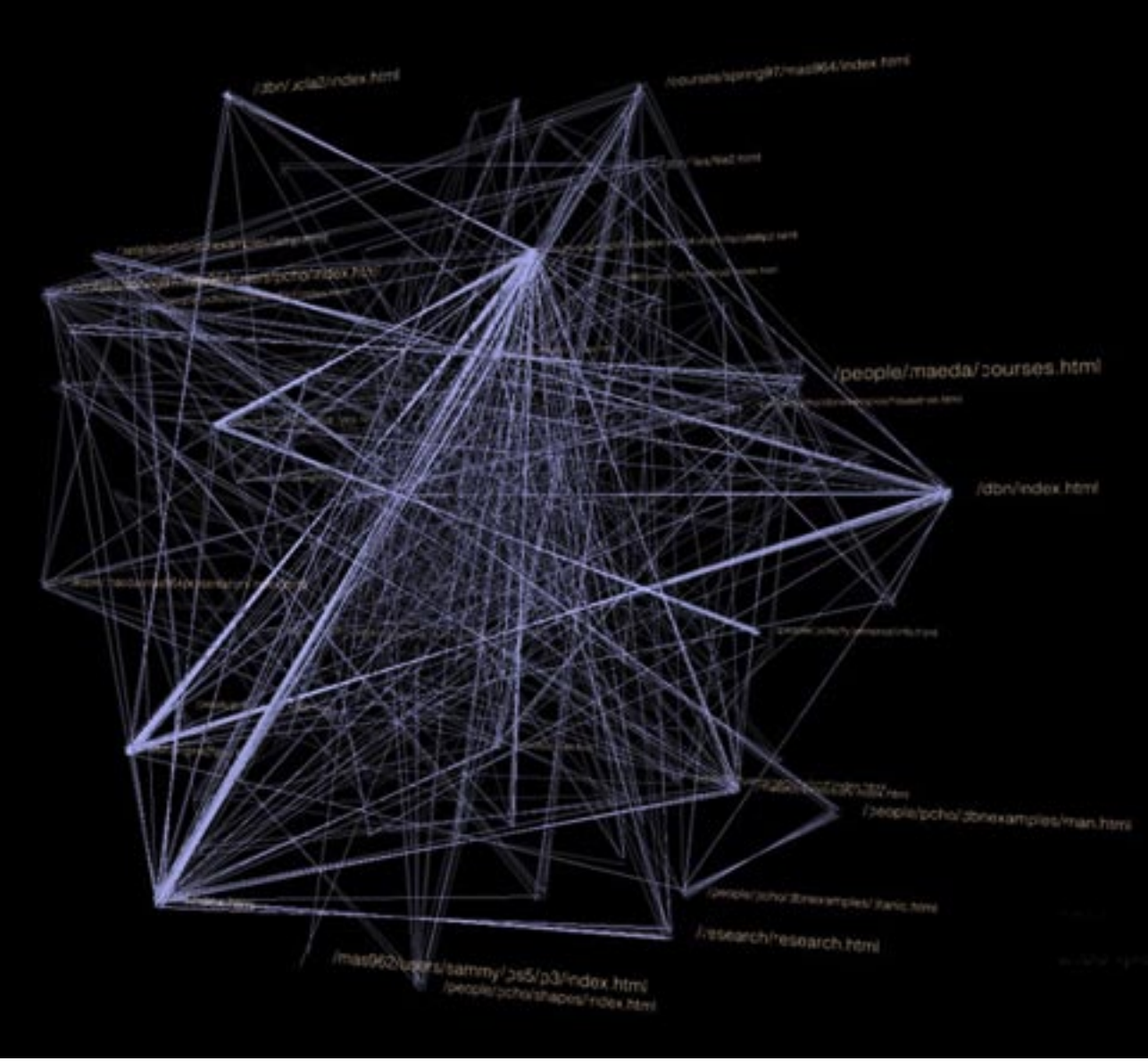
Josh Nimoy : «Zero@wavefunction» (1998),
environnement interactif.
http://www.jtnimoy.com/zerowave/



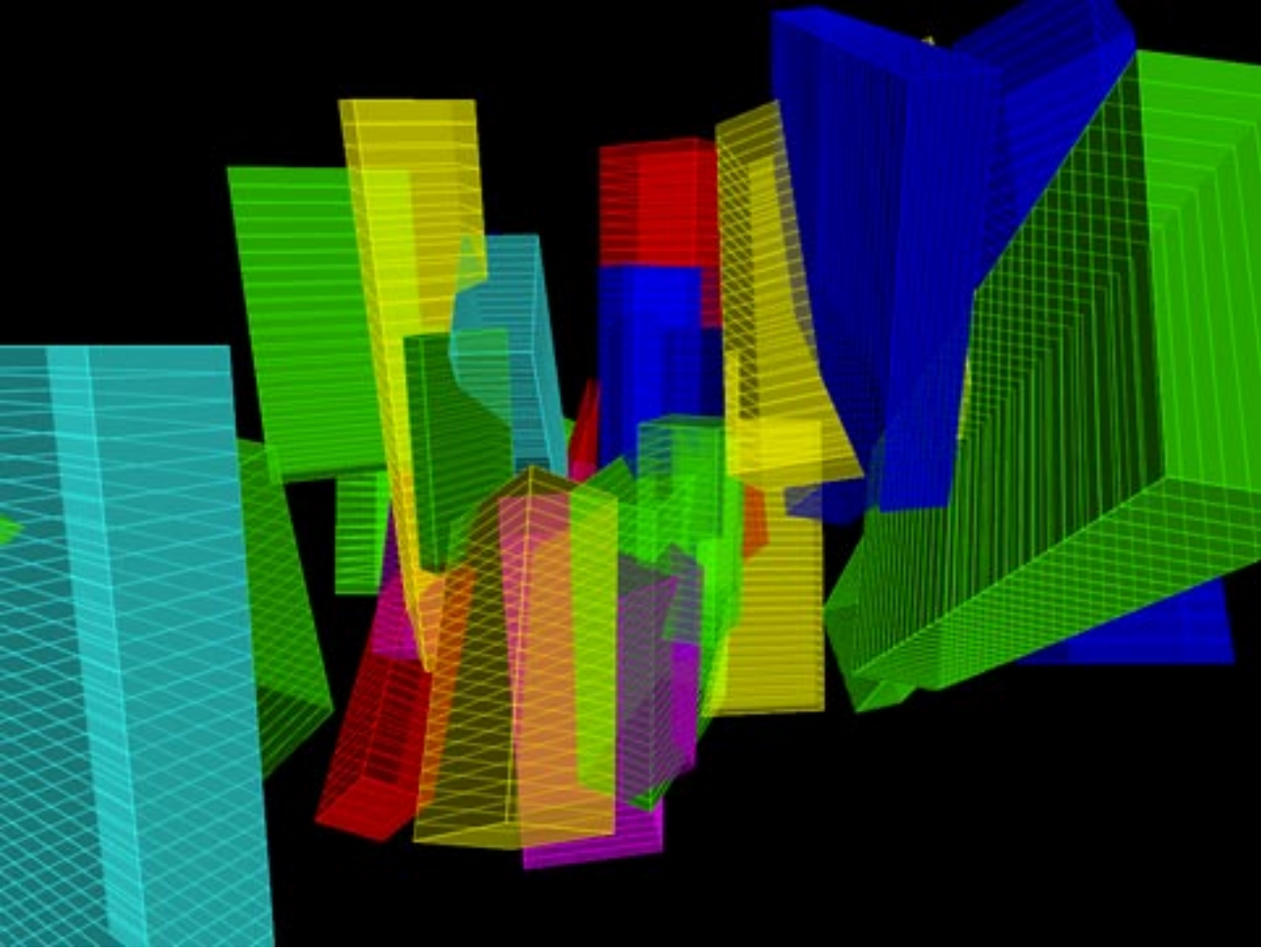
Soda : SodaPlay (2000),
net-art en Java.
http://sodaplay.com/



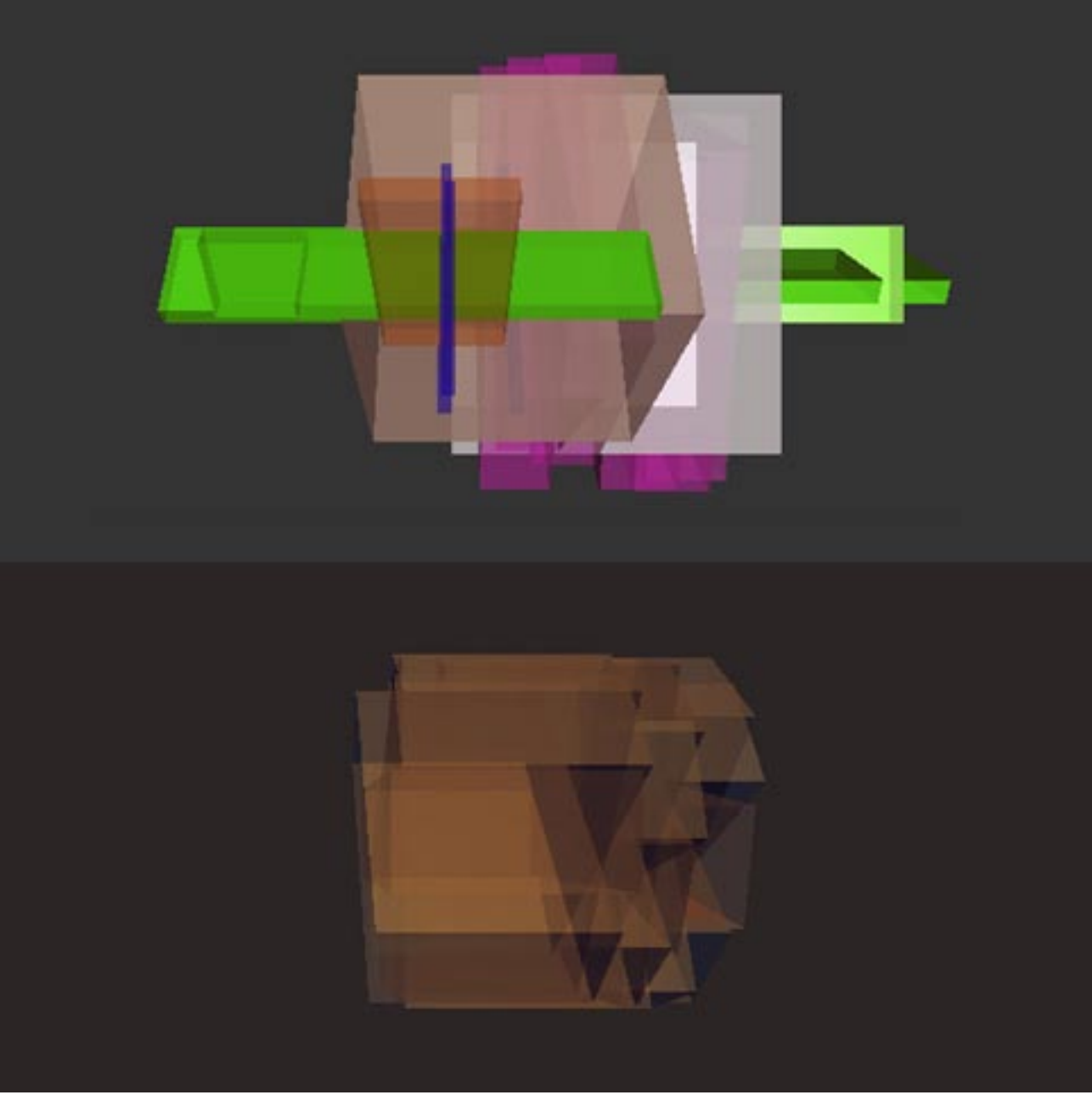
Marius Watz : ElectroPlastique #1 (2005).
http://www.unlekker.net/proj/electroplastique/



Benjamin Fry : Valence (2001).
Visualisation de structures et de relations
au sein de grands ensembles d'informations.
http://acg.media.mit.edu/people/fry/valence/



Marius Watz : ElectroPlastique #1 (2005).
http://www.unlekker.net/proj/electroplastique/



Limiteazero : Active Metaphor (2002),
logiciel «Carnivore Client», visualisations du trafic Internet.
http://www.limiteazero.com/carnivore/

Intention.

Algorithme.

L’ordinateur est un **outil**,
un **exécutant** pointilleux,
méticuleux,
très rapide,
mais totalement dénué d’imagination et d’initiative.

La machine ne fait rien par elle-même;
elle ne fait qu’**appliquer** à la lettre des ordres simples et précis.

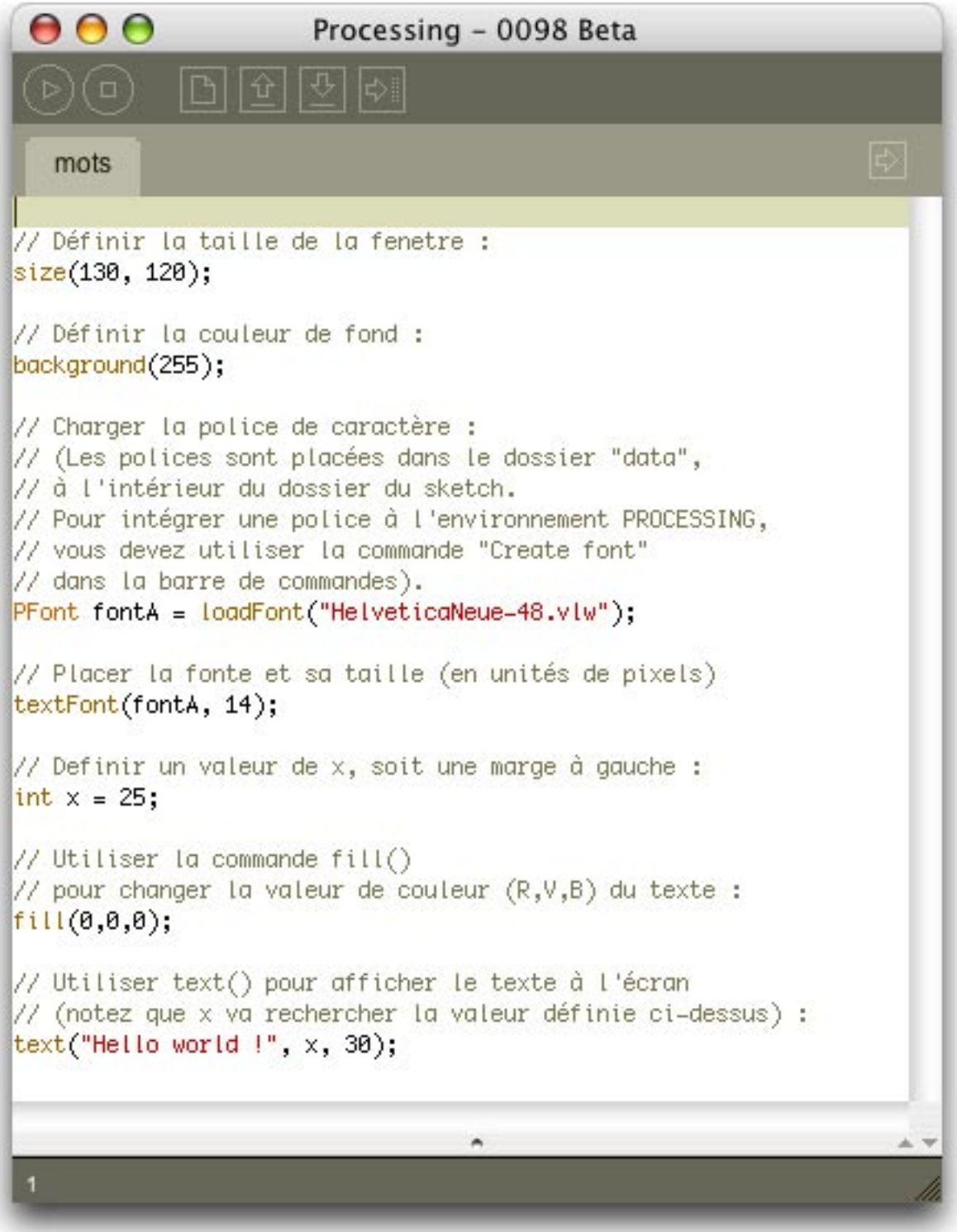
... sauf Hal !

HAL 9000 est le nom du puissant ordinateur doté d'intelligence artificielle, gérant le vaisseau spatial Discovery 1 dans «2001, a space odyssey» de Stanley Kubrick (1968). HAL signifie «Heuristically programmed ALgorithmic computer» et, dans la version française, Carl, soit «Cerveau Analytique de Recherche et de Liaison», ce qui est une assez habile traduction. L'acronyme HAL correspondrait à IBM par décalage d'un rang de chacune des lettres : H->I, A->B, L->M), ce n'est peut-être pas un hasard. Stanley Kubrick et Arthur Charles Clarke ont cependant toujours démenti cette allégation.

http://en.wikipedia.org/wiki/HAL_9000
http://www.robothalloffame.org/hal.html

L’ordinateur reçoit des **instructions** et les **exécute**,
séquentiellement,
sans rechigner
et sans intelligence.

Une suite de telles instructions est un **programme**.



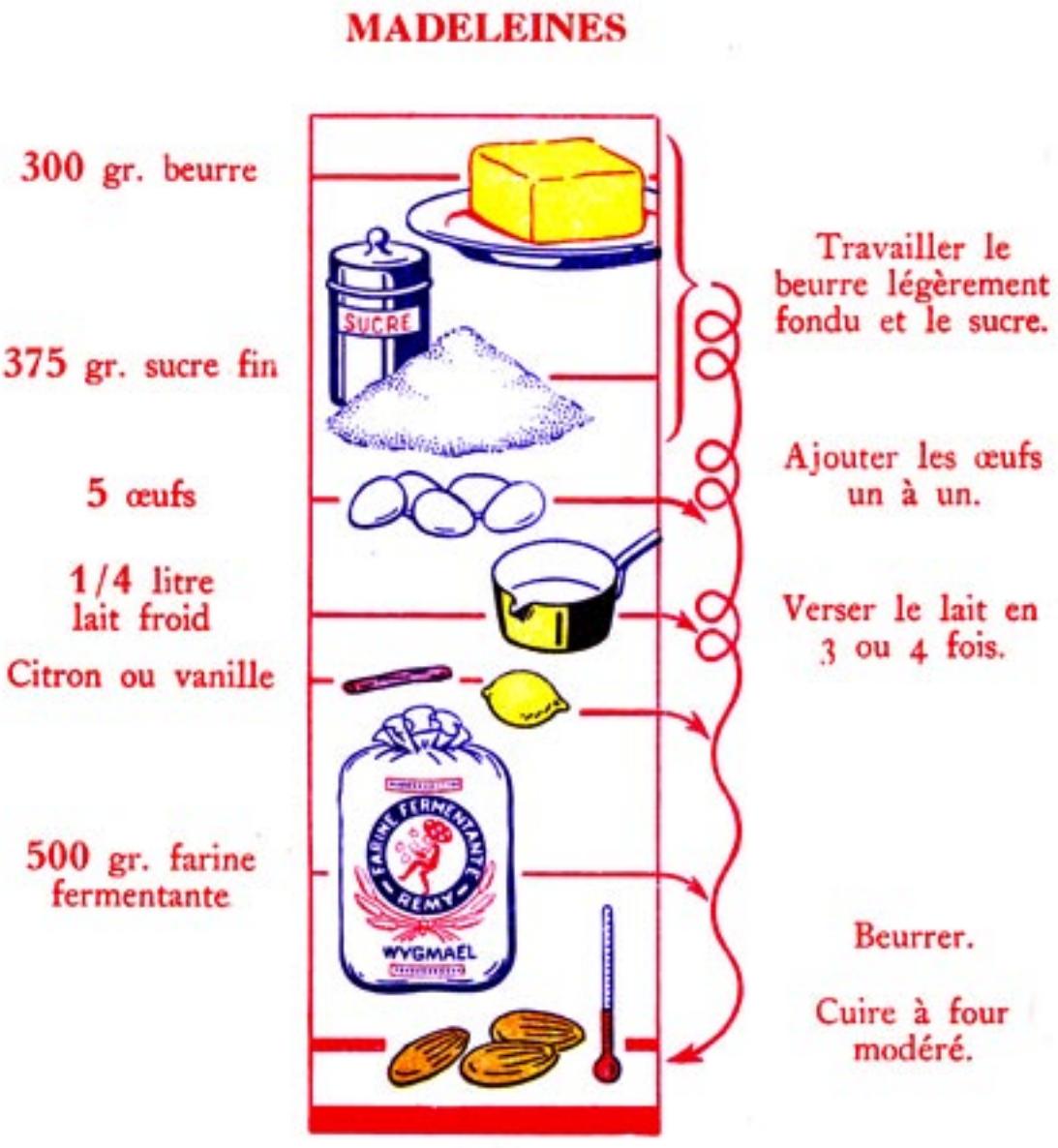
http://processing.org/


Hello
Hello world !

La **programmation** consiste à déterminer la démarche permettant d'obtenir, à l'aide d'un ordinateur, la solution à un problème donné.

Cette marche à suivre s'appelle un **algorithme**.

Une recette est un algorithme.



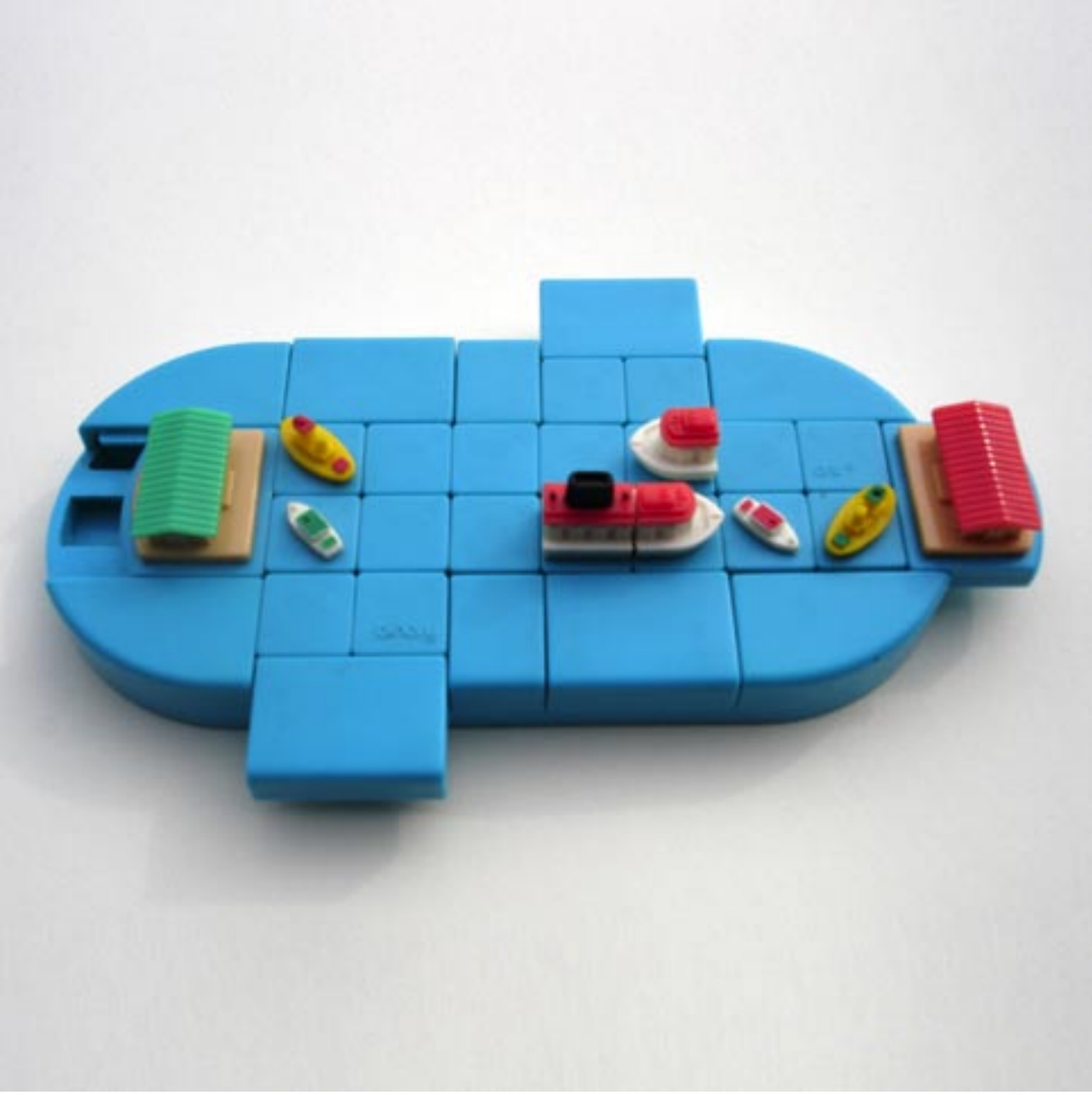
De **87 Rue du Page Bruxelles (Belgique)** A **10 Rue du Tabellion Bruxelles (Belgique)**

Itinéraire **Express** Distance **0.3 km** Voies express **0.0 km** Durée (1) **0H06**
Indemnités **0.0 €** Véhicule **Piéton**

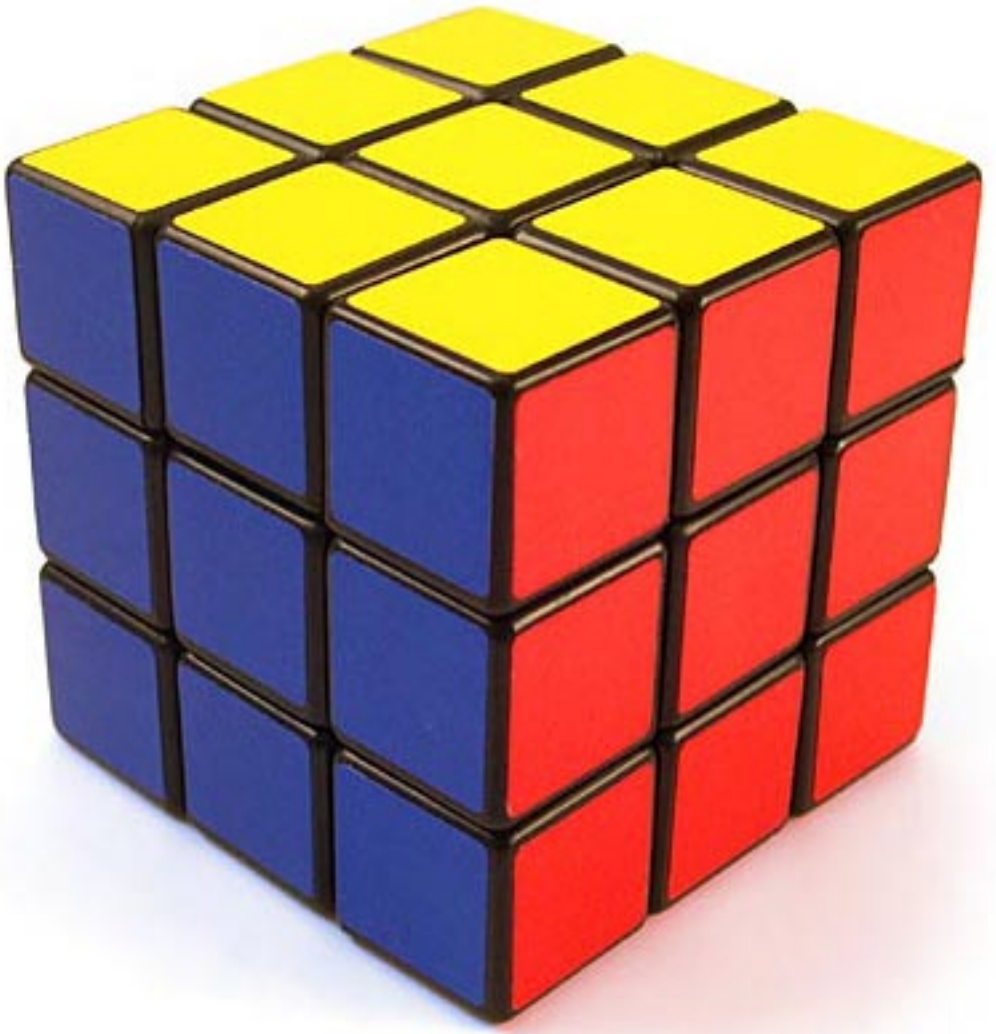
Cumul	Temps		Feuille de route
0 m	0H00		**87 Rue du Page Bruxelles (Belgique)**
			Continuer sur **Rue du Page [27m]**
27 m			Prendre à gauche **Rue Fourmois [120m]**
150 m	0H02		Prendre à droite **Rue du Tabellion [240m]**
390 m	0H06		**10 Rue du Tabellion Bruxelles (Belgique)**

Un mode d’emploi est un algorithme.

Fifteen Puzzle.



Port-to-Port.



Le Rubik’s cube, casse-tête inventé en 1974 par Ernö Rubik, sculpteur et architecte hongrois.
http://fr.wikipedia.org/wiki/Cube_de_Rubik

Le boulier chinois (suan pan): chaque tige correspond, de droite à gauche, respectivement aux unités, dizaines, centaines, milliers etc.
En 1945, Kiyoshi Matsuzaki, un comptable japonais muni d'un soroban (boulier japonais) remporta un match contre Nathan Woods, un opérateur de calculatrice électrique, par un score de 4 à 1.
http://en.wikipedia.org/wiki/Abacus

Man Ray et Marcel Duchamp disputant une partie d’échec.
http://www.marcelduchamp.net/

Gary Kasparov triomphe
contre le programme Deep Blue de IBM en février 1996.
Lors de la revanche, un an plus tard, il perd contre Deeper Blue.
http://www.research.ibm.com/deepblue/

Martin Wattenberg a réalisé un jeu d'échec avec PROCESSING :
http://www.turbulence.org/spotlight/thinking/chess.html

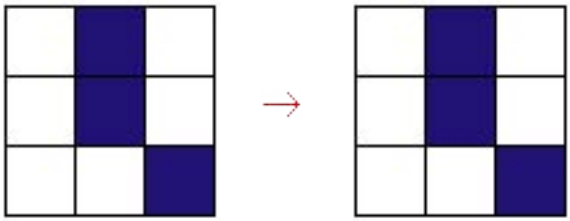
Le «Jeu de la vie» imaginé par John Horton Conway en 1970
est probablement le plus connu des automates cellulaires.
http://en.wikipedia.org/wiki/Conway's_Game_of_Life
http://www.metaphorical.net/code/processing/index2.html?page=life

Le jeu de la vie ne nécessite aucun joueur : il s'agit d'un automate cellulaire, c'est-à-dire un modèle où chaque état conduit mécaniquement à l'état suivant à partir de règles pré-établies.

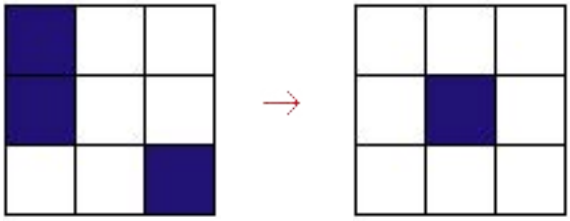
Le jeu se déroule sur une grille à deux dimensions, dont les cases (les «cellules») peuvent prendre deux états distincts : «vivantes» ou «mortes».

À chaque étape, l'évolution d'une cellule est entièrement déterminée par l'état de ses huit voisines de la façon suivante :

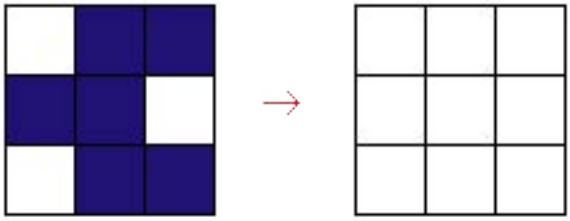
1- Si une cellule vivante (ici en bleu) est entourée de deux ou trois voisines vivantes, alors elle reste vivante. le reste, sinon elle meurt.



2- Si une cellule morte est entourée d'exactement trois voisines vivantes devient vivante (elle naît).



3- Dans tous les autres cas, la cellule meurt.



Pour plus de détails, lire l'article de Jean-Claude Heudin en annexe.

Voir un «Jeu de la vie» programmé avec PROCESSING par Michael Davis :
http://processing.org/learning/examples/cellularautomata1.html

Un autre programmé avec PROCESSING par William Ngan :
http://www.metaphorical.net/code/processing/index2.html?page=life

Construire un algorithme, c'est avant tout :

analyser l'énoncé du problème afin de **définir** l'ensemble des objets à manipuler pour obtenir un **résultat**.

donc :

trouver le **cheminement logique** des tâches à fournir à l'ordinateur pour qu'il les exécute.

Exemple: l'algorithme du café chaud.

Analysons l'énoncé suivant :

Comment faire un café chaud non sucré ?

Chaque mot a son importance, et «non sucré» est aussi important que «café» ou «chaud». Le terme «non sucré» implique qu'il n'est pas nécessaire de prendre du sucre ni une petite cuillère.

Notons que tous les ingrédients et ustensiles nécessaires ne sont pas cités dans l'énoncé. En particulier, nous ne savons pas si nous disposons d'une cafetière électrique ou non. Pour résoudre notre problème, nous devons prendre certaines décisions, et ces dernières vont avoir une influence sur l'allure générale de notre algorithme.

Nous allons
- Définitir les objets manipulés.
- Lister les opérations
- Ordonner la liste des opérations

Supposons que, pour réaliser notre café, nous soyons en possession des ustensiles et ingrédients suivants :
- café moulu filtre
- eau
- pichet
- cafetière électrique
- tasse
- électricité
- table

En fixant la liste des ingrédients et des ustensiles, nous définissons un environnement, une base de travail. Nous sommes ainsi en mesure d'établir une liste de toutes les actions à mener pour résoudre le problème et de construire la marche à suivre permettant d'obtenir un café.

Liste des opérations :

Verser l'eau dans la cafetière,
le café dans la tasse,
le café dans le filtre.
Remplir le pichet d'eau.
Prendre du café moulu,
une tasse,
de l'eau,
une cafetière électrique,
un filtre,
le pichet de la cafetière.
Brancher,
allumer ou éteindre la cafetière électrique.
Attendre que le café remplisse le pichet.
Poser la tasse,
la cafetière sur la table,
le filtre dans la cafetière,
le pichet dans la cafetière.

Cette énumération est une description de toutes les actions nécessaires à la réalisation d'un café chaud.

Chaque action est un fragment du problème donné et ne peut plus être découpée.

Chaque action est élémentaire par rapport à l'environnement que nous nous sommes donné. En définissant l'ensemble des actions possibles, nous créons un langage minimal qui nous permet de réaliser le café.

Ce langage est composé de verbes (Prendre, Poser, Verser, Faire, Attendre, etc.) et d'objets (Café moulu, Eau, Filtre, Tasse, etc.). La taille du langage, c'est-à-dire le nombre de mots qu'il renferme, est déterminée par l'environnement.

Pour cet exemple, nous avons, en précisant les hypothèses, volontairement choisi un environnement restreint. Nous aurions pu décrire des tâches comme «prendre un contrat EDF» ou «planter une graine de café», mais elles ne sont pas utiles à notre objectif pédagogique.

Remarque:
Telle que nous l'avons décrite, la liste des opérations ne nous permet pas encore de faire un café chaud. En suivant cette liste, tout y est, mais dans le désordre. Pour réaliser ce fameux café, nous devons ordonner la liste.

Ordonner la liste des opérations :

01. Prendre une cafetière électrique.
02. Poser la cafetière sur la table.
03. Prendre un filtre.
04. Poser le filtre dans la cafetière.
05. Prendre du café moulu.
06. Verser le café moulu dans le filtre.
07. Prendre le pichet de la cafetière.
08. Remplir le pichet d'eau.
09. Verser l'eau dans la cafetière.
10. Poser le pichet dans la cafetière.
11. Brancher la cafetière.
12. Allumer la cafetière.
13. Attendre que le café remplisse le pichet.
14. Prendre une tasse.
15. Poser la tasse sur la table.
16. Éteindre la cafetière.
17. Prendre le pichet de la cafetière.
18. Verser le café dans la tasse.

L'exécution de l'ensemble ordonné de ces tâches nous permet maintenant d'obtenir du café chaud non sucré.

Remarque:
L'ordre d'exécution de cette marche à suivre est important. En effet, si l'utilisateur réalise l'opération 12 (Allumer la cafetière) avant l'opération 9 (Verser l'eau dans la cafetière), le résultat est sensiblement différent. La marche à suivre ainsi désordonnée risque de détériorer la cafetière électrique.

Pour en savoir plus sur les algorithmes :
http://www.commentcamarche.net/algo/algointro.php3

Créer une **application**, c'est la décomposer en plusieurs **sous-applications** qui, à leur tour, se décomposent en **micro-applications**, jusqu'à descendre ainsi au niveau le plus élémentaire.

Page suivante :
«Les noms des fleurs trouvés par la méthode simple» (1904) par Gaston Bonnier est un livre-machine qui fonctionne par élimination.

1 + Plante ***ayant des fleurs*** (Ces fleurs peuvent être quelquefois très petites, ou vertes, ou peu visibles)..... **2**

1 + Plante ***n'ayant jamais de fleurs,*** c'est-à-dire plante dont on ne voit jamais que les feuilles ou les tiges feuillées, comme les Fougères par exemple (Voir les figures aux nos 1092 à 1104).... **1092**

1092 (*vient de 1*).

× Rameaux ou écailles ***verticillés*** (figures AR) ; ou feuilles

réduites à des collerettes dentées au sommet (figure A) et placées les unes au-dessus des autres ; sporanges (c'est-à-dire petits sacs contenant les spores ou germes de la plante) ***groupés au sommet de la tige en une masse ovale*** (figure AV)..... **1104**

× Plante ***n'ayant pas à la fois*** ces caractères..... **1093**

△ Gaines de ***plus d'un centimètre et demi*** de largeur (figure T, grandeur naturelle) bordées de ***30 à 40 dents***. → **Prêle élevée** [*Equisetum maximum*]. — Figurée en couleurs (tiges vertes) : 3, planche 64.

1104 (*vient de 1092*).

△ Gaines de ***moins d'un centimètre et demi*** de largeur, bordées

de ***moins de 20 dents*** (figure A, grandeur naturelle ; les figures AR et AV représentent deux formes de tiges de la même plante). → **Prêle des champs** (Queue-de-cheval, Queue-de-rat) [*Equisetum arvense*]. — **industrielle ; médicinale**. — Figurée en couleurs : 2 et 2 *bis*, planche 64.

Play
About Us
20Q Store
License It

Pensez à un objet et l'intelligence artificielle essaiera de deviner à quoi vous pensez en posant des questions simples. L'objet que vous choisissez doit être connu de la plupart des gens et ne jamais être une personne, un lieu ou une chose spécifique.

Le grand ?, en haut et à droite, est là pour vous aider.

Q1. Est il Animal, Végétal, Minéral, Autre, ou Inconnue?

Suggestions

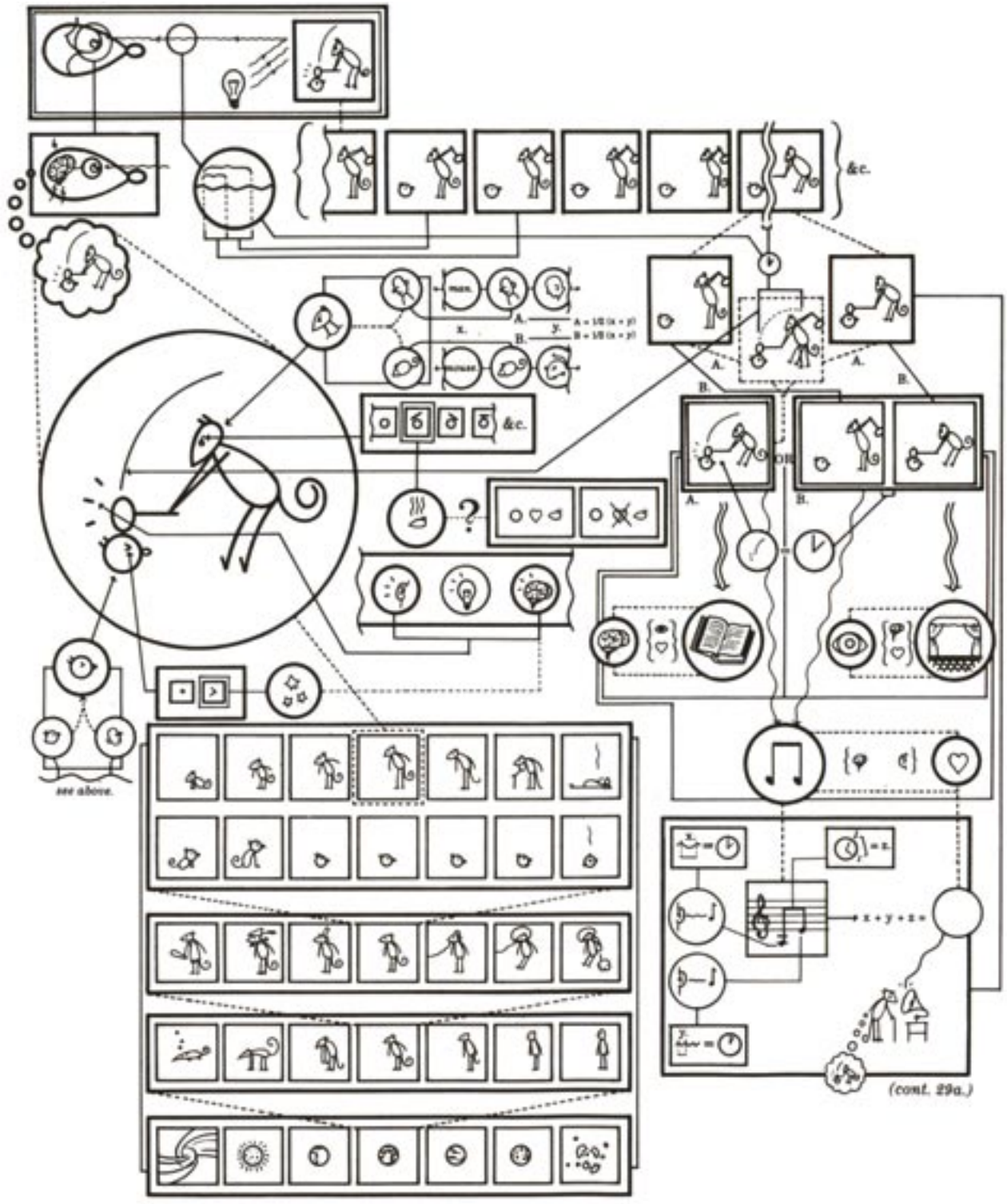
Si vous souhaitez des suggestions pour choisir un objet, le jeu vous recommande ce qui suit :

Choisis au hasard, . . .

une rétine, un minuteur, un rapporteur (pour mesurer les angles), une violette, une mangue.

Ce programme va deviner à quoi vous penser en 20 questions maximum.
http://y.20q.net/
http://www.20q.net/

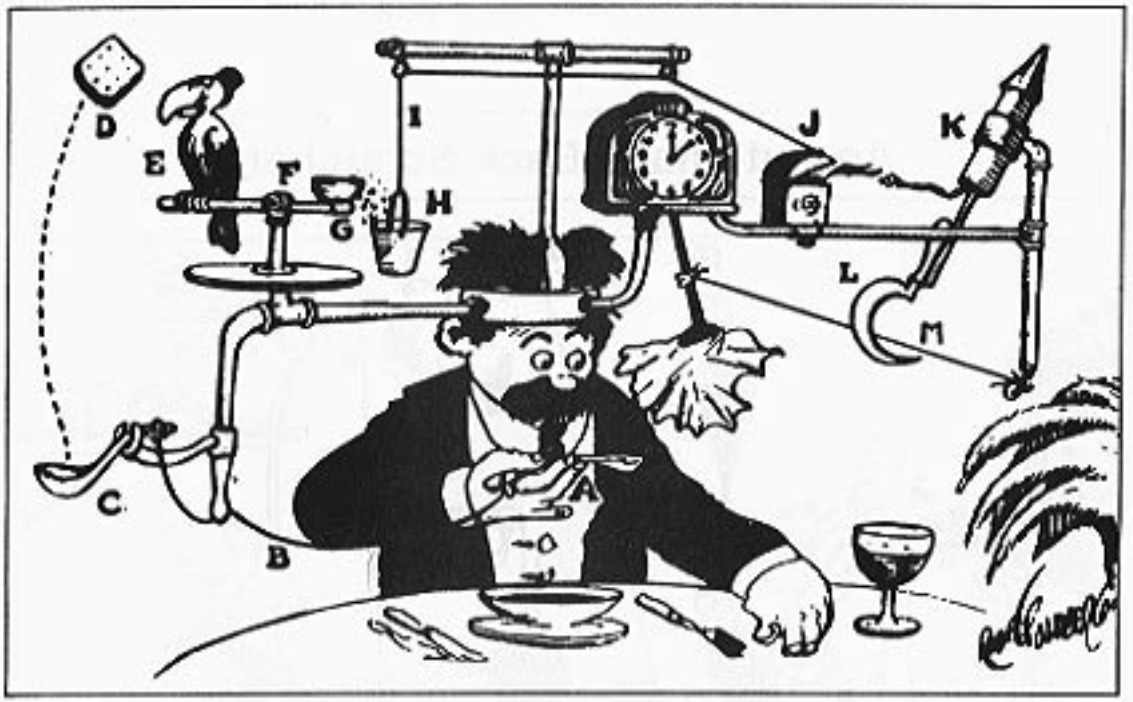
Cluedo : enquêter et déduire un scénario
en éliminant une à une les possibilités invraisemblables.



Chris Ware : Jimmy Corrigan (Issue six, fall 1995, page 16)
http://en.wikipedia.org/wiki/Chris_Ware

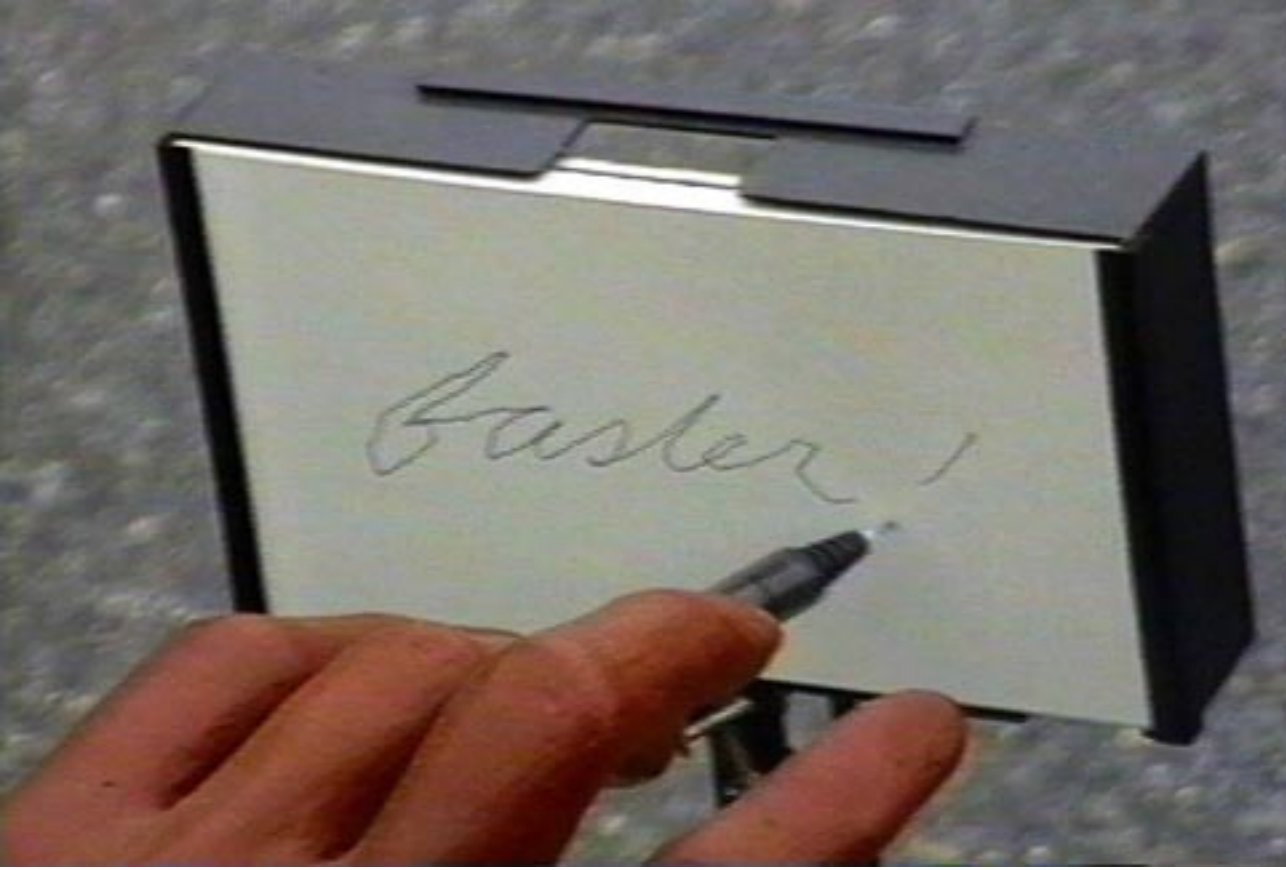
Vous avez encore perdu votre bouton de col : la colère vous fait lever les bras au ciel. Votre poing (**A**) heurte la poire (**B**), projetant sur le ménate (**C**) un jet d'eau pointu. Momentanément aveuglé, l'oiseau (**D**) quitte son perchoir (**E**), tombe dans le chariot de montagnes russes (**F**) qui glisse le long du rail (**G**), tendant la corde (**H**), laquelle active le levier (**I**). La main en bois (**J**) appuie sur la poupée qui parle (**K**). Celle-ci couine : «PLAY BALL!». Le lanceur liliputien de l'équipe des géants (**L**) attrape la balle (**M**) qui est collée au bras du phono (**N**), le mettant en marche. Le disque dit «Où qu'c'est-y qu'il a passé ?». Le père du lanceur (**O**), un penseur encore plus petit que son fils, est intrigué par la question, et marche de long en large pour y réfléchir. Absorbé dans sa réflexion, il passe sous le bureau (**P**), se cogne au bouton de col (**Q**) et crie «Ouille!», vous mettant ainsi sur la trace.

Rube Goldberg.
http://en.wikipedia.org/wiki/Rube_Goldberg

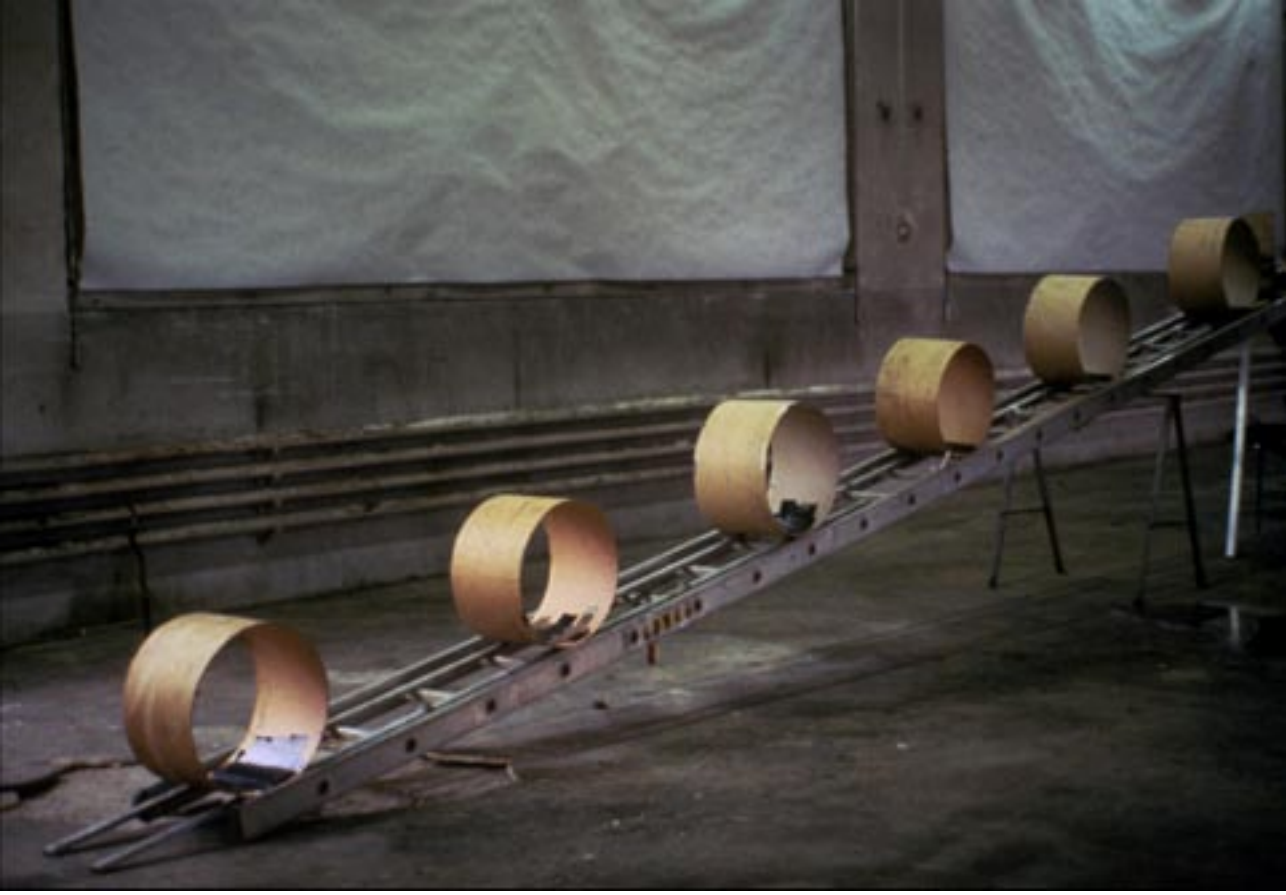


Rube Goldberg : Self-operating napkin.
http://en.wikipedia.org/wiki/Rube_Goldberg

Arthur Ganson : Faster !
http://www.arthurganson.com/



Arthur Ganson : Faster ! (Détail)
http://www.arthurganson.com/



Peter Fischli & David Weiss : Der Lauf Der Dinge, 16 mm (1987).
http://www.tcfilm.ch/lauf_txt_e.htm

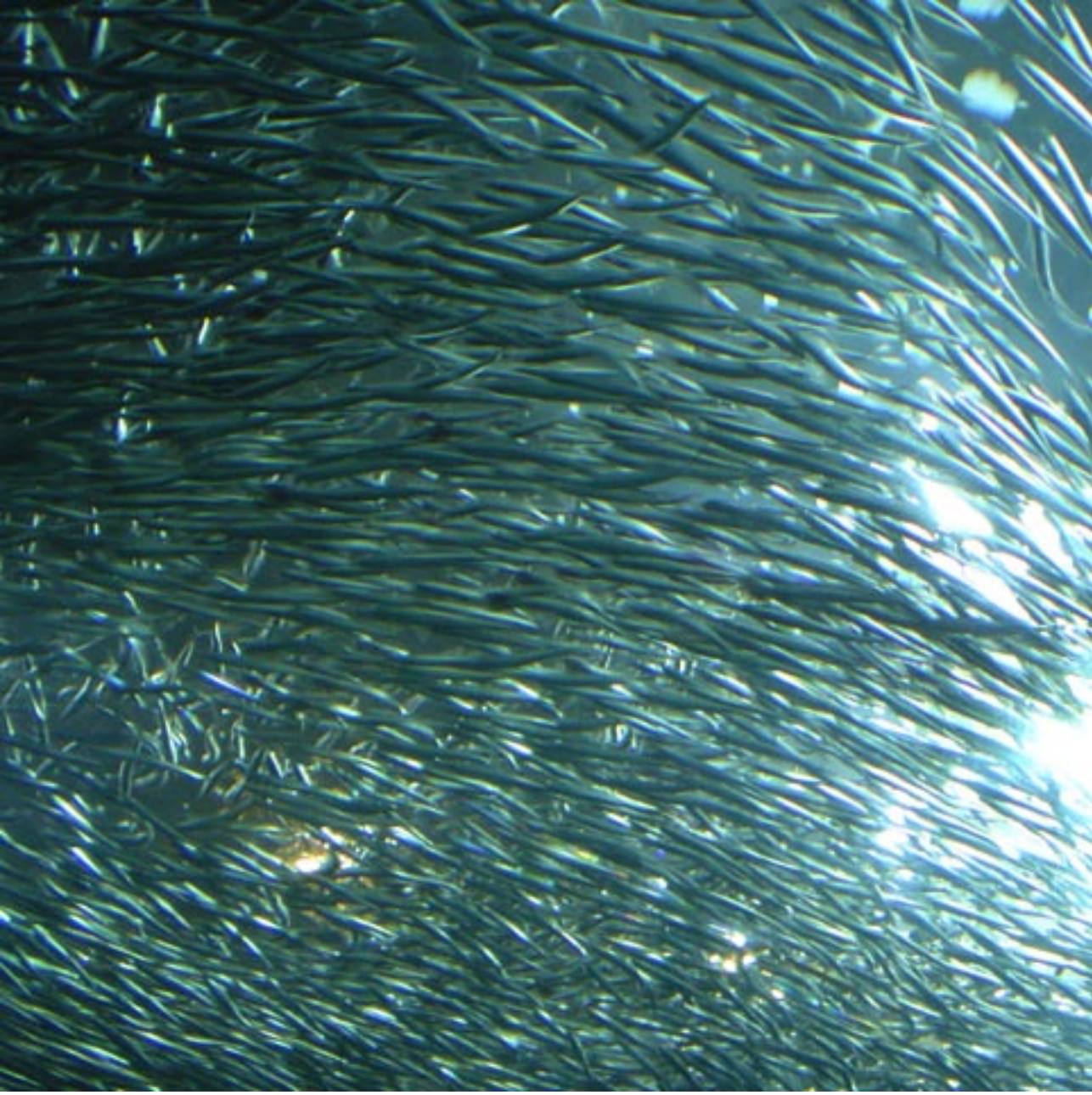
Peter Fischli & David Weiss : Der Lauf Der Dinge, 16 mm (1987).
http://www.tcfilm.ch/lauf_txt_e.htm

Joseph Beuys : Capri Battery (1985).
http://en.wikipedia.org/wiki/Joseph_Beuys

Forme dynamique, geste, comportement, connectivité.

Vol d’oiseau.
Craig Reynolds : Boids Software (1986).
http://www.red3d.com/cwr/boids/index.html

Vol d’oiseau.
Craig Reynolds : Boids Software (1986).
http://www.red3d.com/cwr/boids/index.html



Banc de poisson.
Voir aussi Craig Reynolds : OpenSteer (2003).
http://opensteer.sourceforge.net/

Chenille.

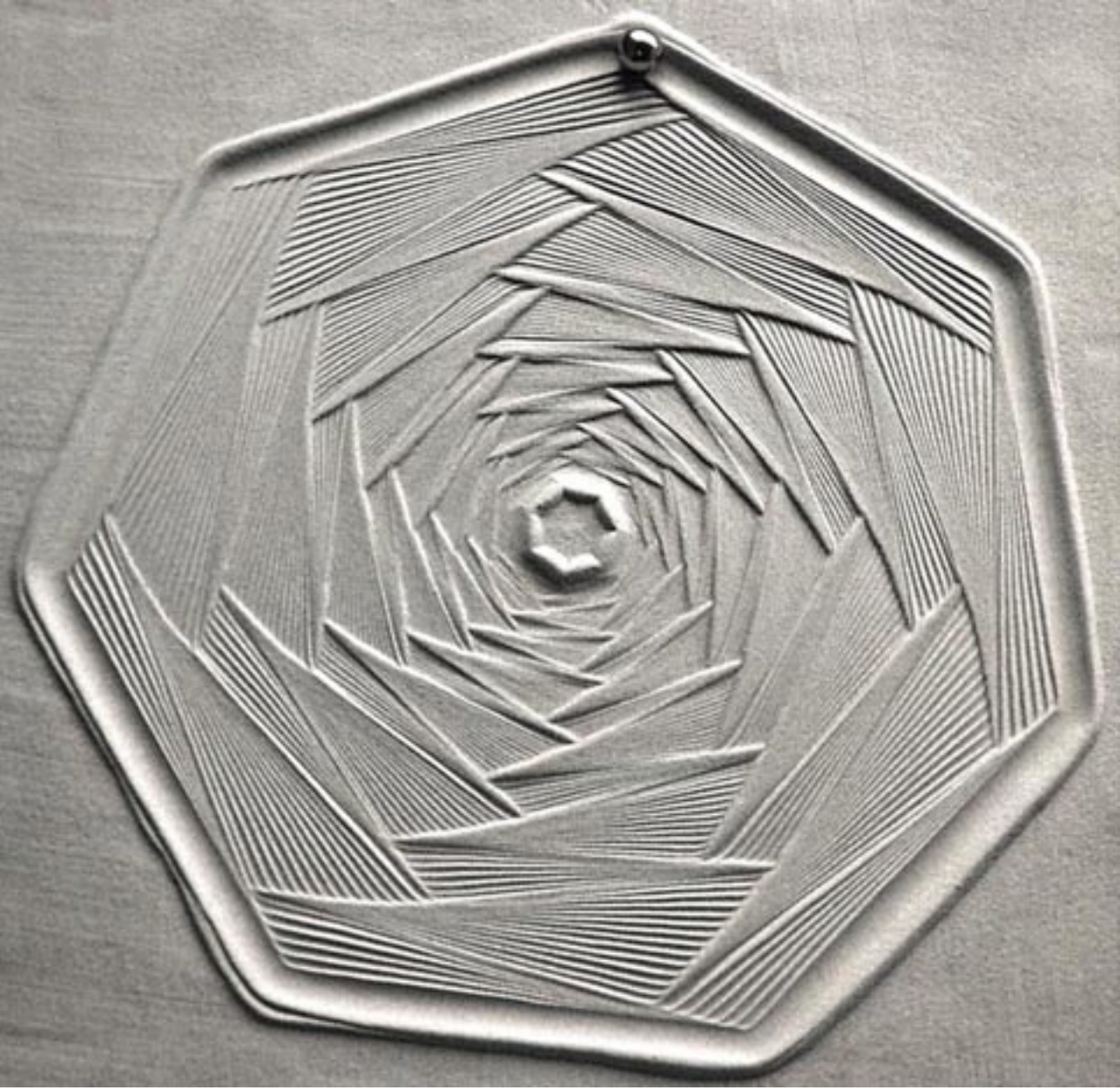
Alvar Aalto : Paravent (1935-1936).
http://www.dmk.dk/details/13405/

L’ouragan Katrina.
http://fr.wikipedia.org/wiki/Ouragan_Katrina

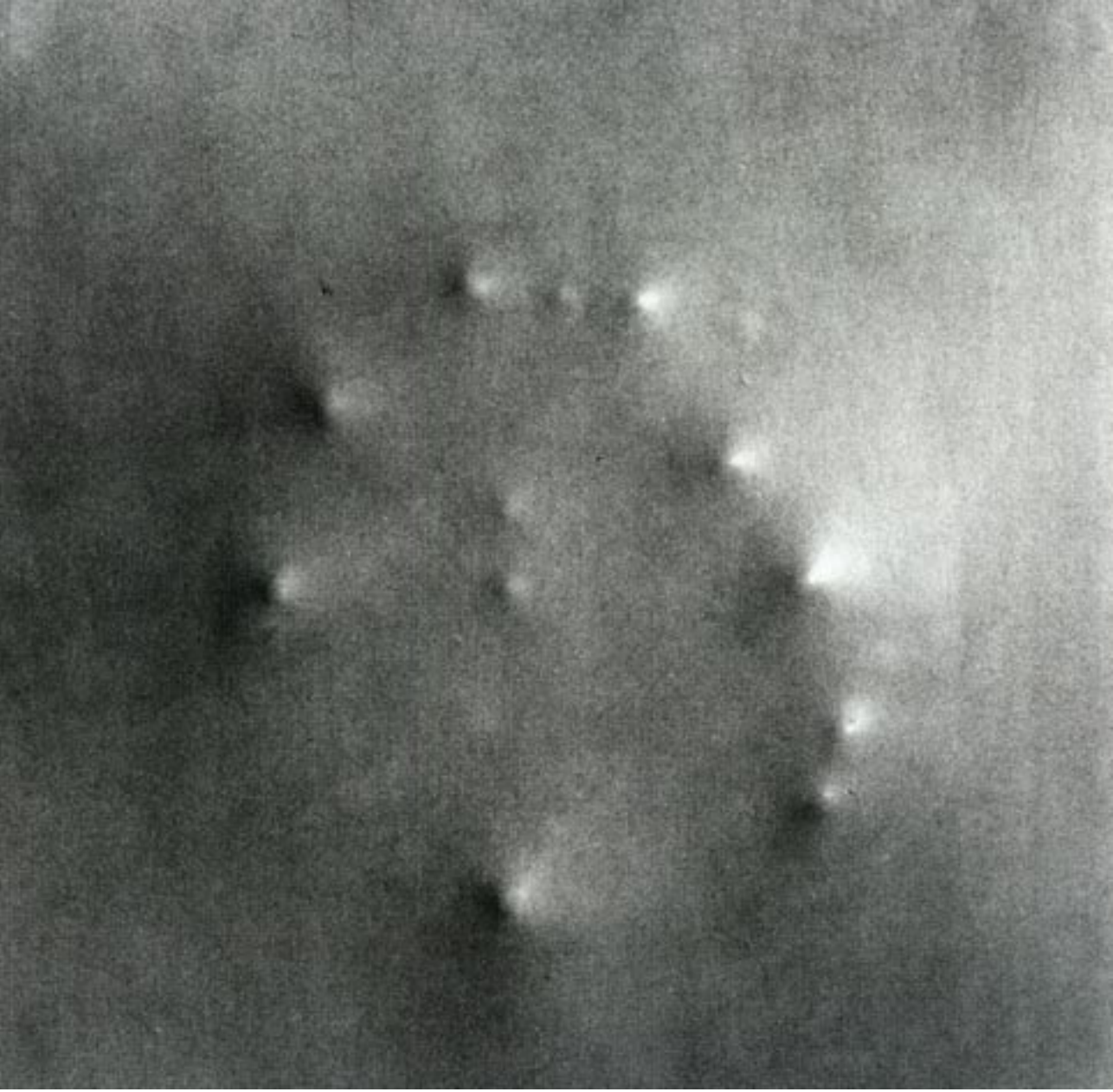
Attraction/magnétisme.

Josef Beuys : Coyote, I like America and America likes me (1974).
http://en.wikipedia.org/wiki/Joseph_Beuys

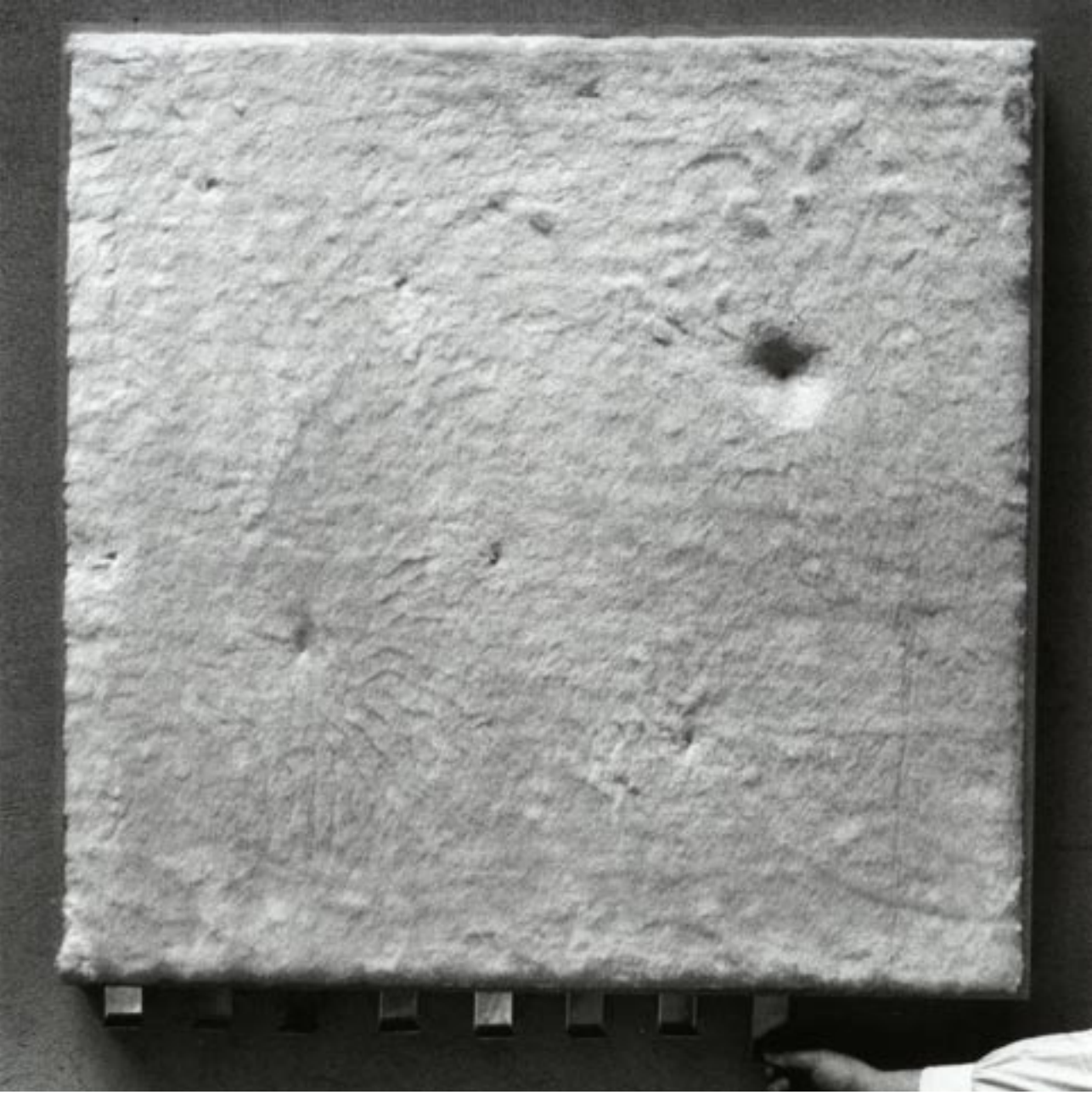
Jardin Zen : le temple de Ryoan-ji à Kyoto.
http://en.wikipedia.org/wiki/Ryoan-ji



Jean-Pierre Hébert : Sand As Medium - Sisyphus (1998),
aimant motorisé, bille, logiciel, sable.
http://hebert.kitp.ucsb.edu/sand/news.html



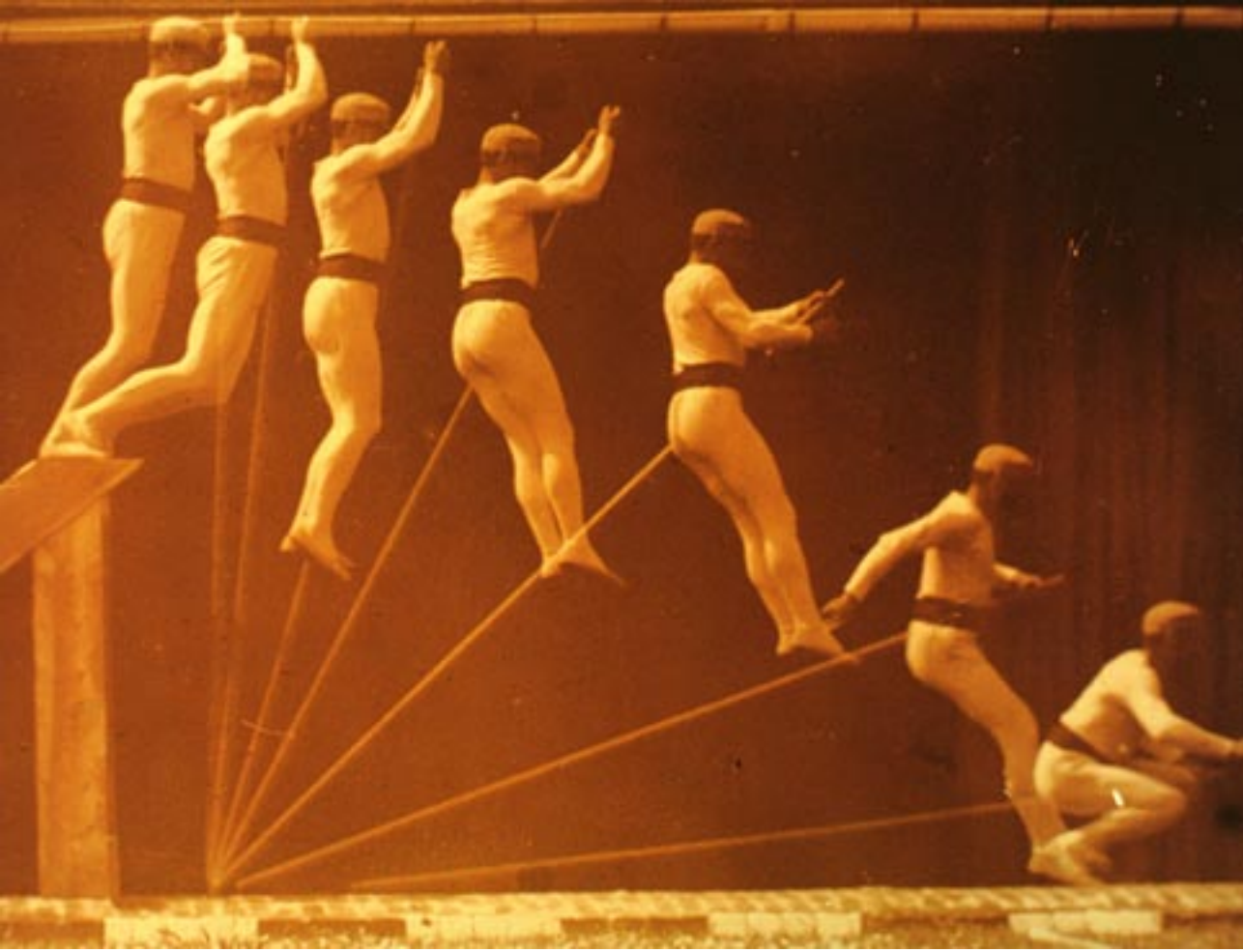
Gianni Colombo : Rilievo Intermutabile (1959).



Gianni Colombo : Superficie in variazione (1959).

MIT Tangible Media Group : Sandscape (2002-2003).
interface tactile, manipulation de sable et modélisation en temps réel.
http://tangible.media.mit.edu/projects/sandscape/

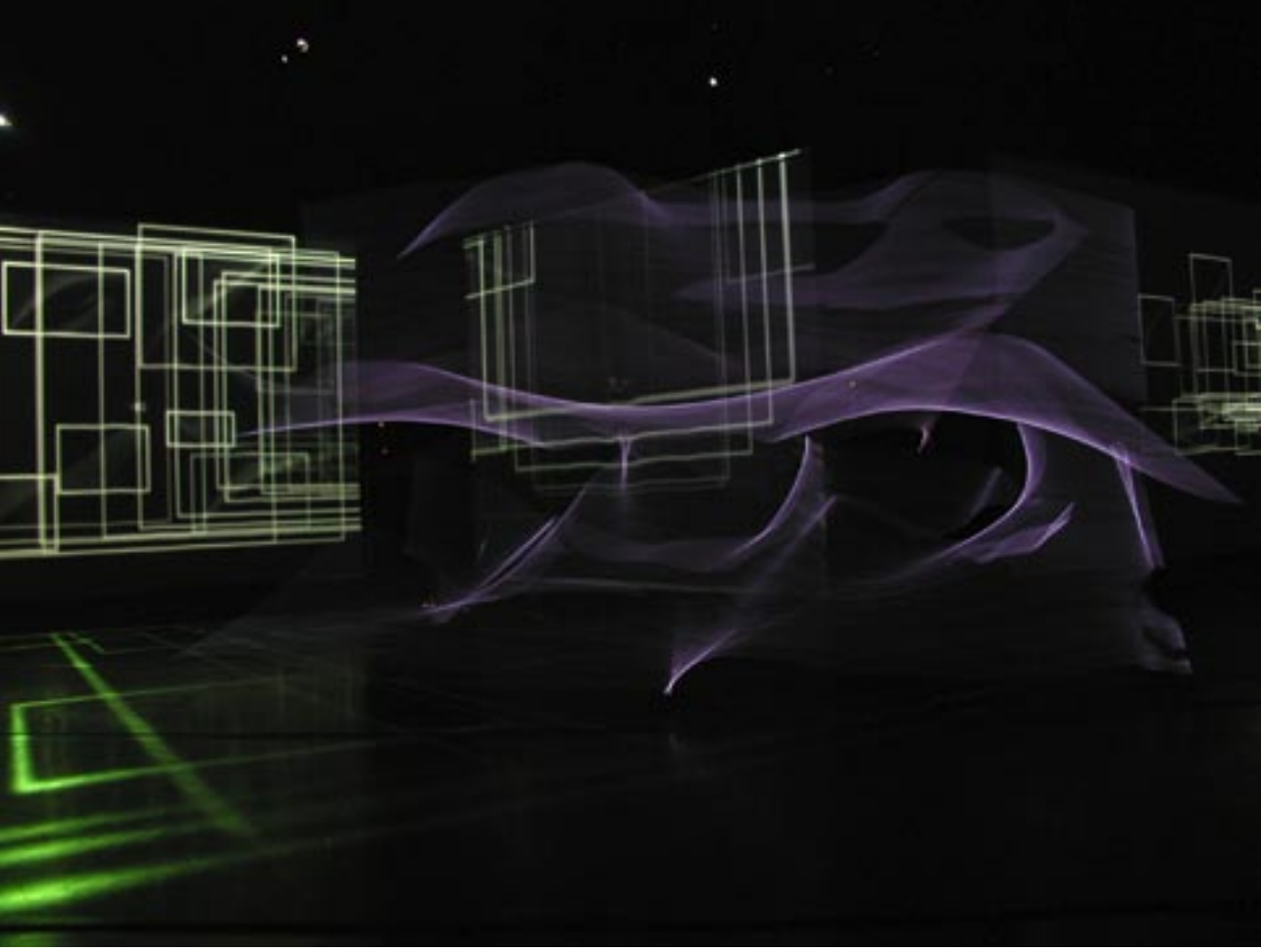
Christa Sommerer & Laurent Mignonneau : Nanoscape (2002),
perception de formes invisibles définies
par un champ magnétique (attraction/répulsion).
http://www.interface.ufg.ac.at/christa-laurent/



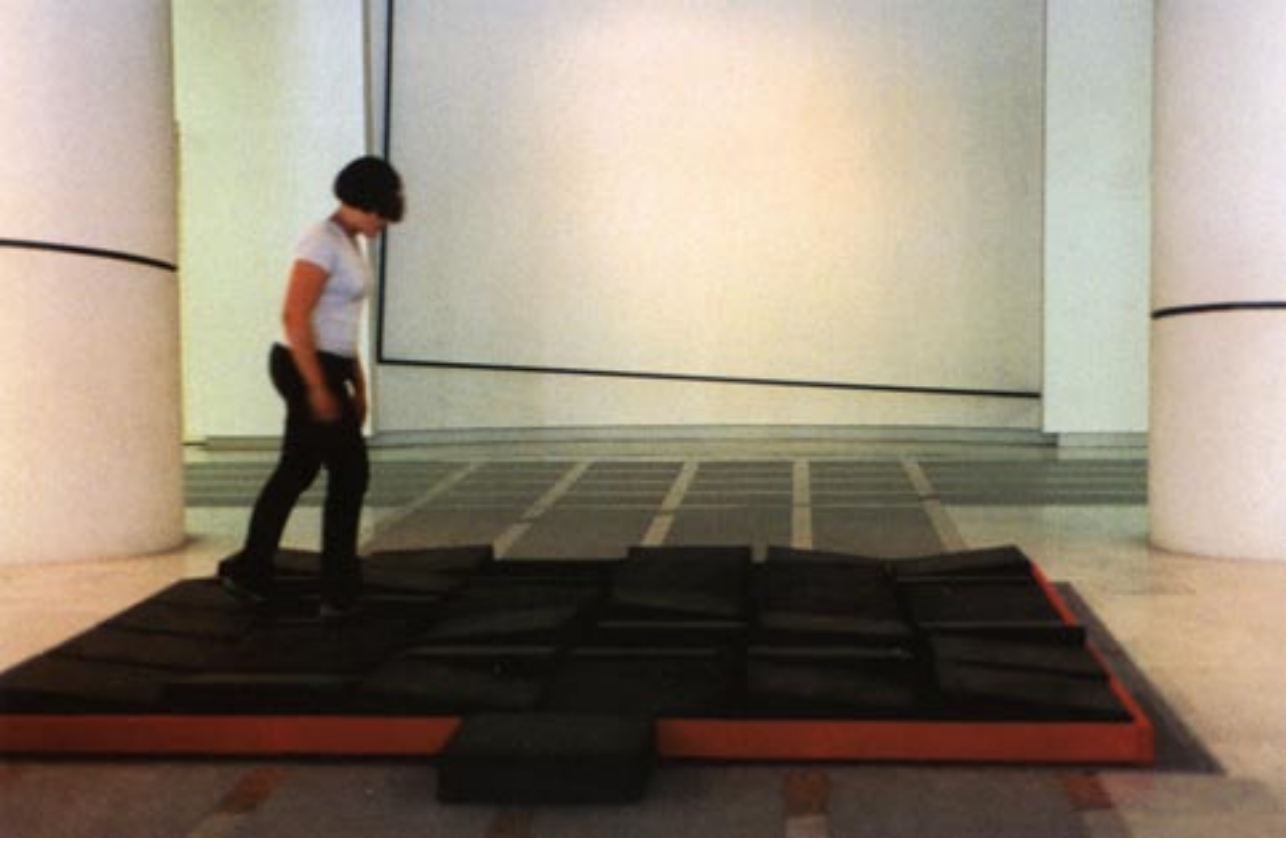
Etienne-Jules Marey : Le saut à la perche (1887-1890).
http://www.expo-marey.com/

David Rokeby : Watch (1995),
détection de mouvement, logiciel, projection.
http://homepage.mac.com/davidrokeby/home.html

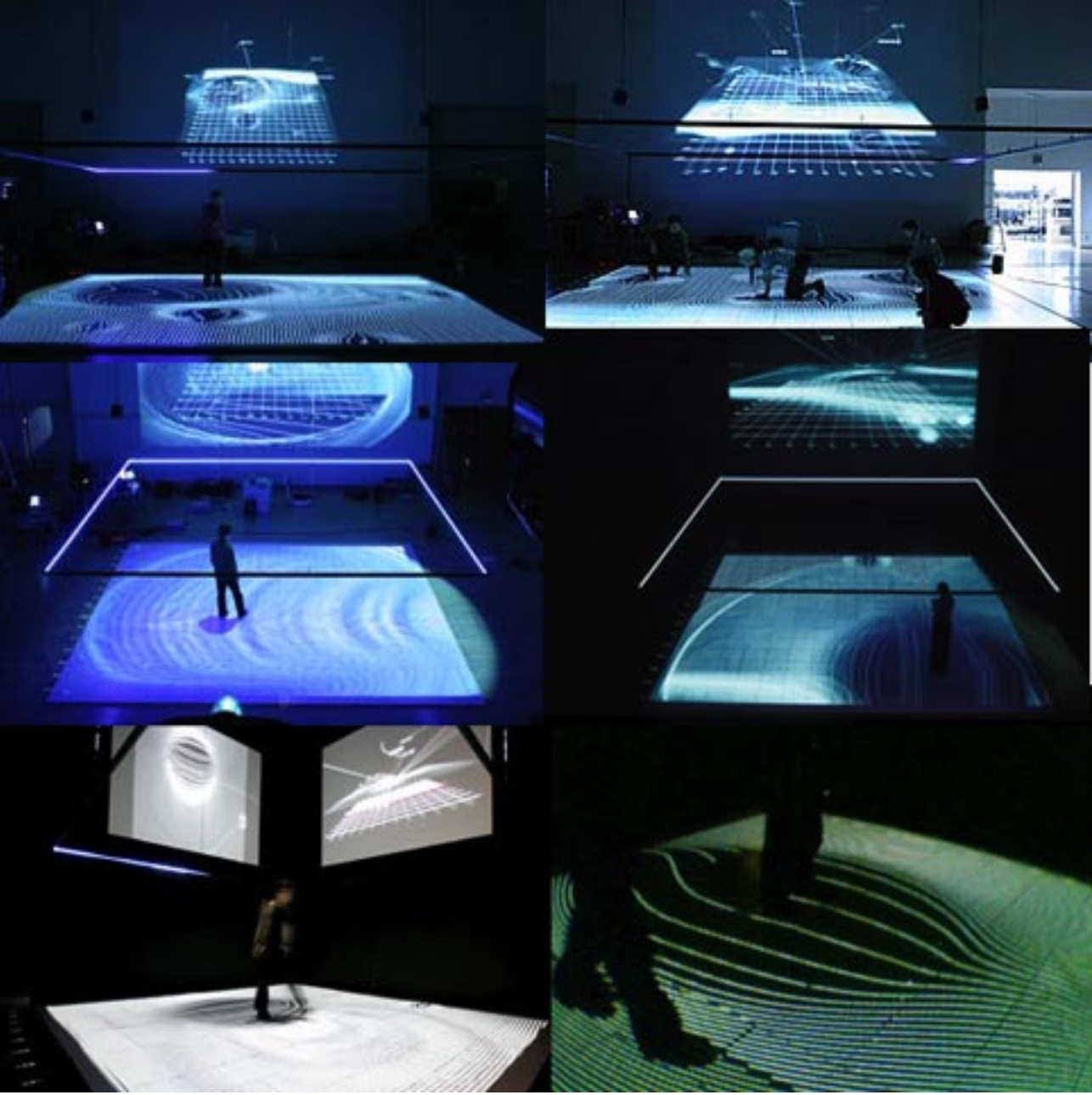
Etienne-Bertrand Weill : Allegro Vivace (1982).



Lab[au] (Laboratory for architecture and urbanism) :
Man in eSPACE.mov (2005-2006),
détection de mouvement, logiciel, projection.
http://www.lab-au.com/



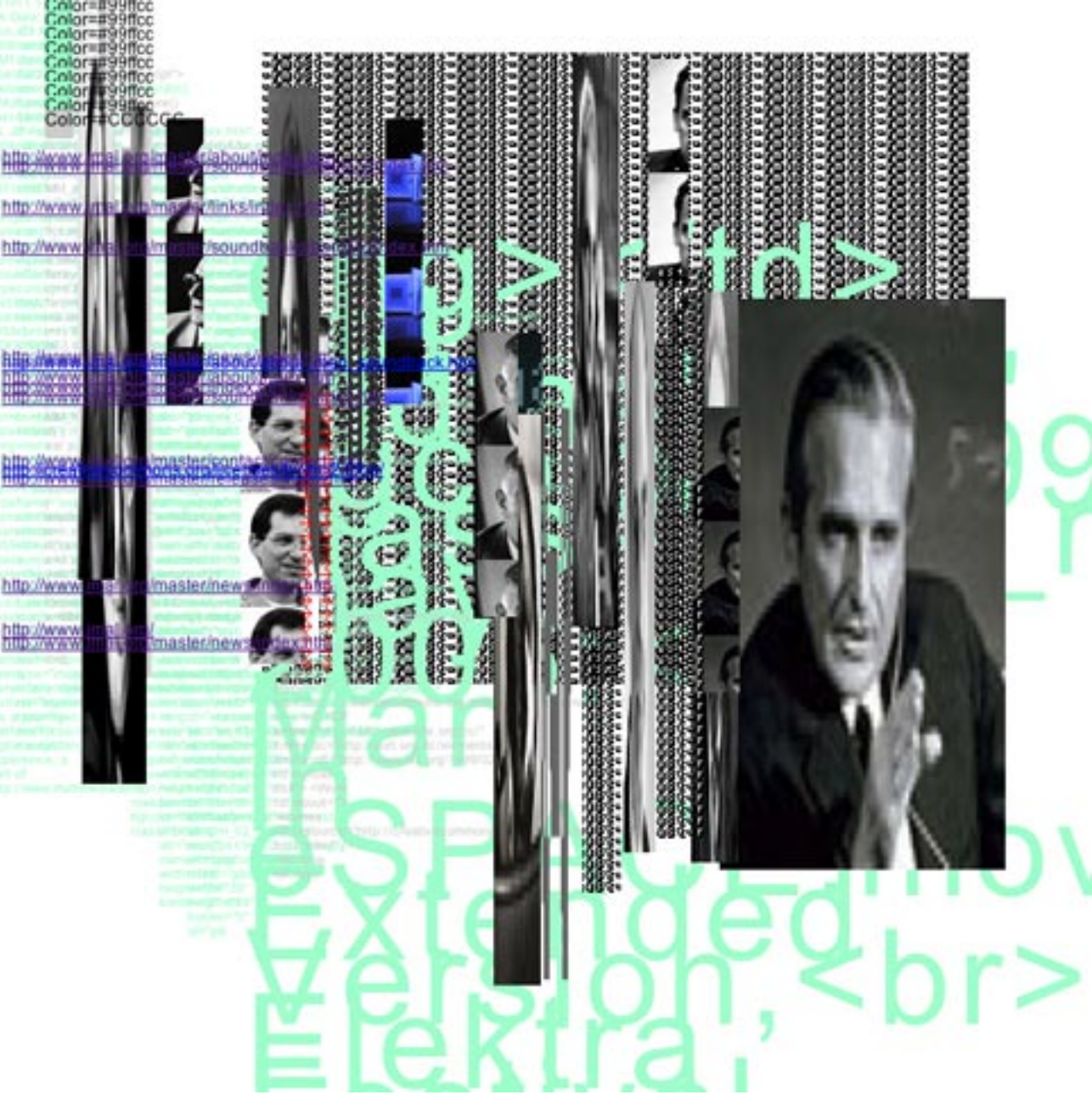
Julio Le Parc : Dalles mouvantes (1964).



Seiko Mikami & Sota Ichikawa : Gravicells - Gravity and resistance (2004), sol réactif, logiciel, projection.
http://www.g--r.com/

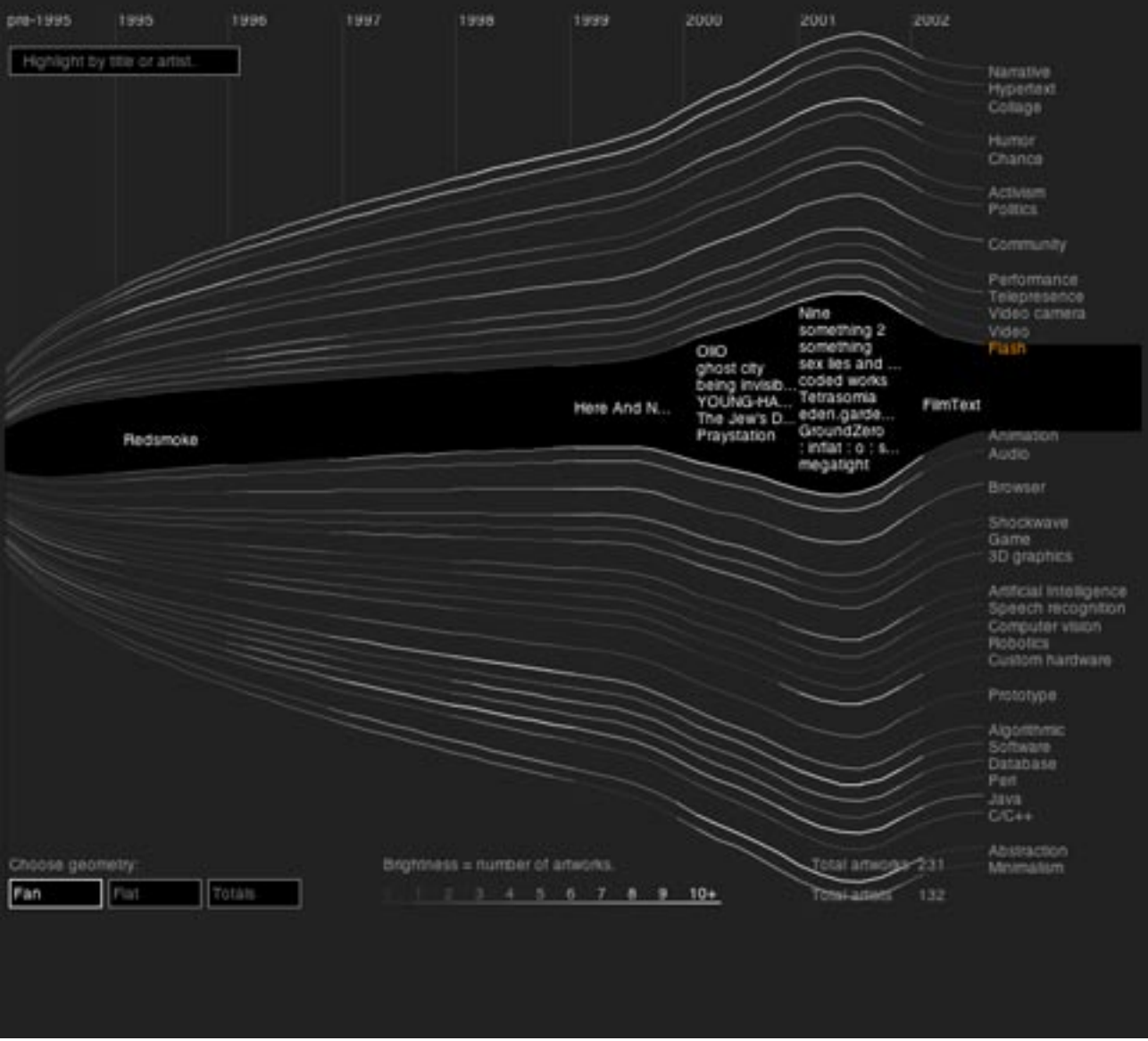
Hanna Haaslahti & Yves Bernard : White Square (2002-2003),
détection de mouvements, logiciel, projection.
http://www.imal.org/WhiteSquare/

David Rokeby : n-Cha(n)t (2001),
machines en réseau, reconnaissance vocale, diffusion synchronisée.
http://homepage.mac.com/davidrokeby/home.html



Mark Napier : Shredder (1998),
altération du code HTML avant sa lecture par le navigateur.
http://potatoland.com/shredder/

Mongrel : Nine(9) - a social software project (2003-2004),
outil dynamique et collaboratif en ligne.
http://www.mongrelx.org/
http://www.linker.org.uk/
http://kop.fact.co.uk/KOP/artists_2003/html/a2.html



Martin Wattenberg : A Net Art Idea Line (2001),
chronologie interactive du net-art.
http://artport.whitney.org/commissions/idealine.shtml

Un logiciel est un objet réactif: il possède la capacité d'interagir dynamiquement avec son environnement.

Cette spécificité soulève certainement des questions de sens, de conception et de réalisation, mais aussi de perception. Elle renouvelle certains aspects de notre rapport à l'art et modifie en tout cas les relations entre art et science. En effet, la notion d'«interaction dynamique» suppose une modélisation de propriétés physiques, une procédure, un traitement de données, l'expression de valeurs, etc... Tout un vocabulaire issu des technologies informatiques au sens large (logique, intelligence artificielle, mathématiques) qui, brutalement, fait irruption dans le domaine de l'art.

Le code est un matériau. Le langage est un matériau. On prend la plume ou la parole pour exprimer des idées ou des points de vue. La grammaire et la réthorique sont des outils propres au langage. Ainsi, l'expression par le code possède ses propres outils : un ordinateur, un environnement de programmation et une syntaxe. Mais il faut surtout des outils conceptuels appropriés qui permettront de manipuler ce code-matériau de manière pertinente.

C'est ici qu'apparaît une possible et surprenante confusion entre le fond (le sens des choses) et la forme (l'apparence des choses). En effet, l'acte de programmation peut être associé à un acte de modélisation : modéliser une forme, modéliser un comportement, modéliser une action, etc.

Edmond Couchot* l'explique clairement :
«[...] techniquement, l'image numérique est étroitement dépendante des processus programmatiques qui la produisent. Or ces modèles de simulation numérique utilisés dans les programmes sont, comme tout modèle scientifique, déjà des interprétations formalisées du réel. [...] Il en résulte que sur un écran d'ordinateur on ne peut figurer, donner une forme visible, sensible, qu'à ce qui est déjà intelligible, déjà interprétation rationnelle du monde. Les artistes se trouvent alors dans la délicate nécessité de créer du sensible (des formes artistiques) avec de l'intelligible (des programmes informatiques), en quelque sorte des résidus applicatifs de la science [...]».

Cette réalité fondamentale de la programmation va nécessiter beaucoup de vigilance pour garder du recul sur le sens et les motivations d'un travail artistique.

** Lire l'extrait «La science comme présence efficiente» en annexe.*

Dans une conférence* au festival Ars Electronica 2003 (Linz, Autriche), Casey Reas citait six aspects spécifiques («noyaux d'expressions numériques») à la réactivité et aux processus d'un logiciel, c'est-à-dire six propriétés exploitables lors de la conception d'un programme :

Forme dynamique:
une forme réactive, qui réagit à des stimuli externes en se reconfigurant.

Geste :
La capacité de transcrire et d'interpréter un geste.

Comportement :
un mouvement possédant l'apparence d'une intention.

Simulation :
la simulation de phénomènes physiques (gravité, vitesse, etc).

Auto-Organisation :
la capacité des éléments à s'auto-structurer, à se régénérer.

Adaptation :
une capacité à changer, à s'adaptater, induisant que le logiciel possède une représentation de lui-même et une compréhension de son contexte.

** Voir en annexe son texte «Code - The Language of our Time».*

Je retiendrai ici les trois premières notions, qui me semblent être les plus génériques. Sans pour autant les minimiser, *simulation, auto-organisation* et *adaptation* traduisent des singularités de *comportement* (souvent cumulées) qui nécessitent une solide expertise en programmation et qui pourront ensuite faire l'objet d'une recherche avancée.

J'ajouterai cependant la notion de «connectivité» qui illustre la capacité d'un logiciel à se connecter à d'autres logiciels, incluant la notion de communauté.

Récapitulons :

Forme dynamique:
une forme réactive, qui réagit à des stimuli externes en se reconfigurant.

Geste :
La capacité de transcrire et d'interpréter un geste.

Comportement :
un mouvement possédant l'apparence d'une intention.

Connectivité :
la capacité d'un logiciel à se connecter à d'autres logiciels

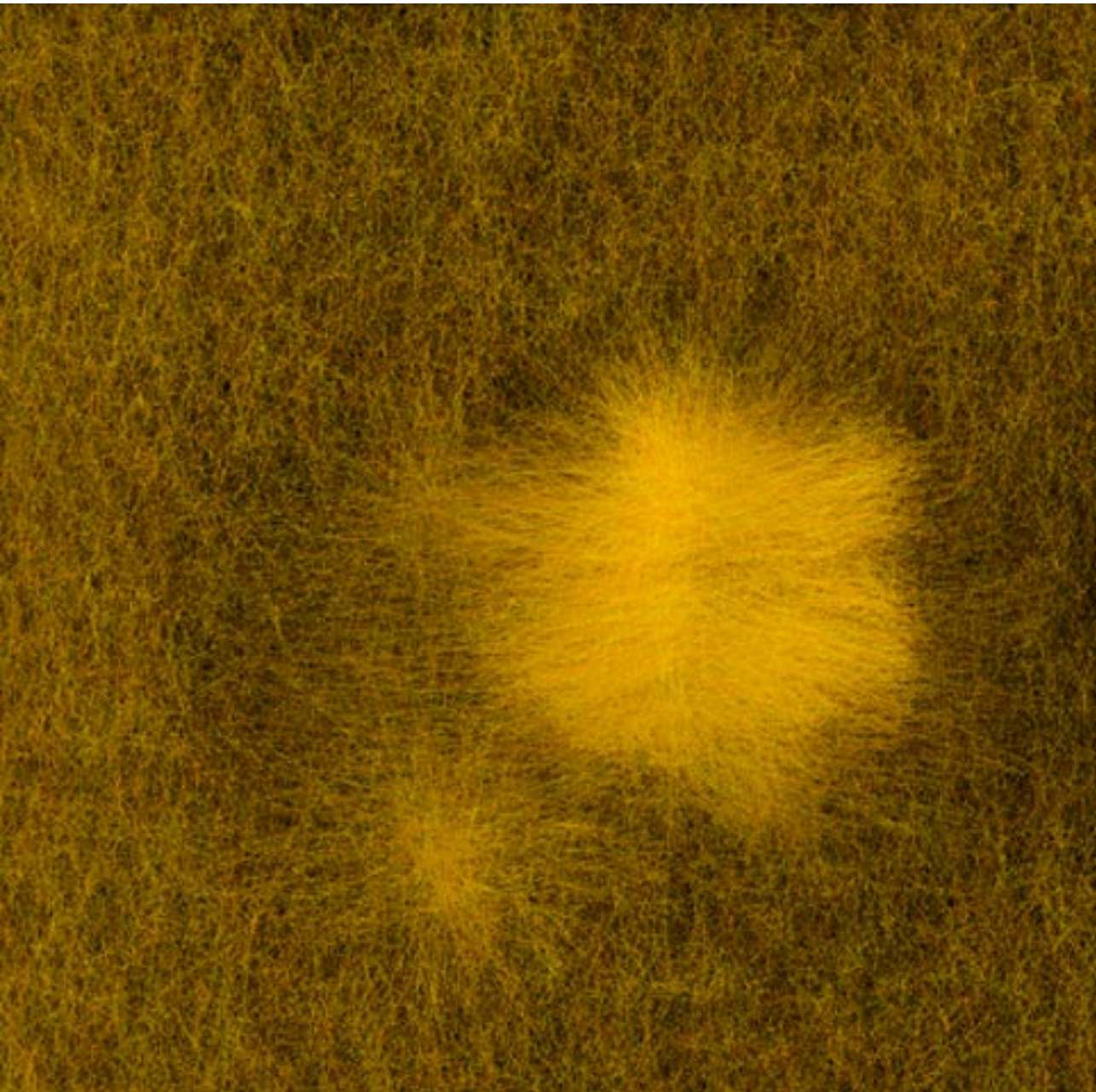
Pour illustrer ces propriétés, j'ai choisi des exemples «complexes» mais bien documentés , suivis d'exemples «simples» que nous examinerons ensemble.

Pour ce qui est de l'installation/performance de LAB[au], il s'agit d'un projet développé avec le logiciel «sPACE» développé par LAB[au], peu documenté en ligne, mais que je connais particulièrement bien.

Forme dynamique :

Keith Peters : Ring of fire, un nuage de particules qui se reconfigure et évolue selon les mouvements et les clics.

http://www.bit-101.com/p5/particles/ringoffire/applet/



Exemple simple de forme dynamique:

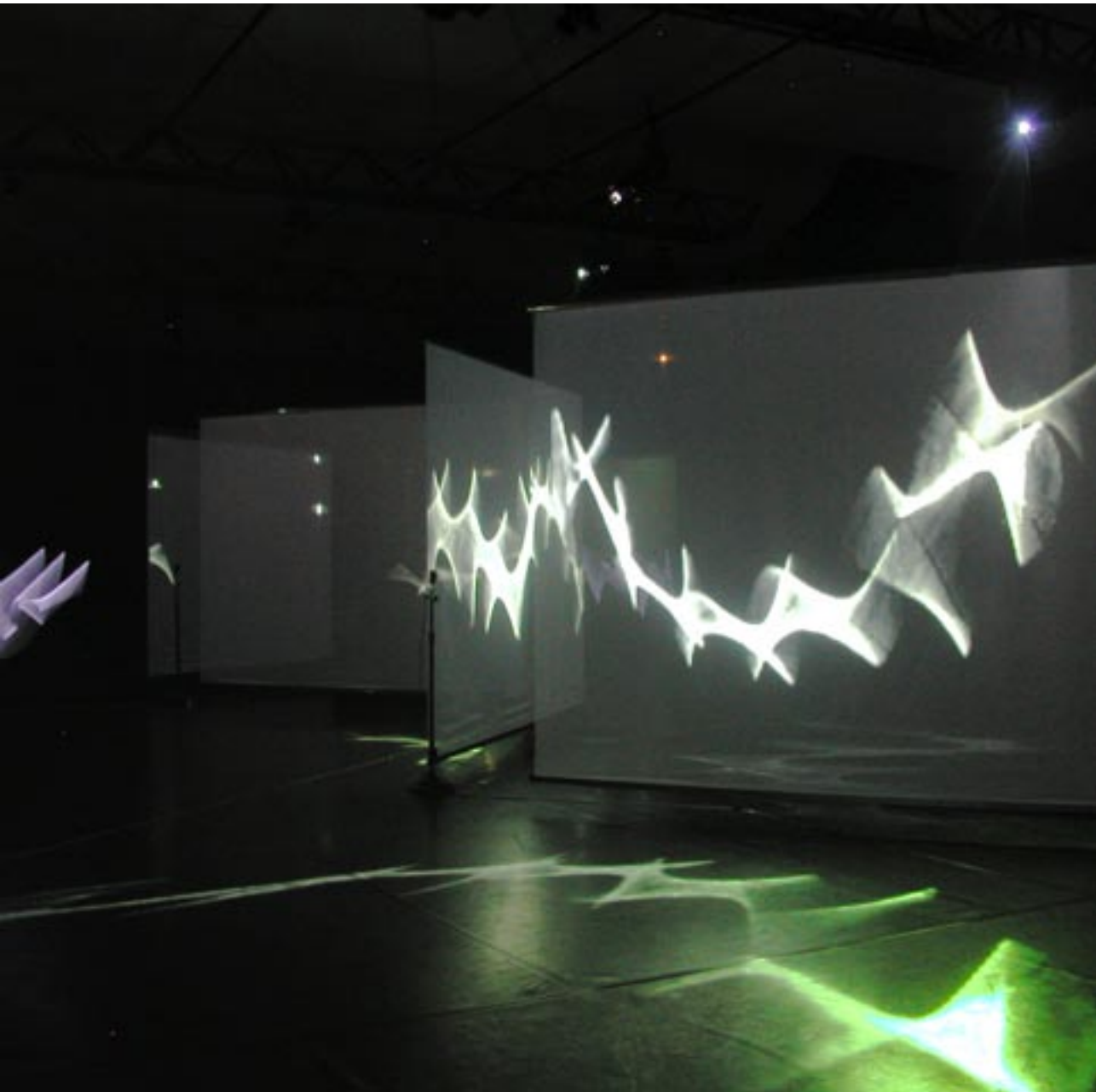
```
// Mouse 2D by REAS
// http://reas.com
// Moving the mouse changes the position and size of each box.
// Updated 21 August 2002

void setup()
{
  size(200, 200);
  noStroke();
  colorMode(RGB, 255, 255, 255, 100);
  rectMode(CENTER);
}

void draw()
{
  background(51);
  fill(255, 80);
  rect(mouseX, height/2, mouseY/2+10, mouseY/2+10);
  fill(255, 80);
  rect(width-mouseX, height/2, ((height-mouseY)/2)+10, ((height-mouseY)/2)+10);
}
```

Voir en ligne:
http://processing.org/learning/examples/mouse2d.html

Geste :

LAB[au] : «Man in eSPACE.mov», installation-performance (2004/2006), captation video, software, projection multi-écrans.

http://www.lab-au.com/
http://www.mast-r.org/



Exemple d’une traduction de geste:

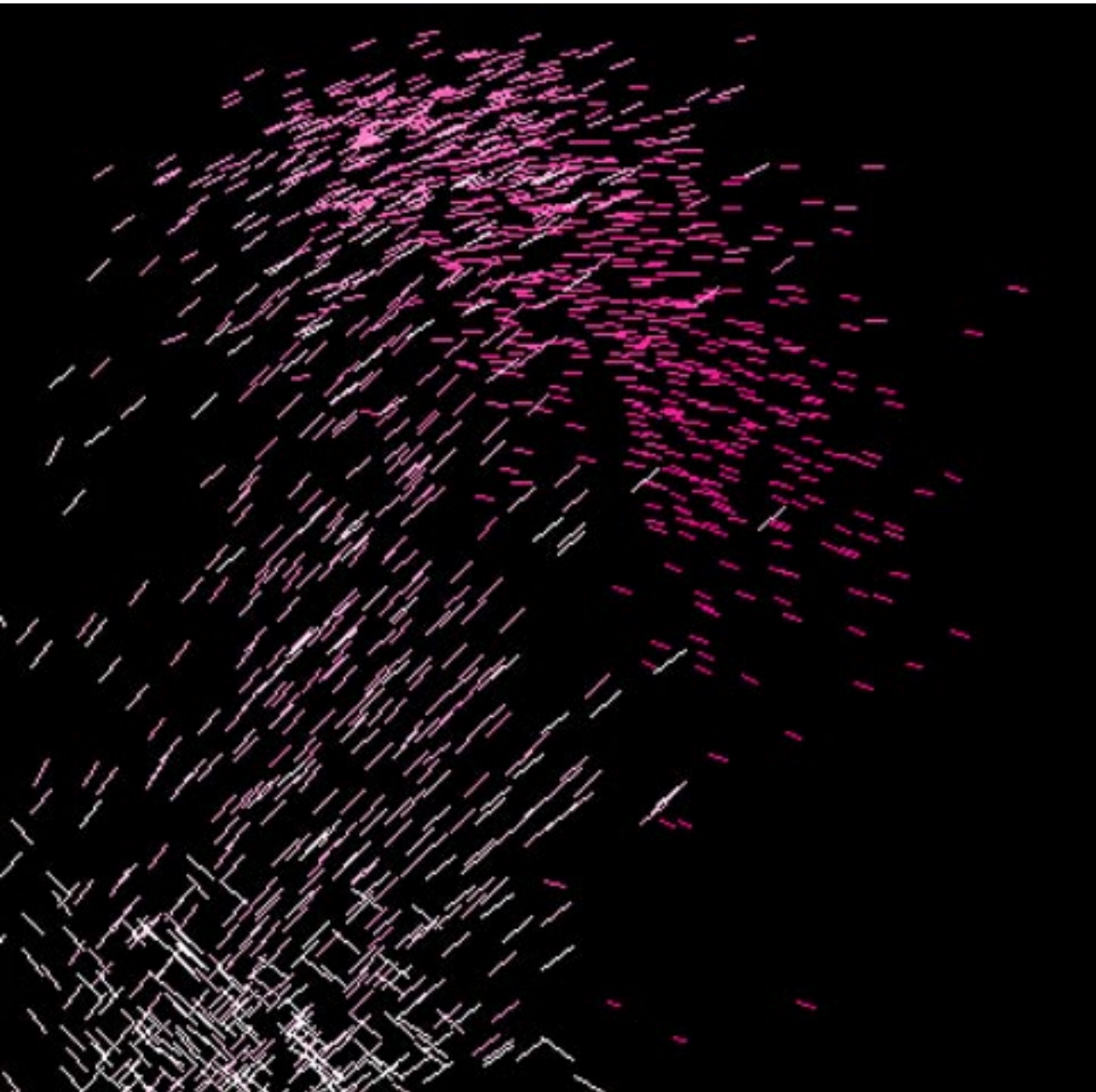
```
// Click by REAS
// http://reas.com
// Move the mouse to position the shape.
// Press the mouse button to invert the color.
// Updated 21 August 2002

int size = 30;

void setup() {
  size(200, 200);
  ellipseMode(CENTER);
  fill(126);
  noStroke();
  rect(0, 0, width, height);
}

void draw() {
  if(mousePressed) {
    stroke(255);
  } else {
    stroke(51);
  }
  line(mouseX-30, mouseY, mouseX+30, mouseY);
  line(mouseX, mouseY-30, mouseX, mouseY+30);
}
```

Voir en ligne:
http://processing.org/learning/examples/click.html

Comportement :

Florito (Markus Graf) : Fly Swarm, créé avec PROCESSING, semble être un essaim d'insectes possédant une inertie (un poids) et une vitesse propres.
http://www.florito.net/FlySwarm/applet/index.html

Voir aussi :
William Ngan : «Pond», créé avec PROCESSING, décrit le comportement d'un groupe de poissons dans un étang face à certains types d'évènements.
http://metaphorical.net/nature/pond_p5.html



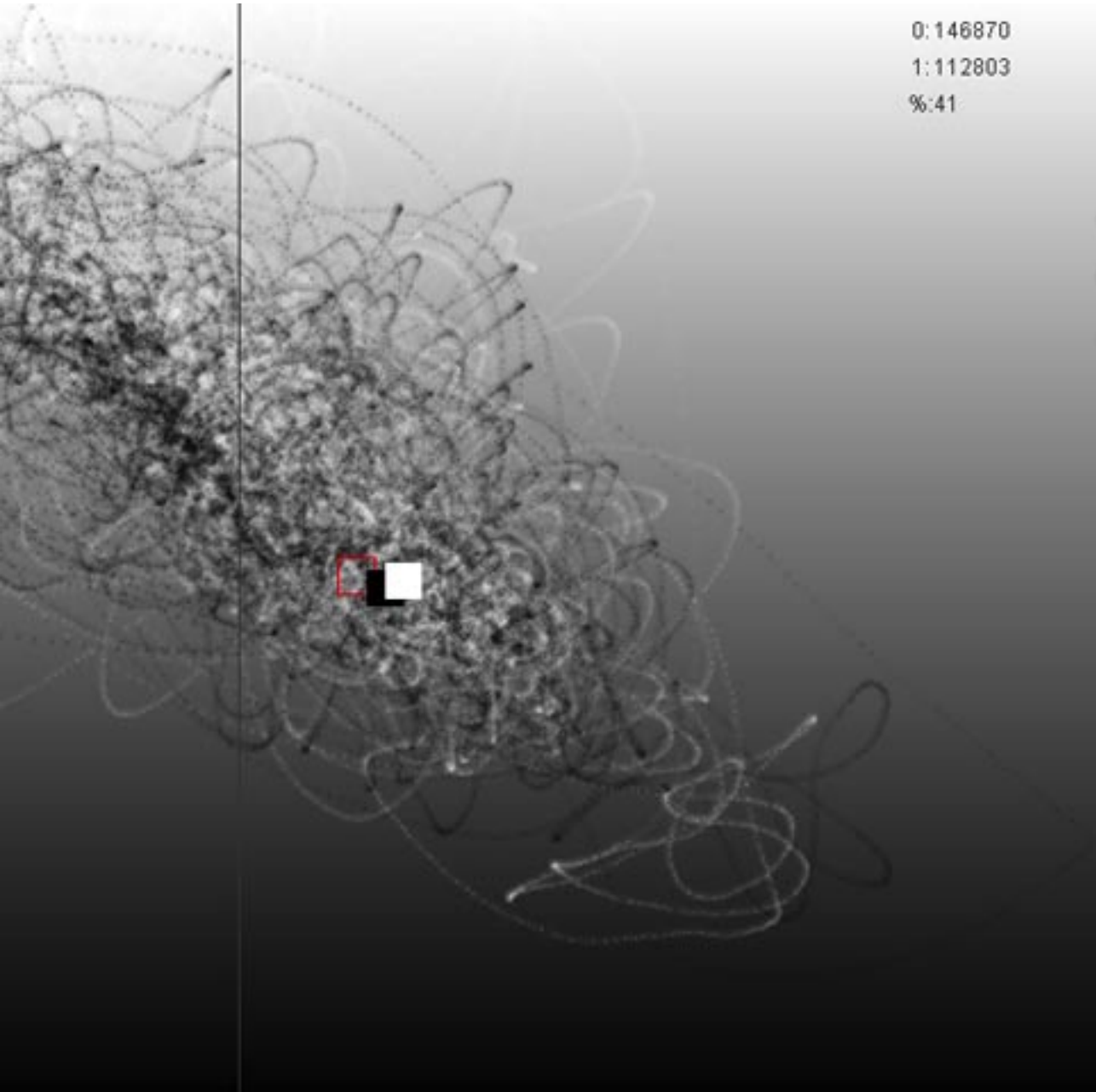
Exemple : contrariété.

```
// déclaration d'une variable globale
int largeur;

// définir les conditions d'affichage
void setup() {
  largeur = 400; // donner une valeur à notre variable
  size(largeur, 400); // taille de la fenêtre
  noStroke(); // pas de trait de contour
  smooth(); // pas de crénelage

}

// mode «dessin»
void draw() {

  int x; // déclaration d'une variable

  background(255, 100, 20);  // le fond annule la trace (supprimez-le pour voir)

  x = largeur - mouseX; // utilisation de la variable
  fill (10, 150, 20); // couleur du cercle
  ellipse(x,mouseY,55,55); // définition d'un cercle
}
```

Connectivité :

Black and white crée un dessin d'après le flux de données binaires de CNN.COM.
Les 0 apparaissent en noir et déplacent le traceur horizontalement.
Les 1 apparaissent en blanc et déplacent le traceur verticalement.
Les deux traceurs s'attirent mutuellement.

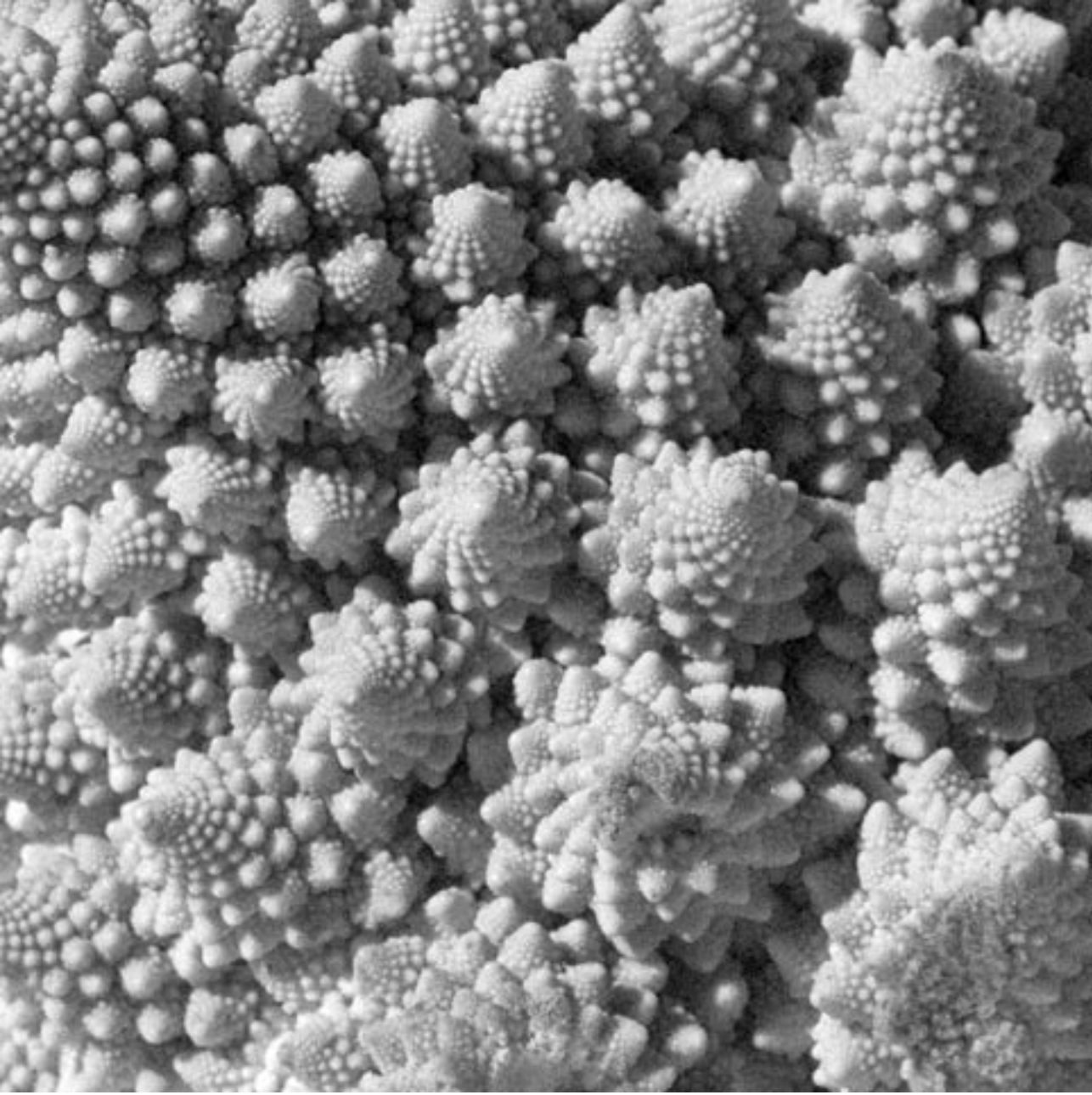
http://www.potatoland.org/blackwhite
http://itserve.cc.ed.nyu.edu/RSG/carnivore/
http://itserve.cc.ed.nyu.edu/RSG/carnivore/processing.php



Exemple : Accès à Internet (affichage d'images provenant d'une webcam).

```
// Laurent Doucet 2005.
// http://siterg.noads.ws/p5/ghost_nyc/index.html

// définir la taille de la fenêtre
void setup() {
size(352,240);
}

// mode «dessin»
void draw() {

// chargement de l'image
PImage a;
a = loadImage(«http://images.earthcam.com/ec_metros/ourcams/fridays.jpg»);

// masque alpha
tint(255, 255, 255, 50);

// position de l'image
image(a,0,0);

}
```

Récursivité.



Fractales : le chou Romanesco.
http://fr.wikipedia.org/wiki/Fractal
http://en.wikipedia.org/wiki/Fractal

Fractales : arbres.
http://fr.wikipedia.org/wiki/Fractal
http://en.wikipedia.org/wiki/Fractal

Fractales : la famille des fougères.
http://fr.wikipedia.org/wiki/Fractal
http://en.wikipedia.org/wiki/Fractal

En informatique, un objet récursif est un objet qui se contient lui-même, ou que l'on définit à partir de lui-même, ou encore une fonction qui s'appelle elle-même.

Les fractales sont des figures récursives.
Une image fractale possède deux propriétés particulières :

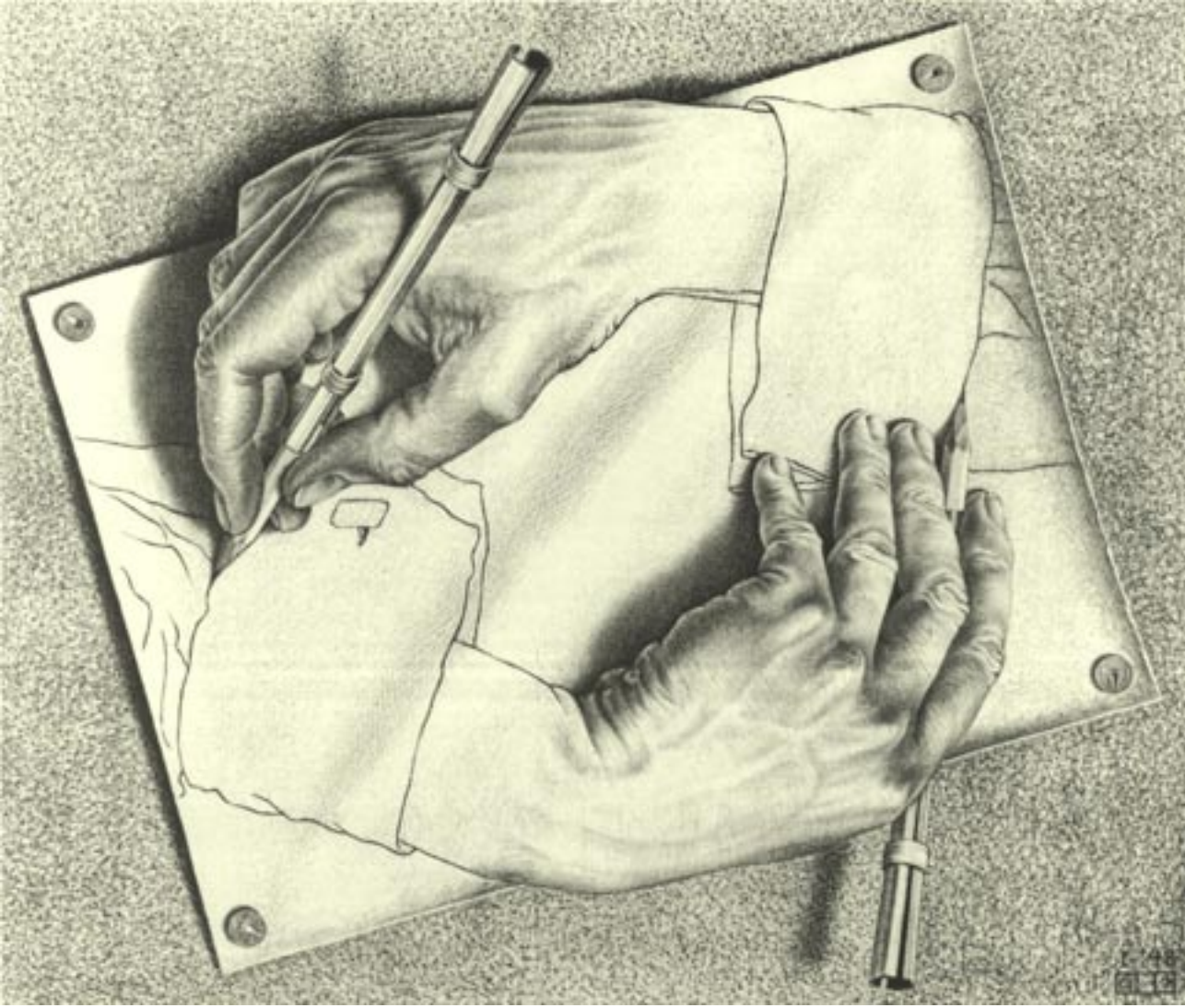
- elle est issue d'un processus itératif (processus en boucle, répété indéfiniment dans lequel le résultat obtenu à une étape du processus est réinjecté dans l'étape suivante),

- elle présente un caractère d'auto-similarité (toutes ses parties, jusqu'à la plus petite, ressemblent au tout, et inversement).

On obtient une image fractale en partant d'un objet graphique auquel on applique une certaine transformation qui ajoute un élément de complexité, puis en appliquant la même transformation au nouvel objet ainsi obtenu, ce qui accroît encore sa complexité... et en recommençant à l'infini ce processus d'itération *.

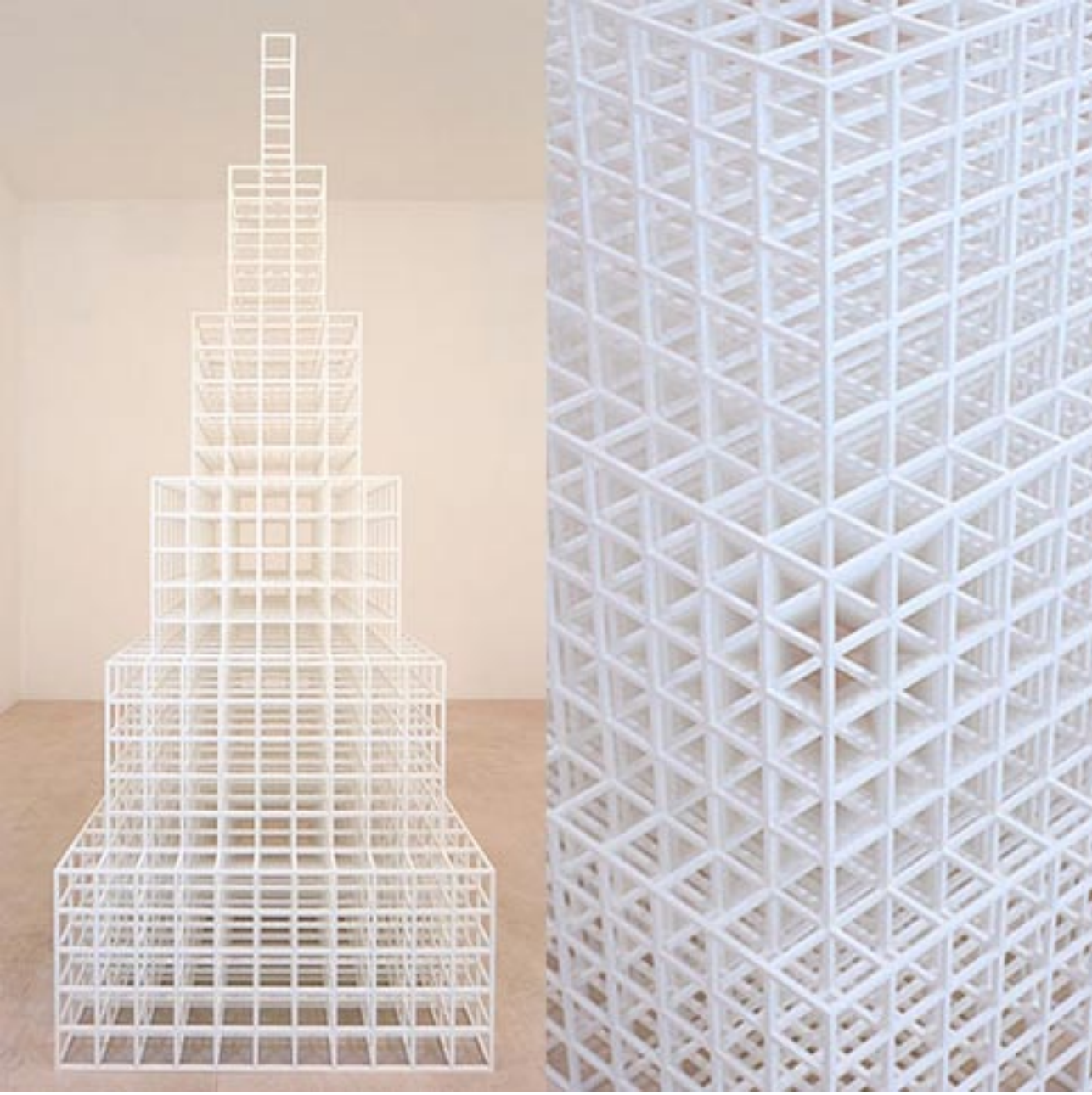
* Itération : séquence d'instructions destinée à être exécutée plusieurs fois (autant de fois qu'on peut en avoir besoin).

Michael Kuehne et Barbara Thompson : Tables gigognes laquées.
http://www.wetter-indochine.com/



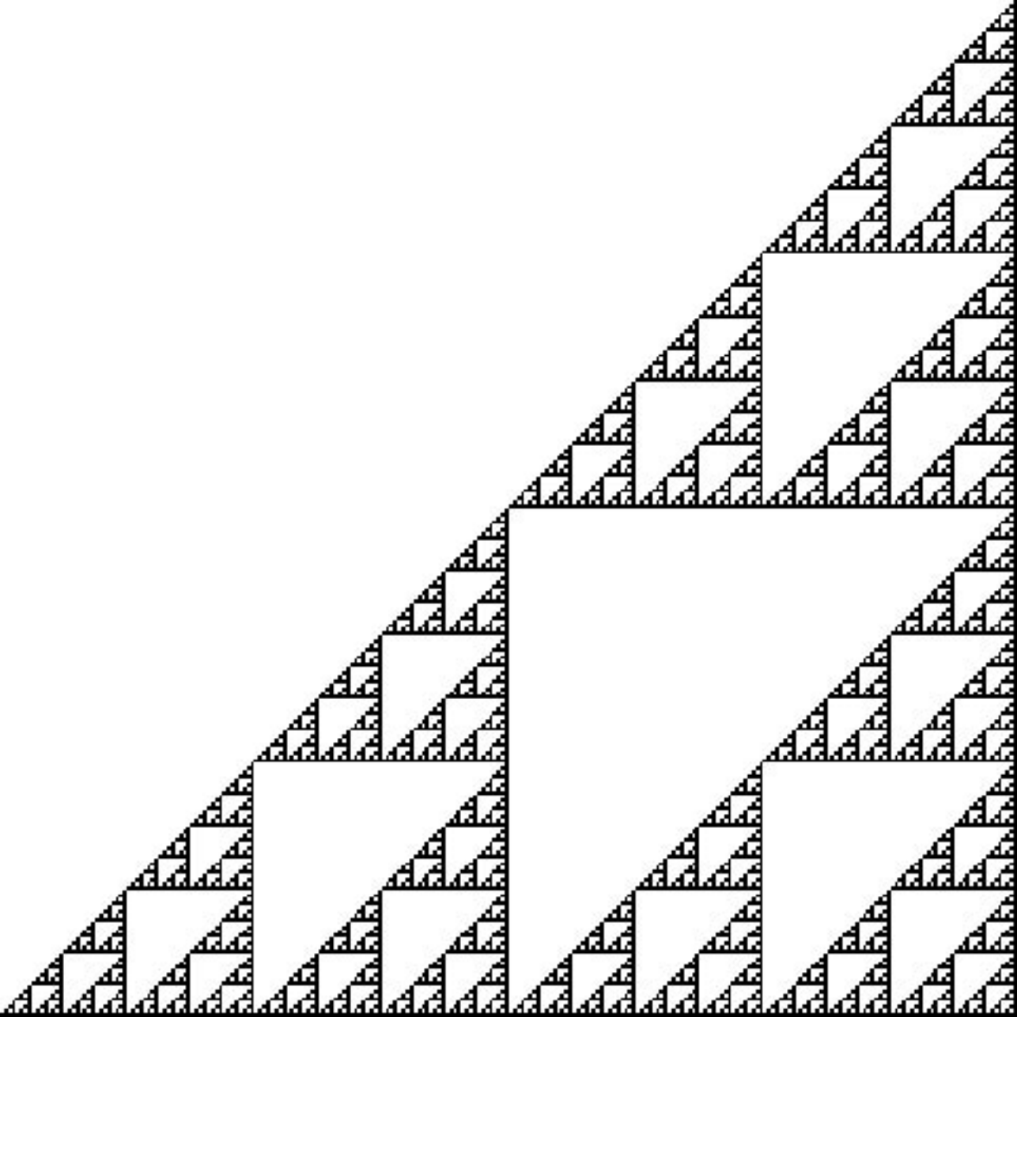
MC Escher : .
http://aixa.ugr.es/escher/table.html

René Magritte : La Clairvoyance (1936).

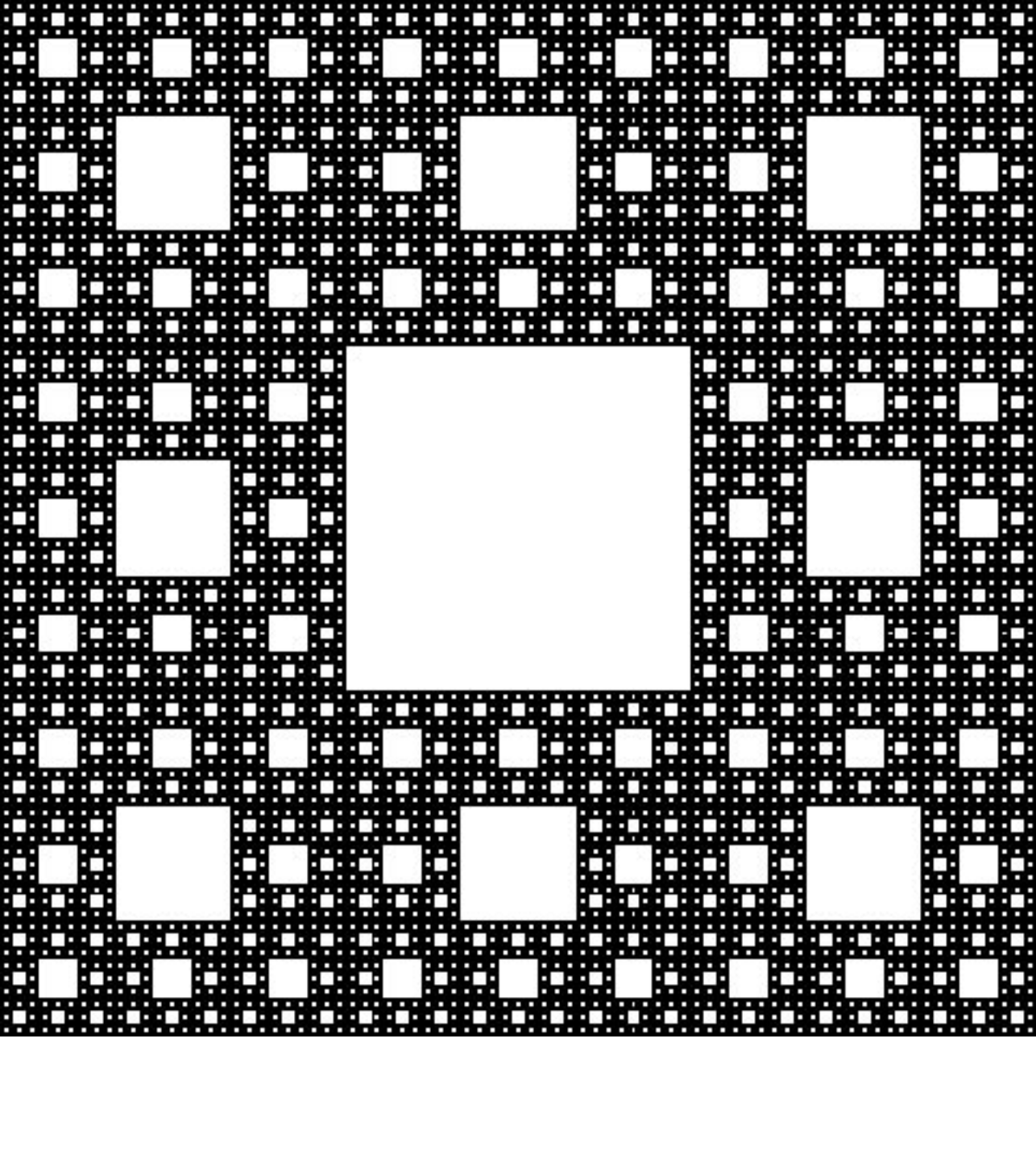


Sol LeWitt : 1357911 (2005).
http://www.galeriepieceunique.com/infoframes/lewitt1.htm

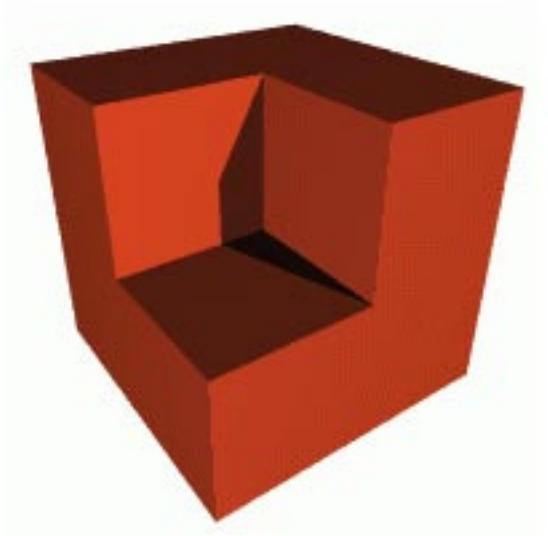
Peter Weir : The Truman Show (1998).
http://en.wikipedia.org/wiki/The_Truman_Show

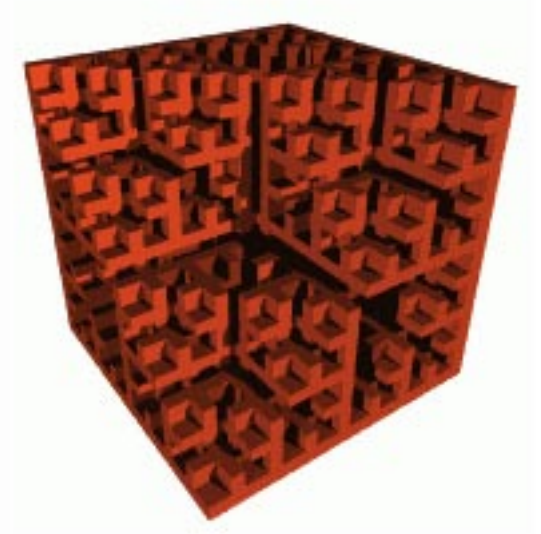


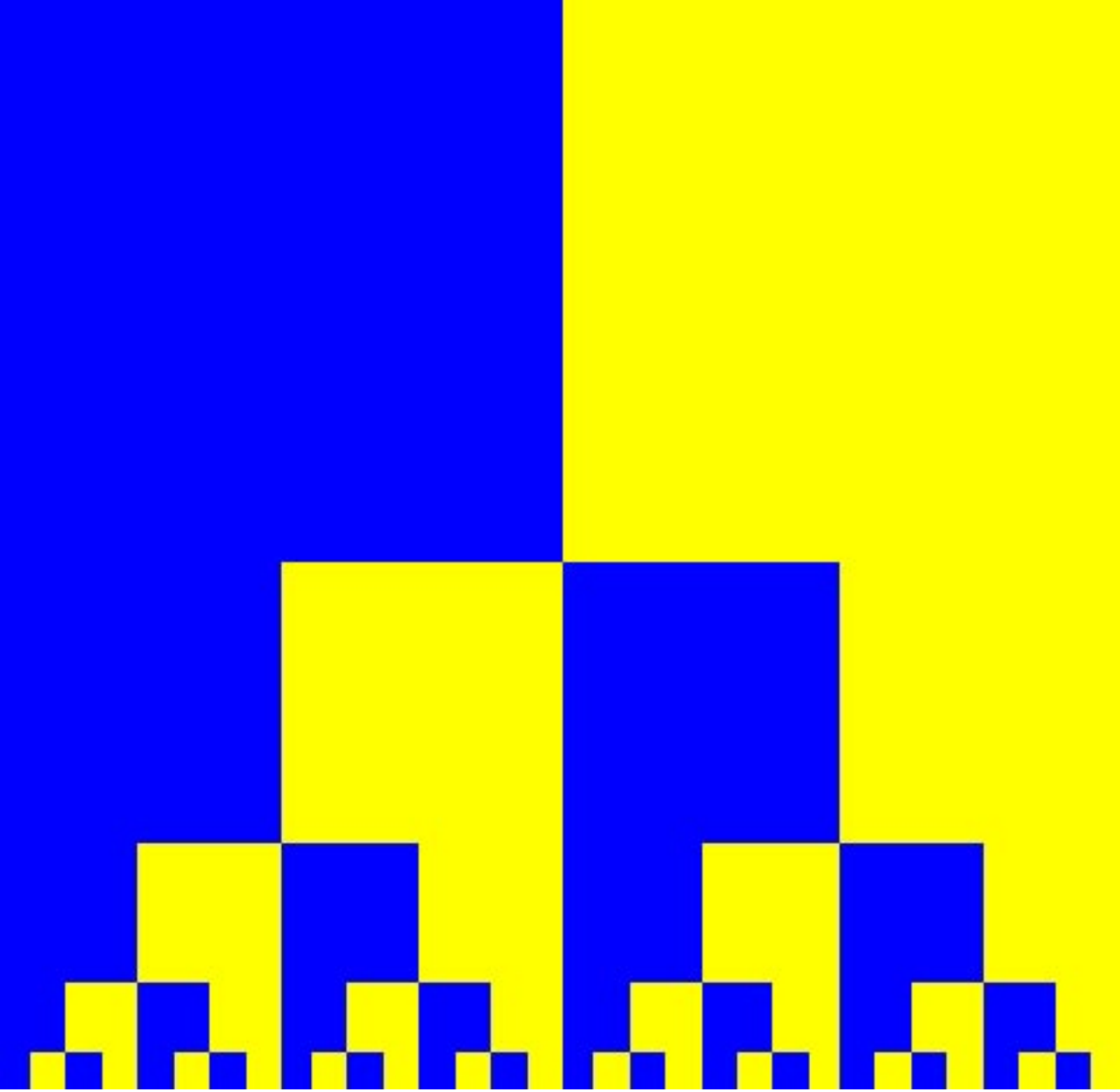
Triangle de Sierpinski.
http://fr.wikipedia.org/wiki/Triangle_de_Sierpinski

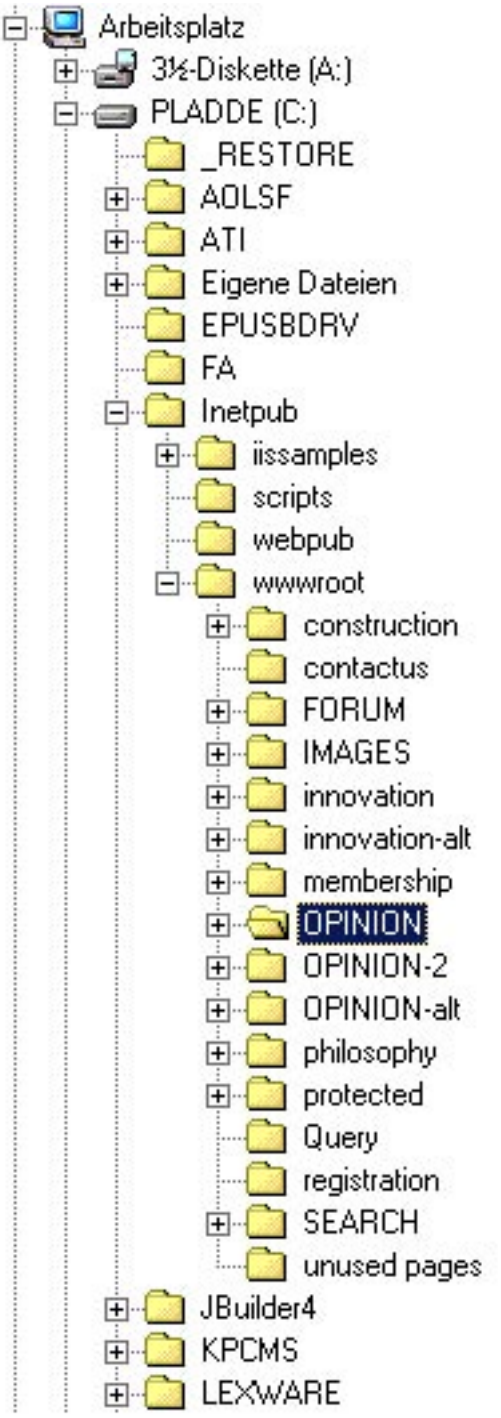


Tapis de Sierpinski.
http://fr.wikipedia.org/wiki/Tapis_de_Sierpinski

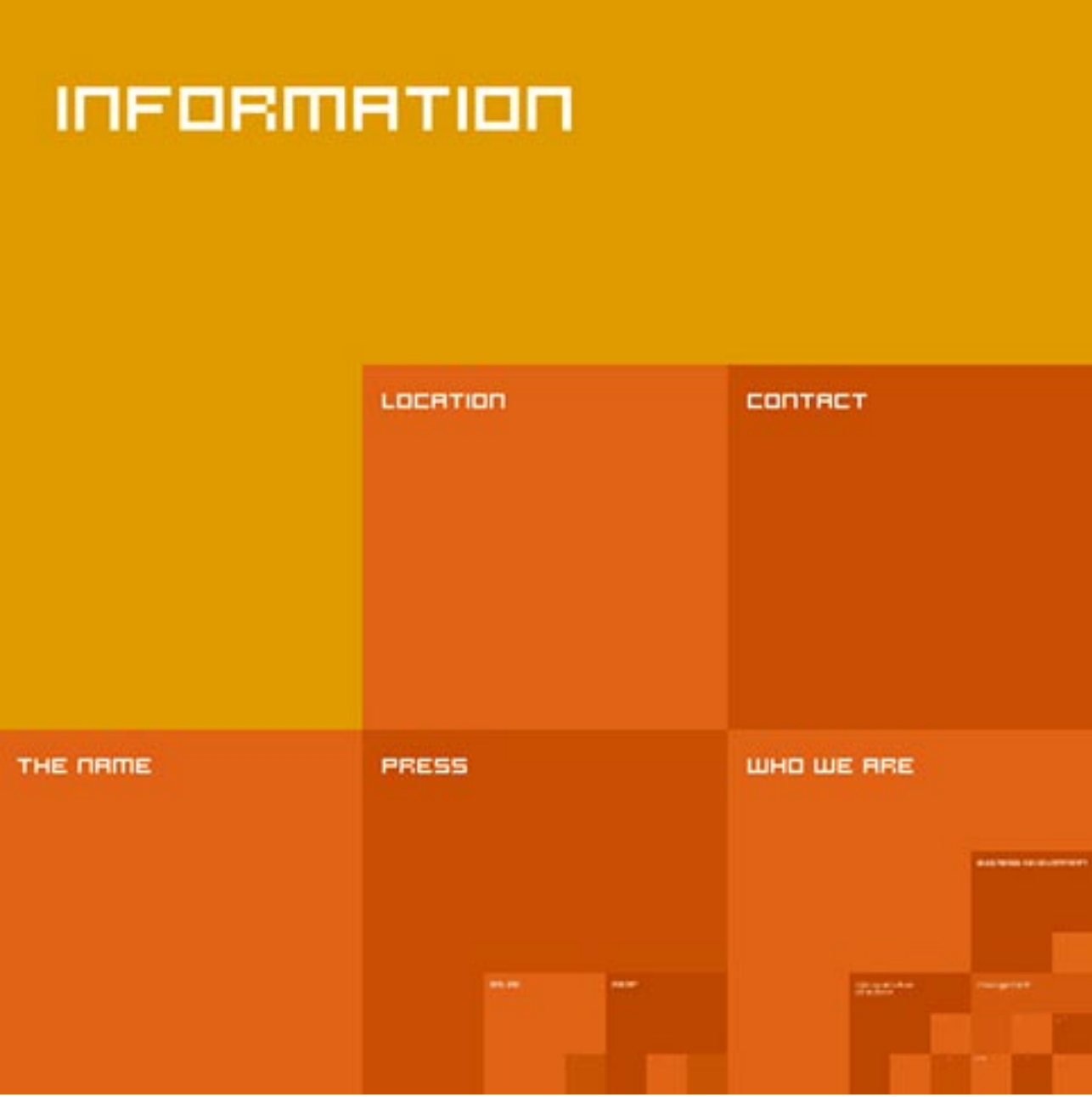








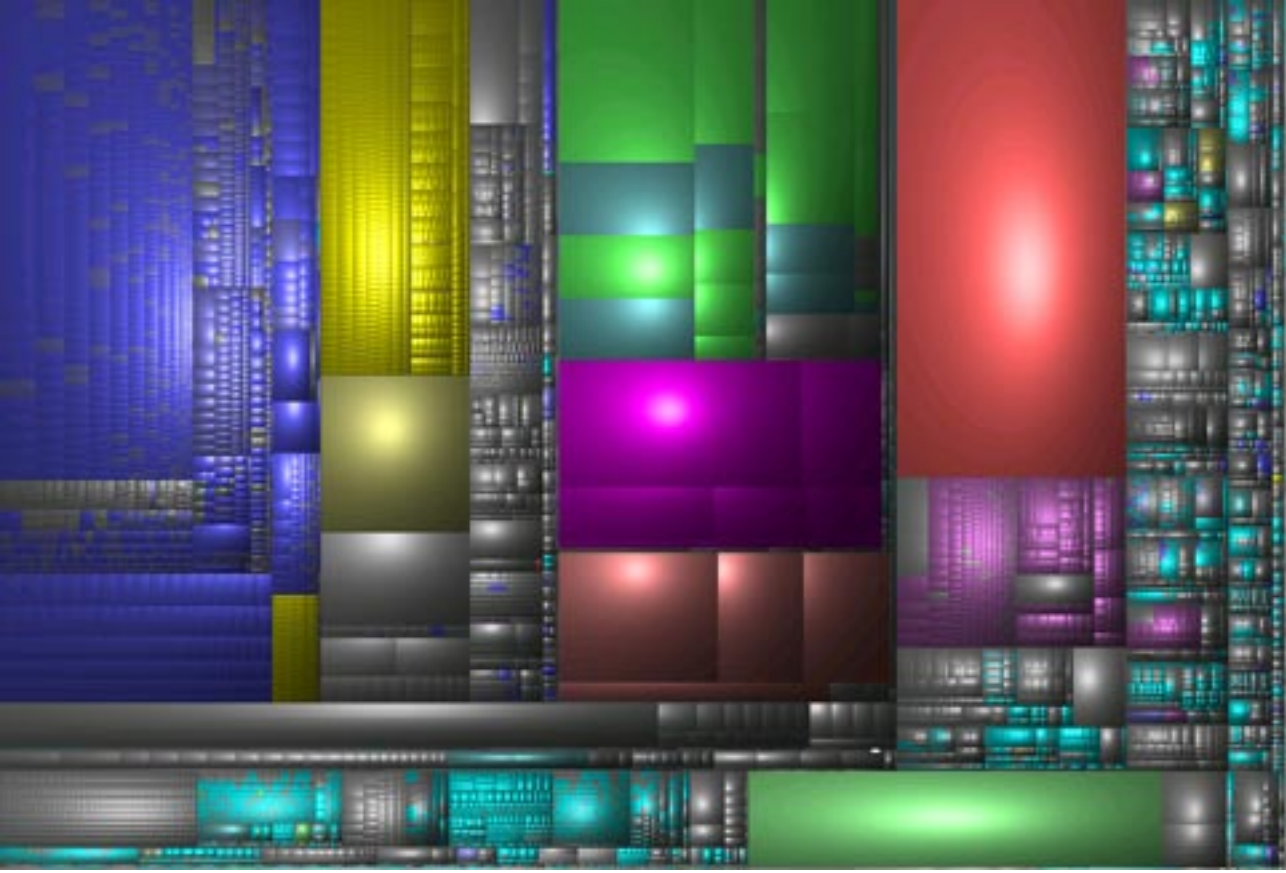
Windows tree.



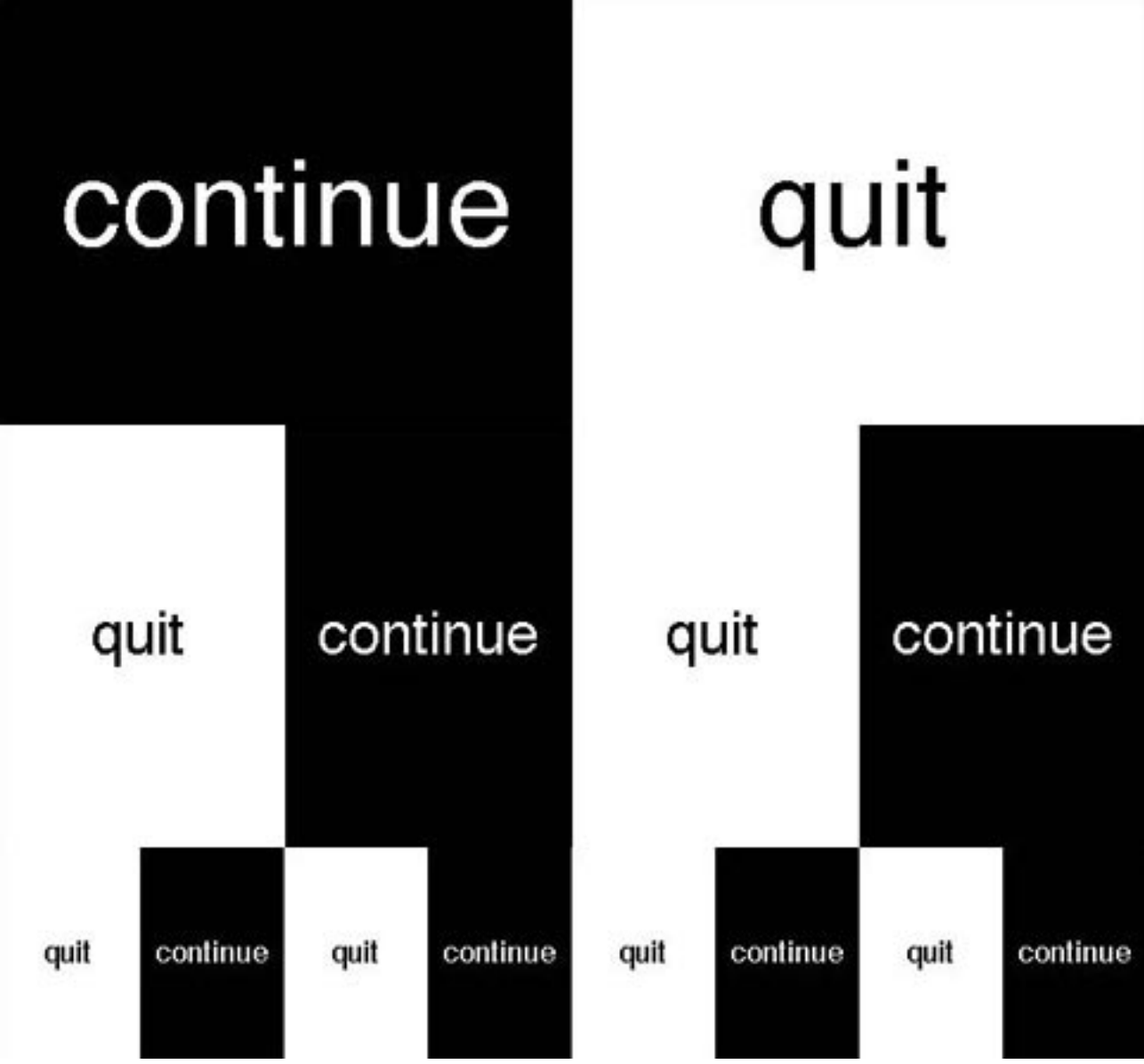
Navigation à base de figures récursives.
http://www.relevare.com/

Martin Wattenberg : Map of the market (2004/2006).
http://www.bewitched.com/
http://www.smartmoney.com/marketmap/

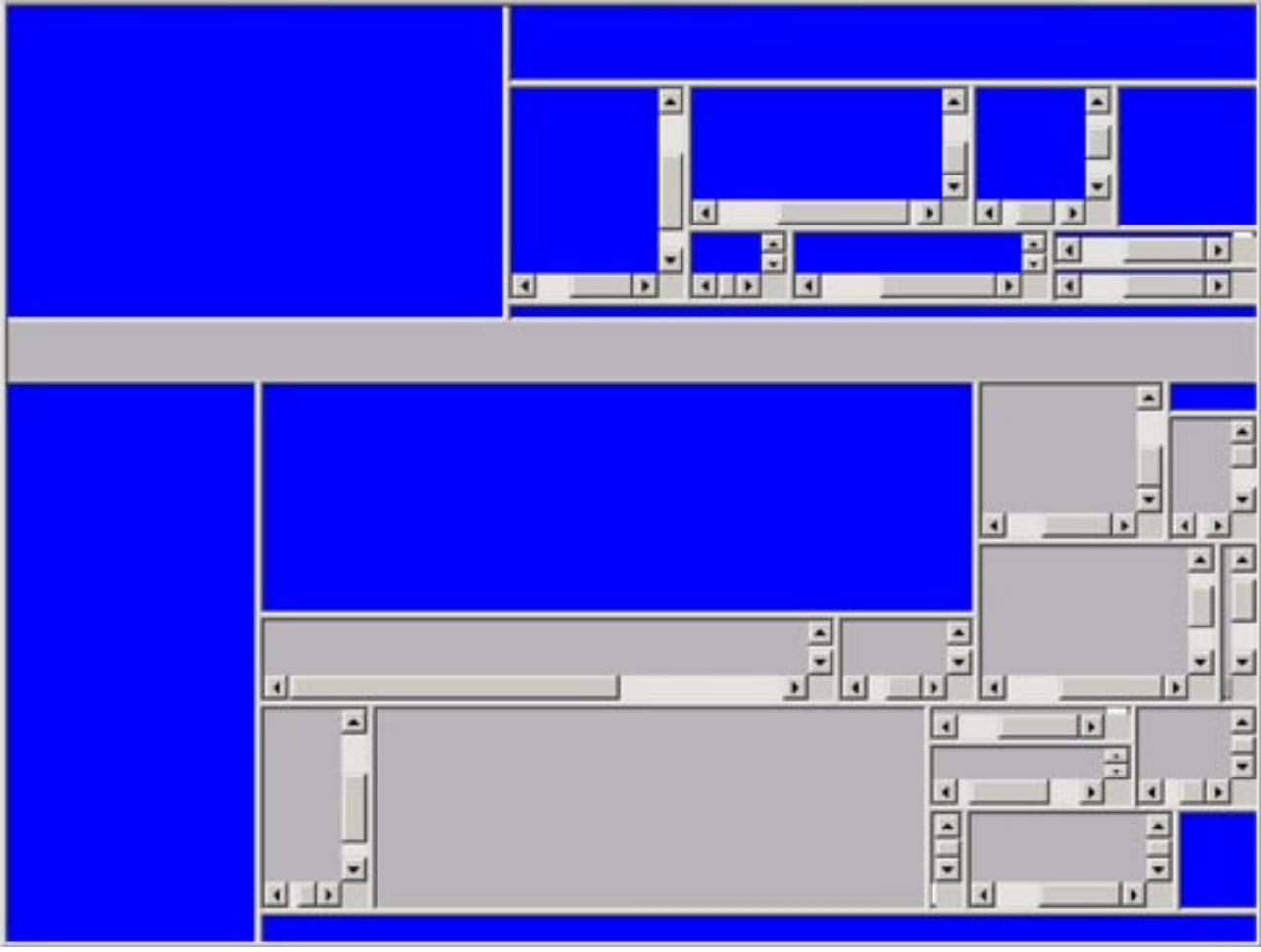
Marcos Weskamp : Newsmap (2004).
http://www.marumushi.com/apps/newsmap/



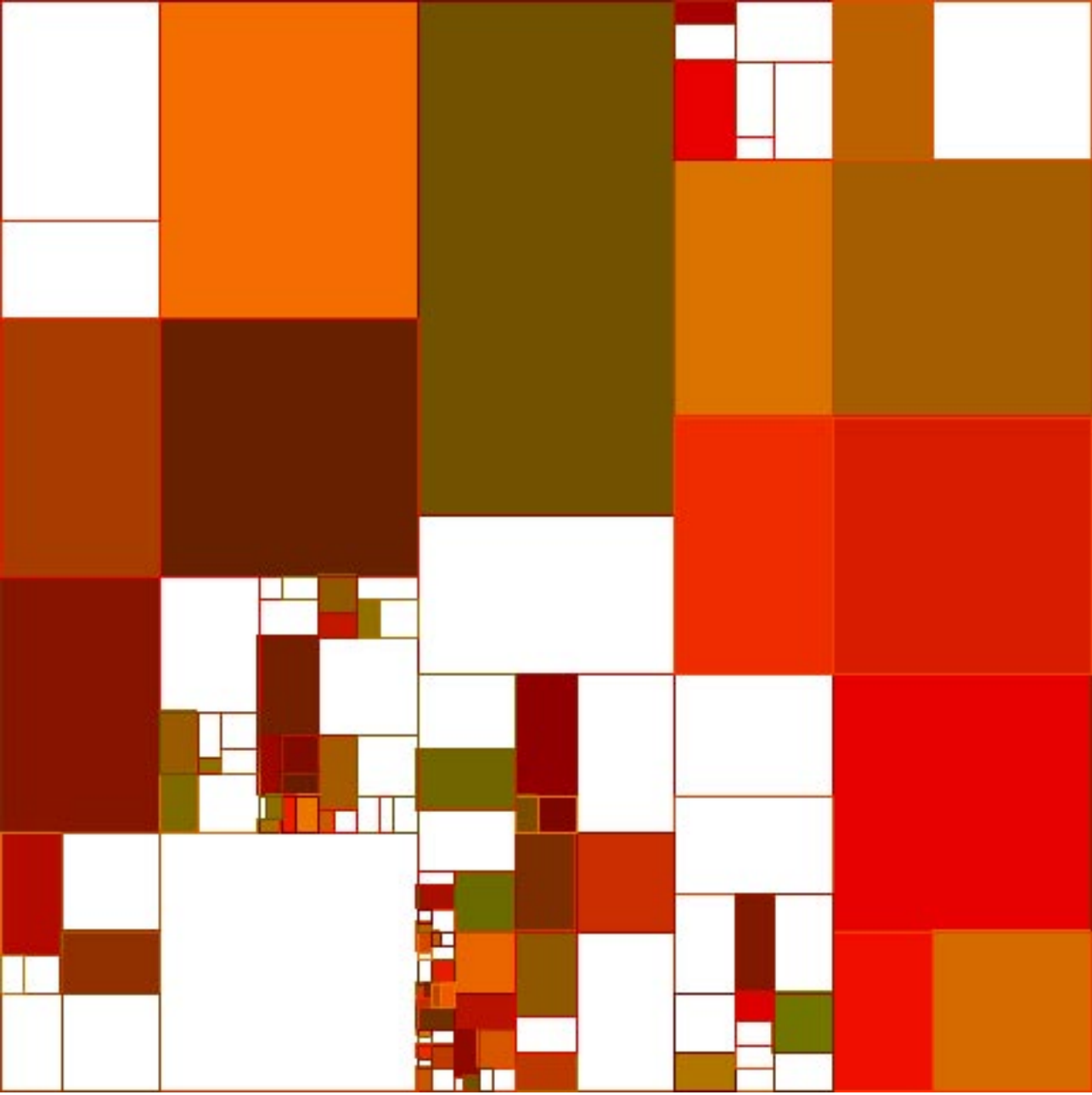
Faculty of Mathematics & Computer Science
(Technische Universiteit Eindhoven) : SequoiaView (2002).
http://www.win.tue.nl/sequoiaview/



Dieter Kiessling : Continue (1997),
CD-ROM «Artintact» publié par ZKM
(Zentrum für Kunst und Medientechnologie), Karlsruhe (Ger).
http://www.dieter-kiessling.de/cd.htm



Jan Robert Leegte : Scrollbar - Composition Blue (2000).
http://www.leegte.org/works/online/composition_blue/index.htm



William Ngan : Golden ratio.
http://www.metaphorical.net/code/processing/index2.html?page=golden_ratio

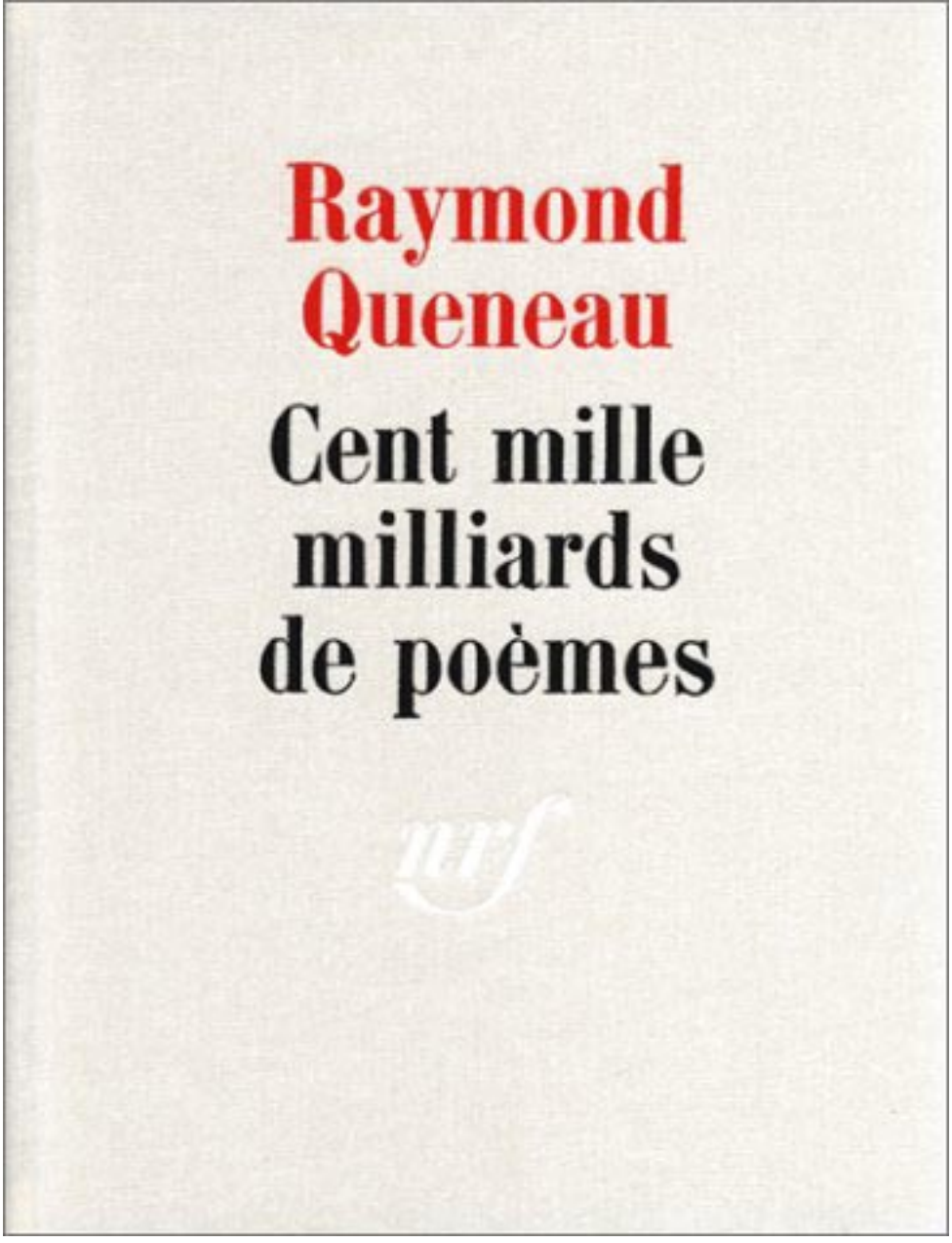
Exemple de récursivité : une fonction s’appelle elle-même.

```
// Recursion
// by CASEY REAS
// http://reas.com
// A demonstration of recursion, which means functions call themselves.
// Notice how the drawCircle() function calls itself at the end of its block.
// It continues to do this until the variable «level» is equal to 1.

// Updated 26 October 2004

void setup()
{
  size(200, 200);
  noStroke();
  smooth();
  noLoop();
}

void draw()
{
  drawCircle(126, 170, 6);
}

void drawCircle(int x, int radius, int level)
{
  float tt = 126 * level/4.0;
  fill(tt);
  ellipse(x, 100, radius*2, radius*2);
  if(level > 1) {
    level = level - 1;
    drawCircle(x - radius/2, radius/2, level);
    drawCircle(x + radius/2, radius/2, level);
  }
}
```

Aléatoire.

Dresser une procédure stricte telle qu'un algorithme n'empêche pas l'intrusion d'une fonction **aléatoire**.
La notion de **hasard** est formalisée par la fonction «**Random**».

http://www.ditl.info/arttest/art6037.php



Raymond Queneau : Cent mille milliards de poèmes (1961).

«Ce petit ouvrage permet à tout un chacun de composer à volonté cent mille milliards de sonnets, tous réguliers bien entendu. C'est somme toute une sorte de machine à fabriquer des poèmes, mais en nombre limité ; il est vrai que ce nombre, quoique limité, fournit de la lecture pour près de deux cents millions d'années (en lisant vingt-quatre heures sur vingt-quatre)».

Le livre est composé de dix feuilles, chacune séparée en quatorze bandes horizontales, chaque bande portant sur son recto un vers. Le lecteur peut donc, en tournant les bandes horizontales comme des pages, choisir pour chaque vers une des dix versions proposées par Queneau. Les dix versions de chaque vers ont la même scansion et la même rime, ce qui assure que chaque sonnet ainsi assemblé est régulier dans sa forme.

Il y a donc 10^{14} soit 100 000 000 000 000 poèmes potentiels.

Queneau ajoute: «En comptant 45s pour lire un sonnet et 15s pour changer les volets à 8 heures par jour, 200 jours par an, on a pour plus d'un million de siècles de lecture, et en lisant toute la journée 365 jours par an, pour 190 258 751 années plus quelques plombes et broquilles (sans tenir compte des années bissextiles et autres détails)».

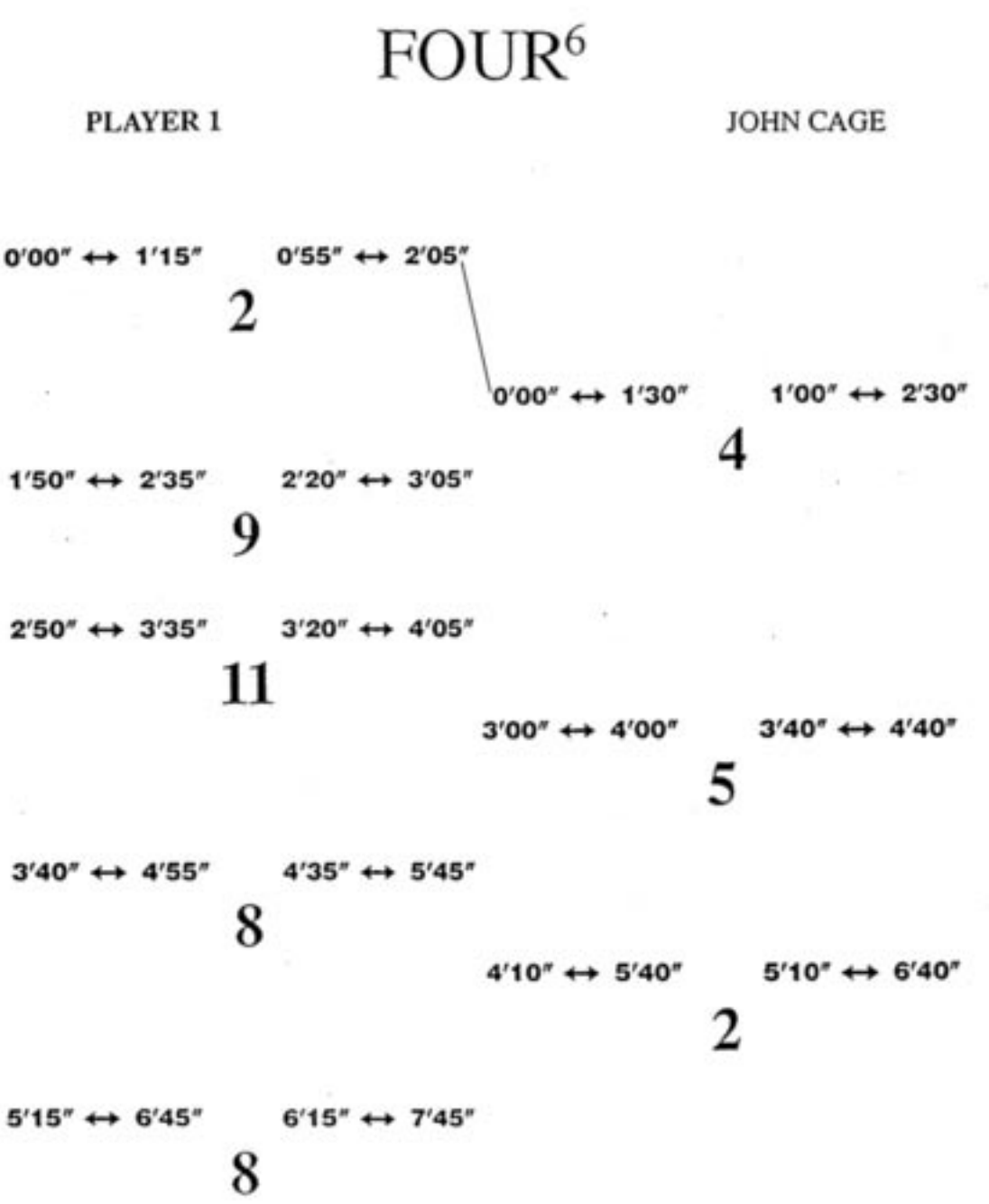
Le roi de la pampa retourne sa chemise
pour la mettre à sécher aux cornes des taureaux
le cornédbif en boite empeste la remise
et fermentent de même et les cuirs et les peaux

Je me souviens encor de cette heure exeuquise
les gauchos dans la plaine agitaient leurs drapeaux
nous avions aussi froids que nus sur la banquise
lorsque pour nous distraire y plantions nos tréteaux

Du pôle à Rosario fait une belle trotte
aventures on eut qui s'y pique s'y frotte
lorsqu'on boit du maté l'on devient argentin

L'Amérique du Sud séduit les équivoques
exaltent l'espagnol les oreilles baroques
si la cloche se tait et son terlintintin

Voir une modélisation de ce livre par Manny Tan :
http://www.uncontrol.com/_massin/massin_small.html



John Cage, extrait de «Four 6», instrumentation libre (1989).

John Cage a inventé le «piano préparé», piano dont le son est altéré en plaçant divers objets — la préparation — dans ses cordes.

«Le piano préparé est en réalité un ensemble de percussion confié aux mains d'un seul interprète» — John Cage.

Les positions de John Cage s'opposent au caractère rationnel de la construction musicale européenne. Pour Cage, nourri de pensée orientale, tout son est musique, et il est insensé de l'organiser selon des structures précises dans des œuvres qui seraient des produits finis.

http://www.olats.org/pionniers/pp/cage/cage.php

James Tenney préparant un piano.

DADAPHONE

écrire à :
TRISTAN TZARA
32, Avenue Charles Floquet
Administration : AU SANS PAREIL, 37, Avenue Kléber

N° 7

PRIX :
1 FR. 50

PARIS
MARS 1920

POUR FAIRE UN POÈME DADAÏSTE.

Prenez un journal.

Prenez des ciseaux.

Choisissez dans le journal un article ayant la longeur
que vous comptez donner à votre poème.

Découpez l'article.

Découpez ensuite avec soin chacun des mots qui forment cet article
et mettez-les dans un sac.

Agitez doucement.

Sortez ensuite chaque découpe l'une après l'autre.

Copiez consciencieusement dans l'ordre où elles ont quitté le sac.
Le poème vous ressemblera.

Et vous voilà un écrivain infiniment original et d'une sensibilité charmante,
quoiqu'incomprise du vulgaire.

(Tristan Tzara)

Exemple : une variable random.

```
// définir les conditions d'affichage
void setup() {
  size(400,400);
  background(255);
  smooth(); // pas de crenelage
  framerate(10); // images par sec.
  // noLoop(); // Cette commande arreterait void draw()
}

// dessiner une ligne
void draw() {
  line(200,200, random(0,400), random(0,400) );
}
```

Programmation.

Langage.

Un **langage de programmation** est un dialecte dans lequel on peut exprimer des programmes.

Ce langage fournit une **syntaxe**, c'est-à-dire une façon de représenter les ordres donnés à l'ordinateur qui sera plus facilement manipulable par un être humain.

Ce langage évolué (dit de «haut niveau») sera ensuite traduit en **langage machine** (dit de «bas niveau»), langue maternelle de l'ordinateur composée exclusivement de 0 et de 1.

Le programme, dans sa forme compréhensible par un humain, est appelé «**code source**».

Pour en savoir plus :
http://www.commentcamarche.net/langages/langages.php3

Voici une même quantité exprimée sous trois formes différentes :

13

s'exprime avec deux symboles, chacun étant choisi parmi les chiffres 0 à 9. Nous disons que nous avons utilisé deux «positions» d'un code à dix «moments», ces dix moments étant les dix chiffres 0, 1, 2, 3, 4, 5, 6, 7, 8 et 9).

Treize

s'exprime avec six symboles (lettres), chacun étant choisi parmi les vingt-six lettres de l'alphabet. Nous disons que nous avons utilisé six positions d'un code à vingt-six moments.

XIII

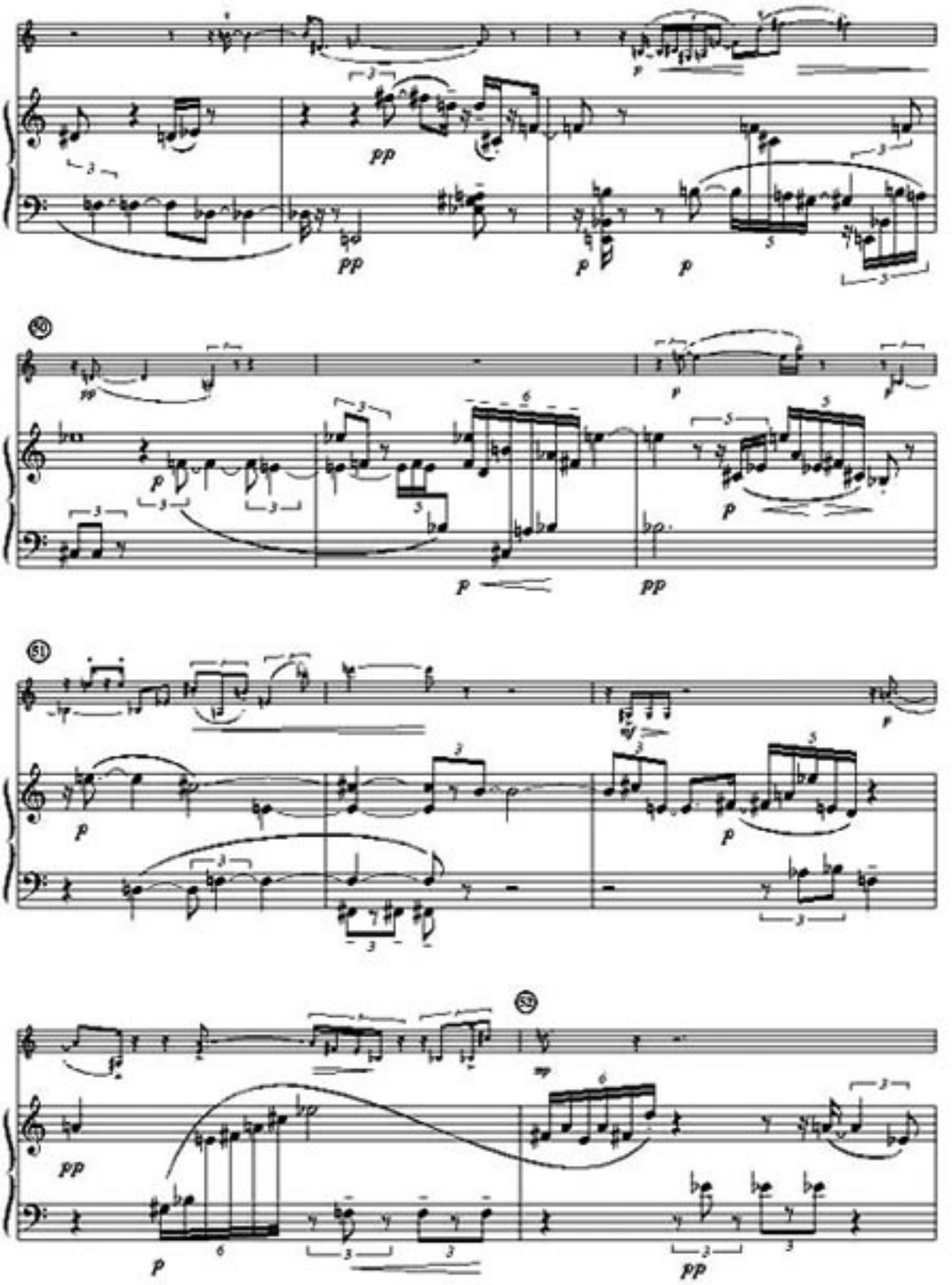
s'exprime avec 4 positions d'un code à sept moments: les chiffres romains (I, V, X, L, C, M, D).

Quant aux codes binaires employés par l'ordinateur ce sont tout simplement des codes à deux moments. Il suffit donc de deux symboles pour exprimer une information binaire.

On emploie tout simplement les deux premiers chiffres de notre système décimal, c'est-à-dire 0 et 1.

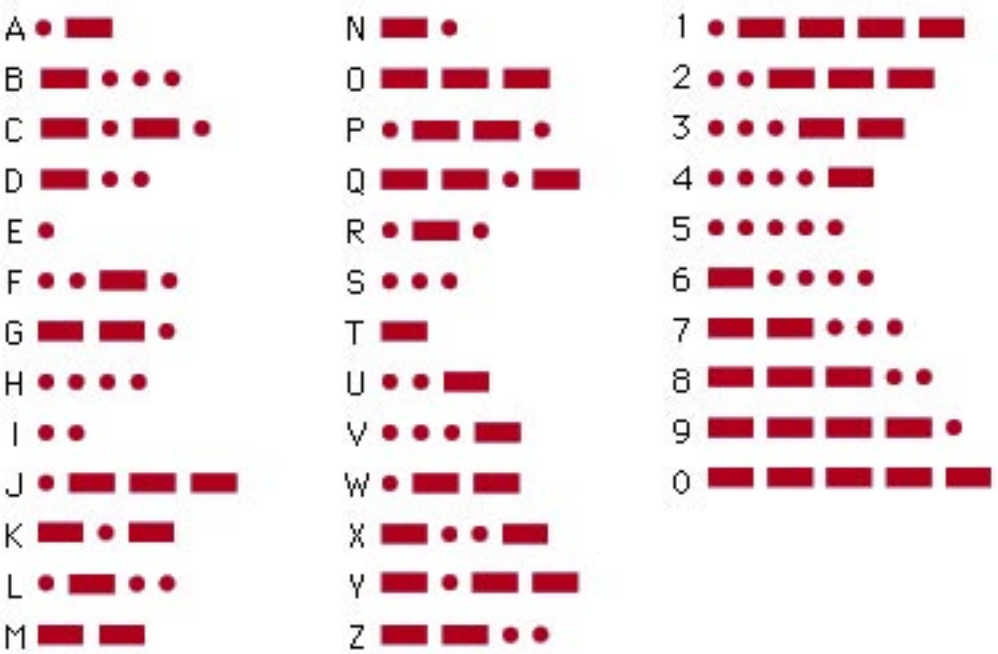
Ainsi : 100110101 représente une information binaire utilisant huit positions. Chaque position porte le nom de «bit». Le terme bit est donc l'équivalent pour les codes binaires, des termes chiffres ou lettres employés par les codes rencontrés précédemment.

La Pierre de Rosette, découverte en 1799,
décryptée par Jean-François Champollion après huit ans de travail.
http://fr.wikipedia.org/wiki/Pierre_de_Rosette

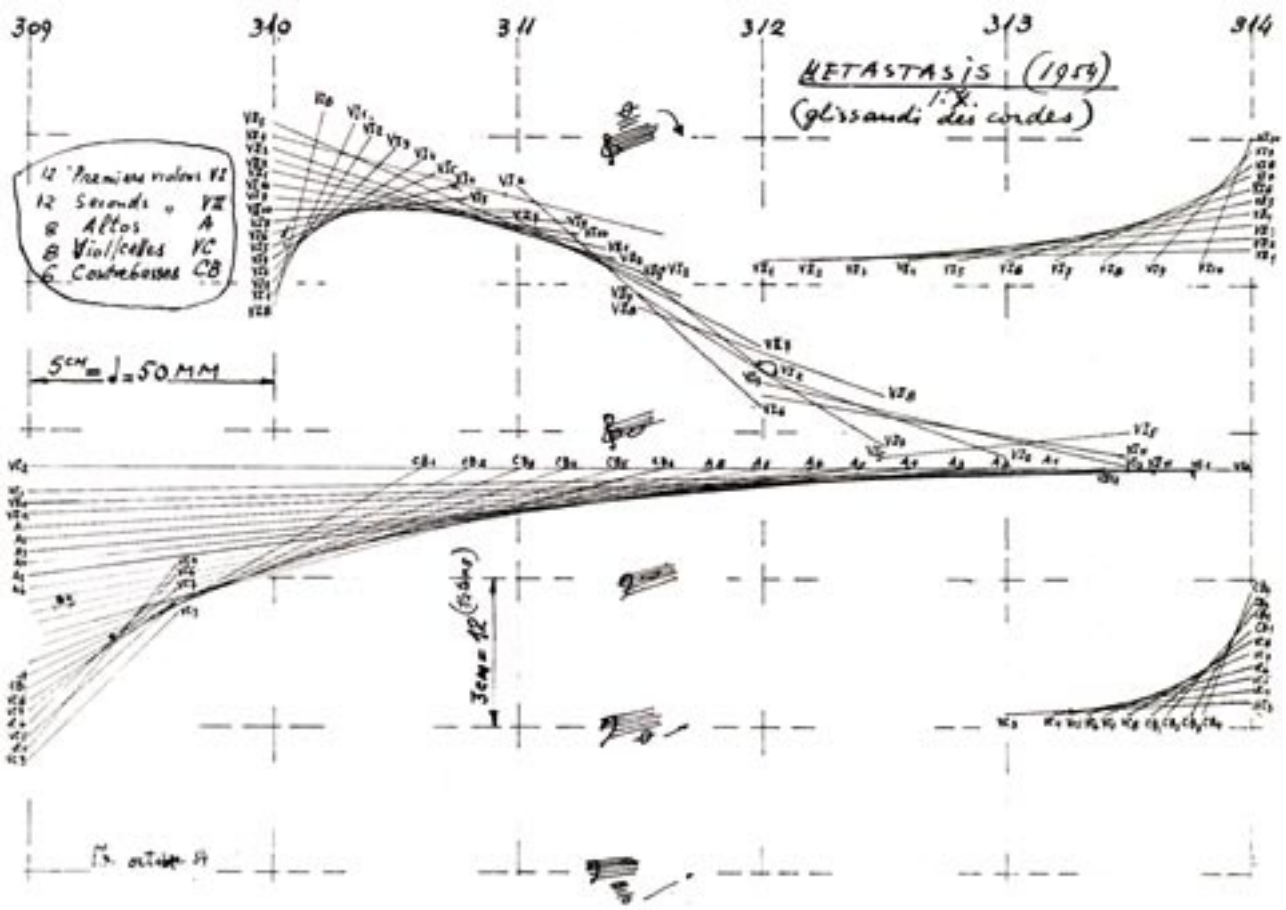

50
51
52

THE SEMAPHORE ALPHABET.

CHARACTERS	HAND FLAGS	CHARACTERS	HAND FLAGS	CHARACTERS	HAND FLAGS	CHARACTERS	HAND FLAGS
A		H		O		V	
B		I		P		W	
C ANSWERING SIGN		J		Q		X	
D		K		R		Y	
E		L		S		Z	
F		M		T		ATTEN-TION	
G		N		U		BREAK	

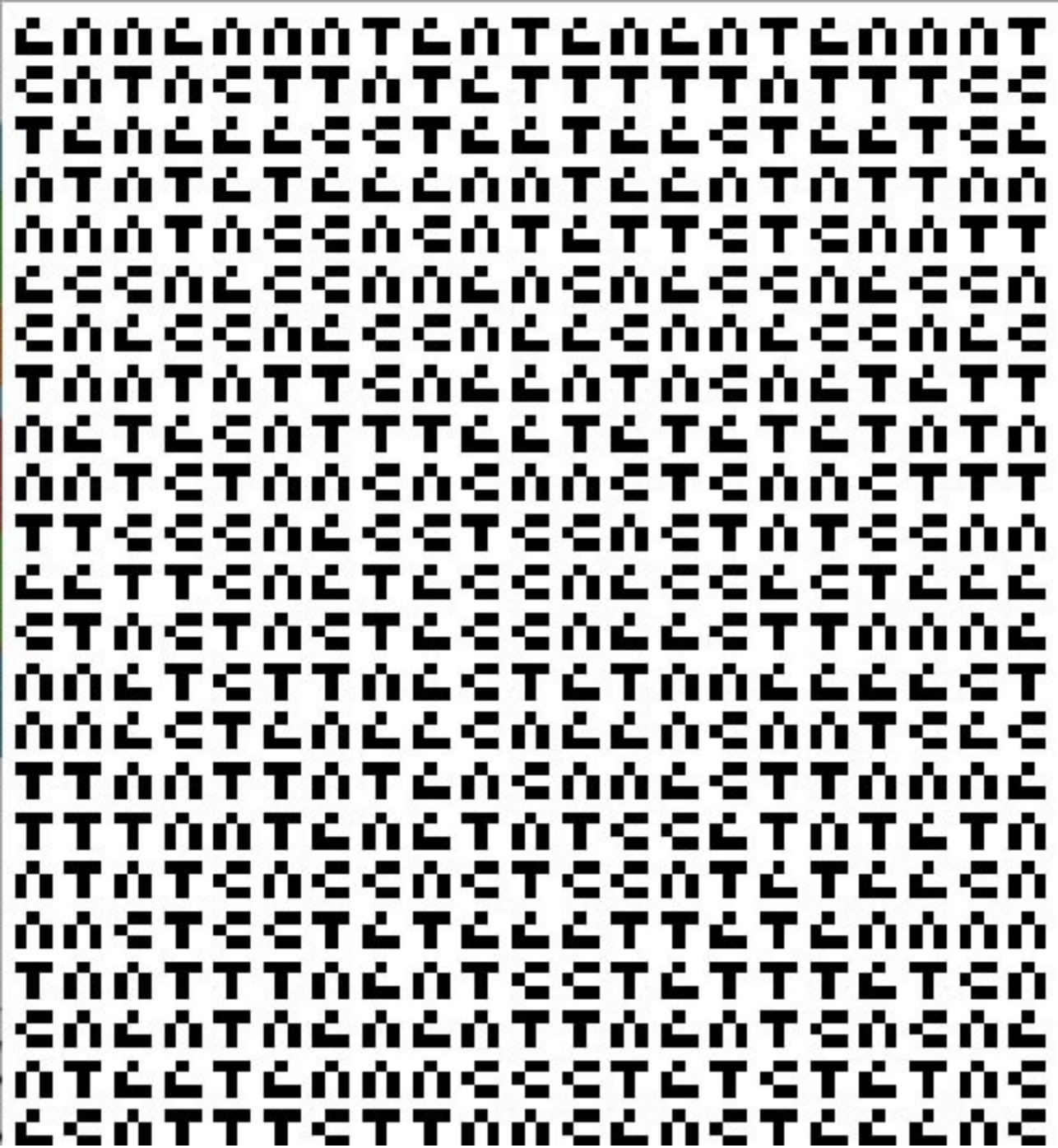


Le code Morse, conçu en 1838 par Samuel Morse.
Il est intéressant de noter que Samuel Morse était un peintre.
http://en.wikipedia.org/wiki/Samuel_Morse

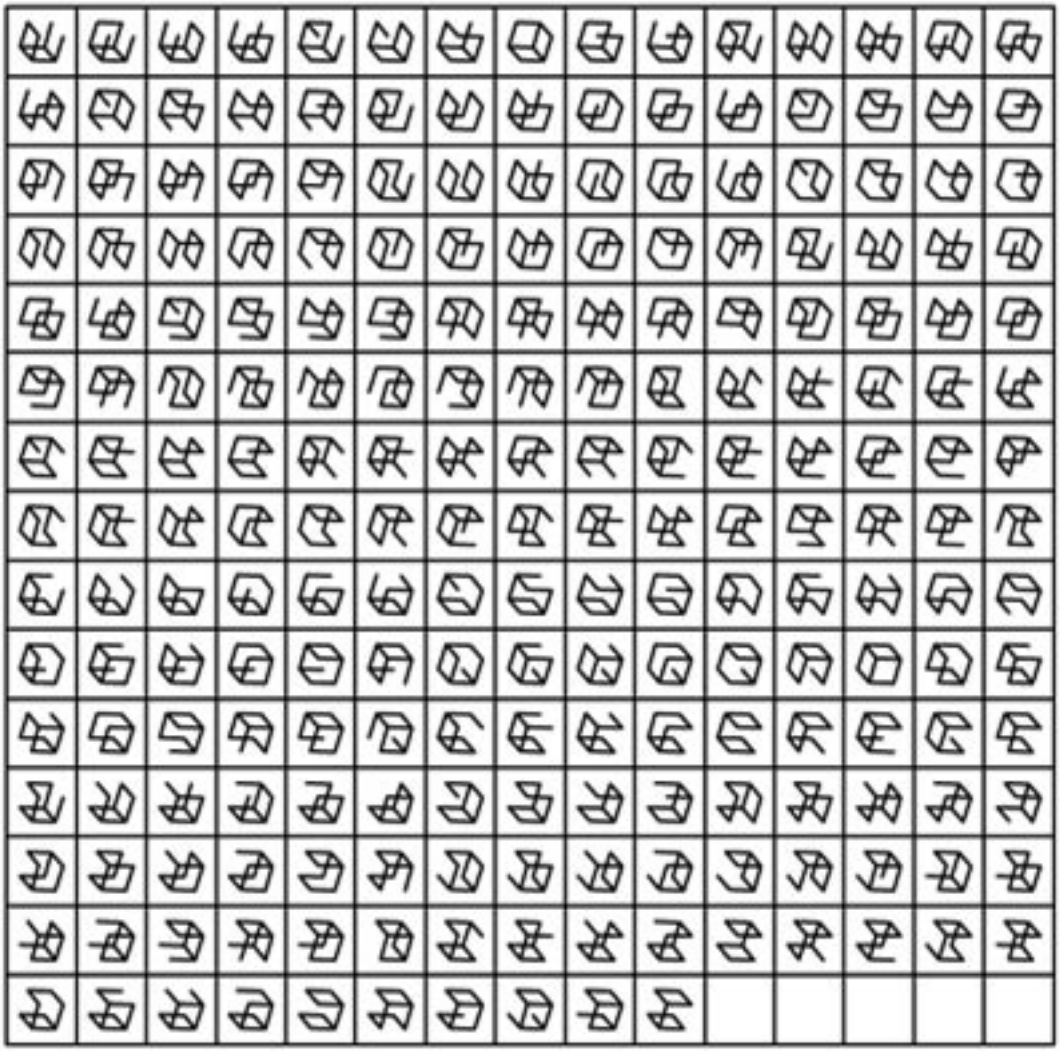


Iannis Xenakis : Metastasis (1954).
http://www.iannis-xenakis.org/

http://fr.wikipedia.org/wiki/Braille

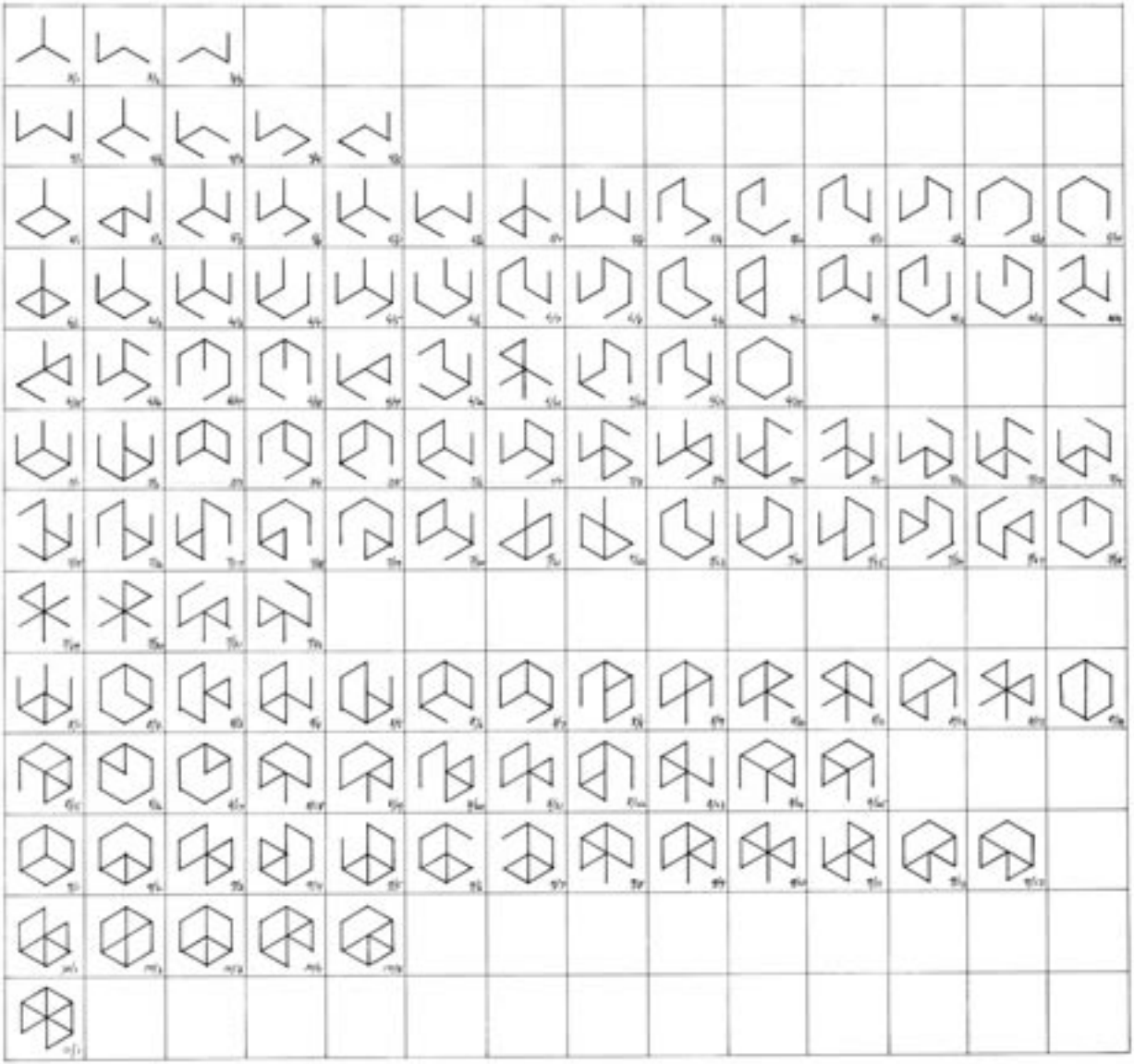


Benjamin Fry : Chromosome 22.
http://acg.media.mit.edu/people/fry/chromosomes/22/



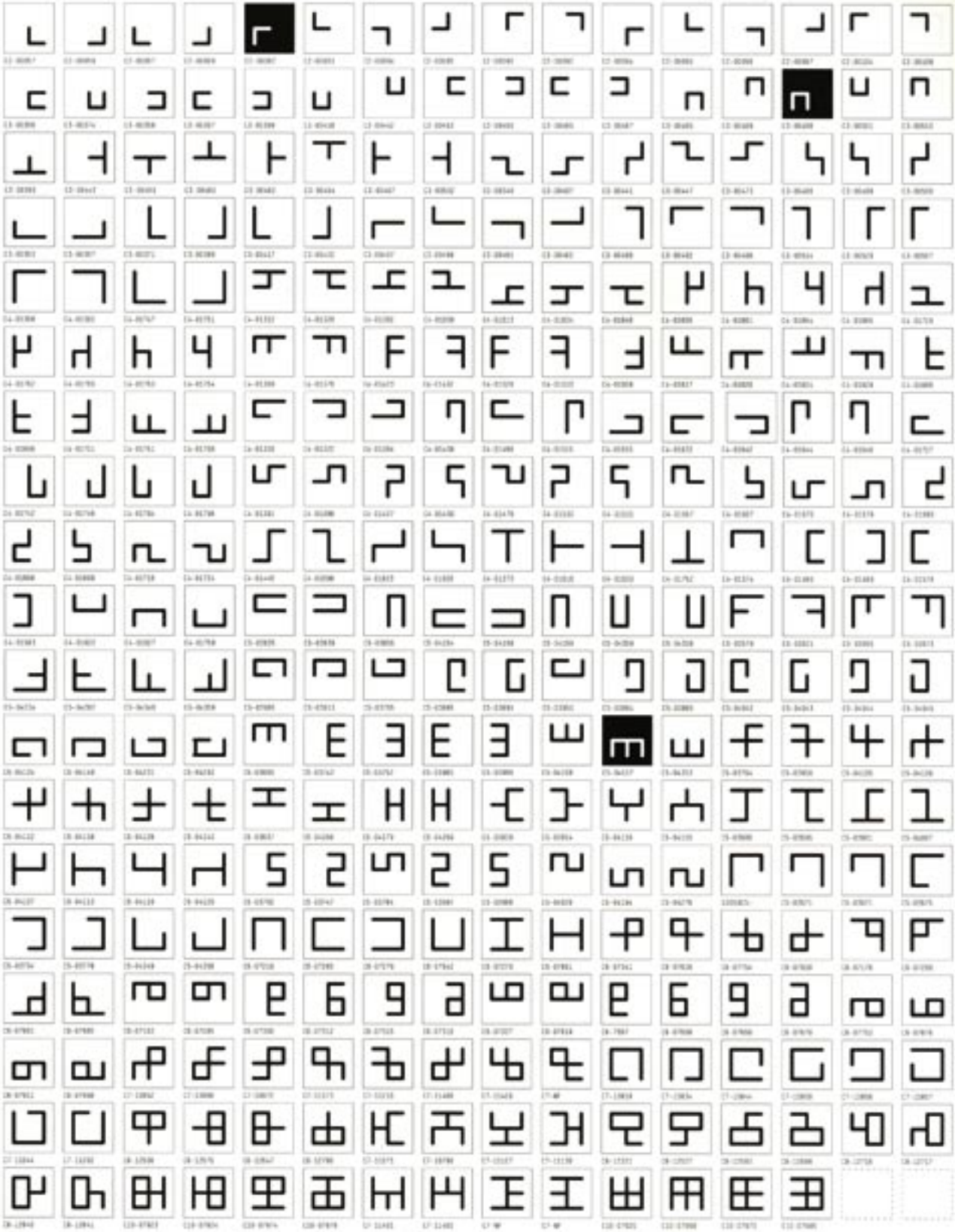
Combinatorial possibilities of missing lines in a cube (at a given rotation),
«Cubic Limit» serie by manfred mohr (1972-76).
http://www.emohr.com/

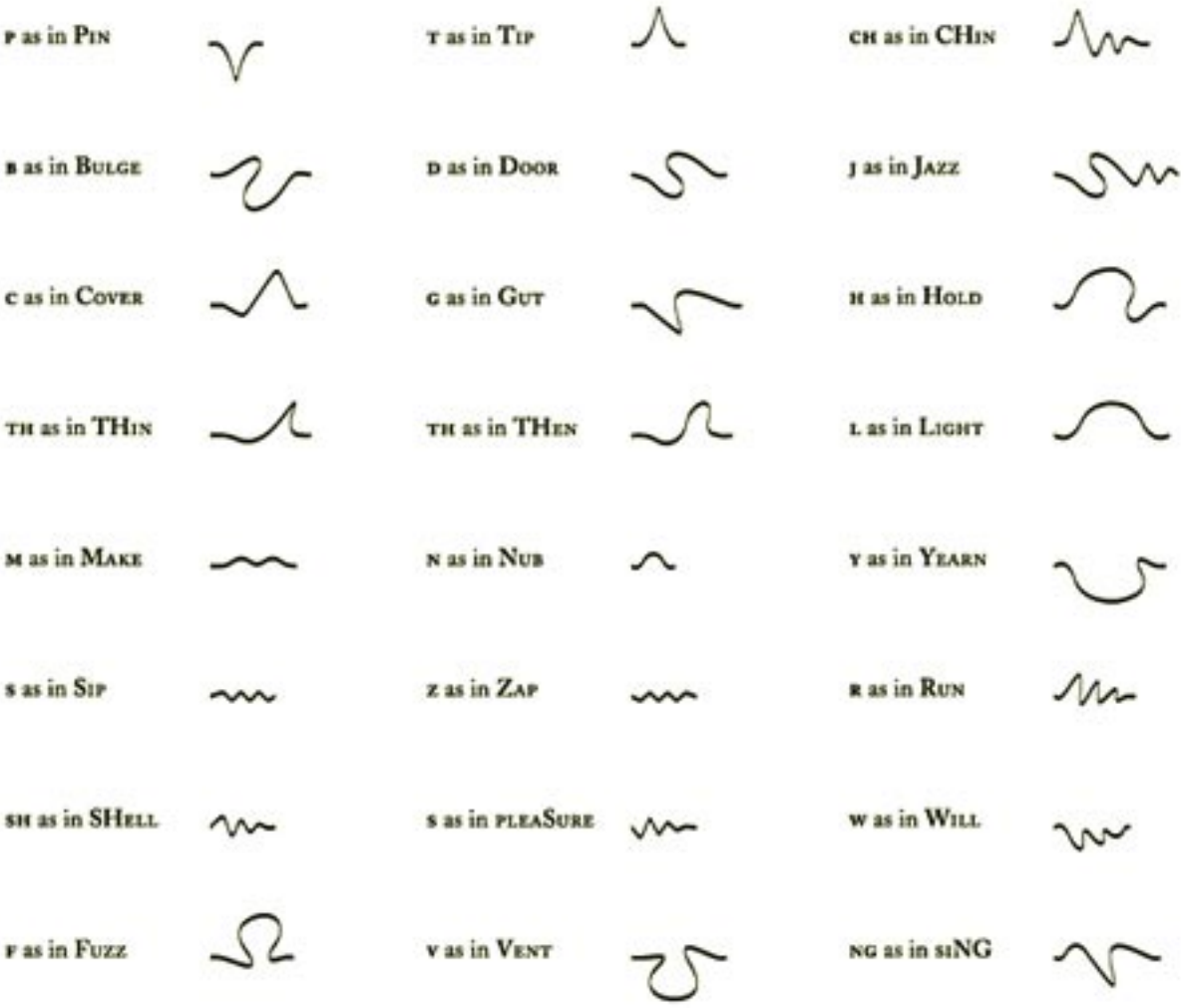
Combinatorial possibilities of missing lines in a cube (at a given rotation),
«Cubic Limit» serie by manfred mohr (1972-76).
http://www.emohr.com/



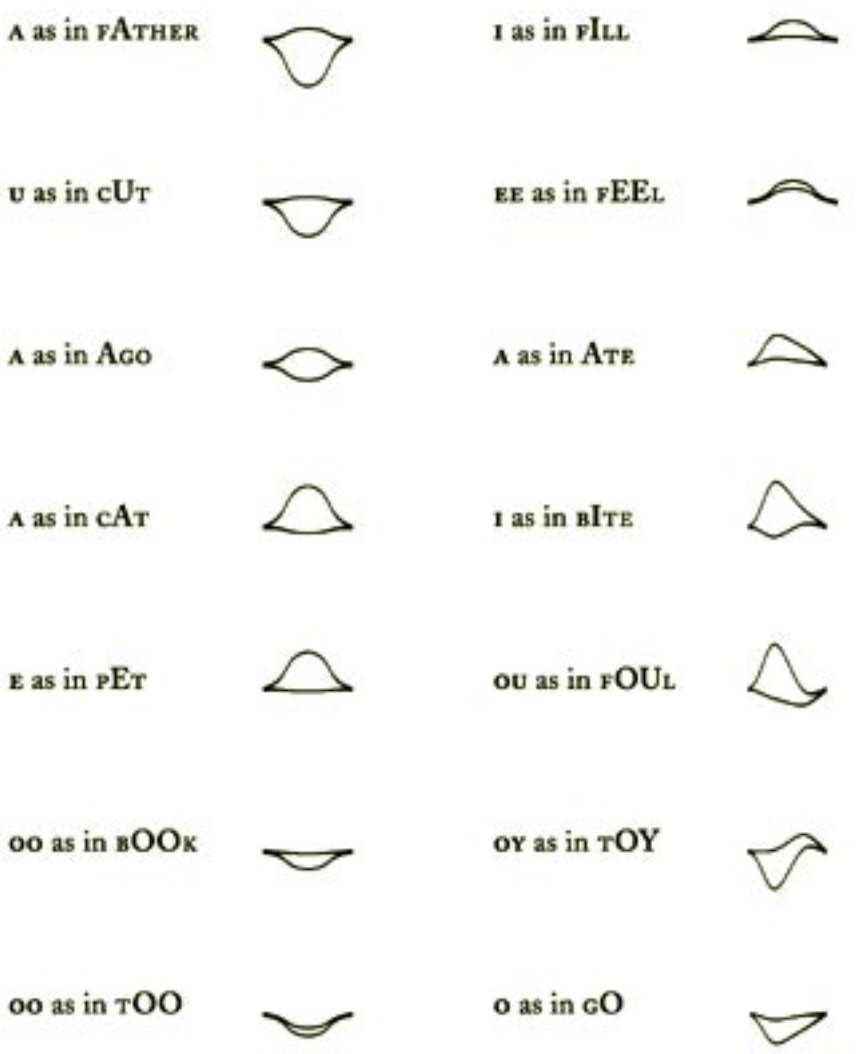
Sol LeWitt : Variations of Incomplete Open Cubes (1974).
http://www.barbarakrakowgallery.com/

Page suivante :
Norm (Dimitri Bruni & Manuel Krebs) & Jürg Lehni : signs.
http://www.norm.to/pages/generator_3.html

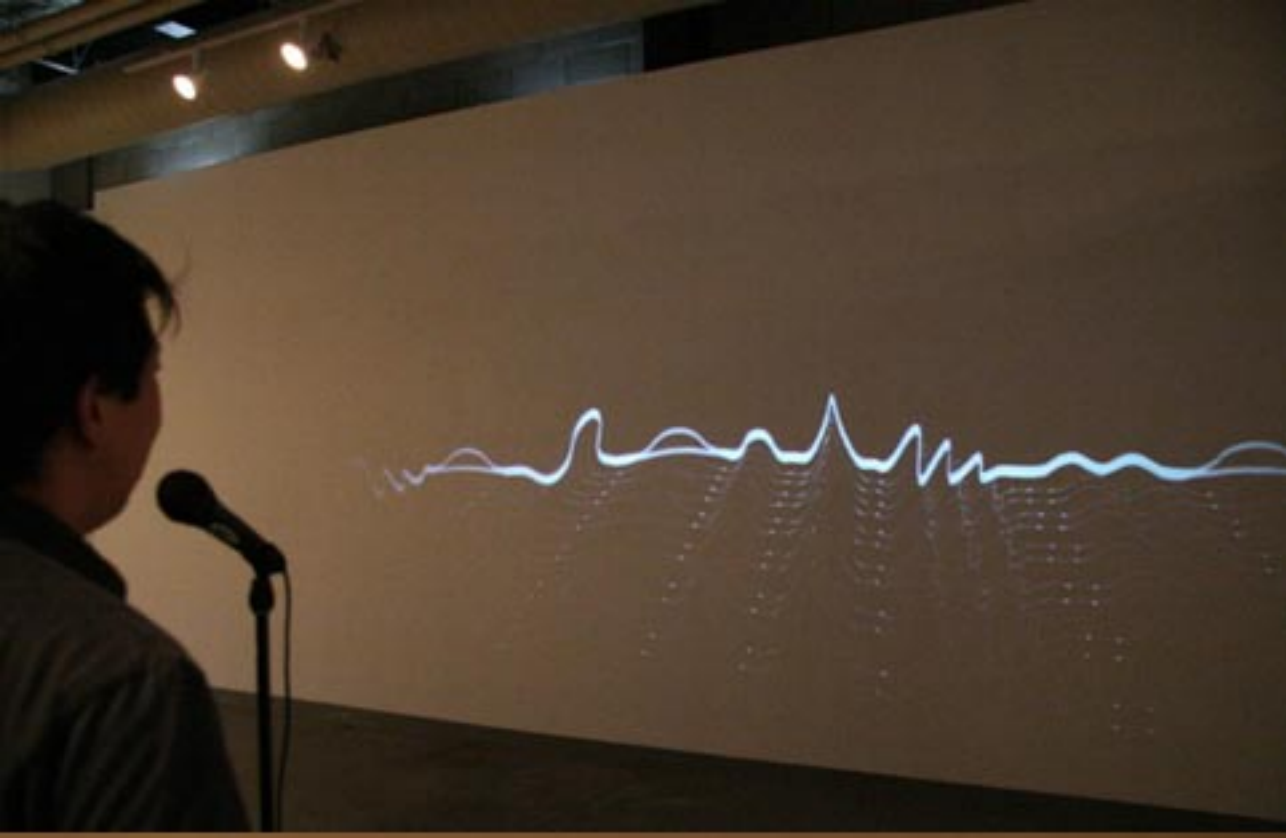


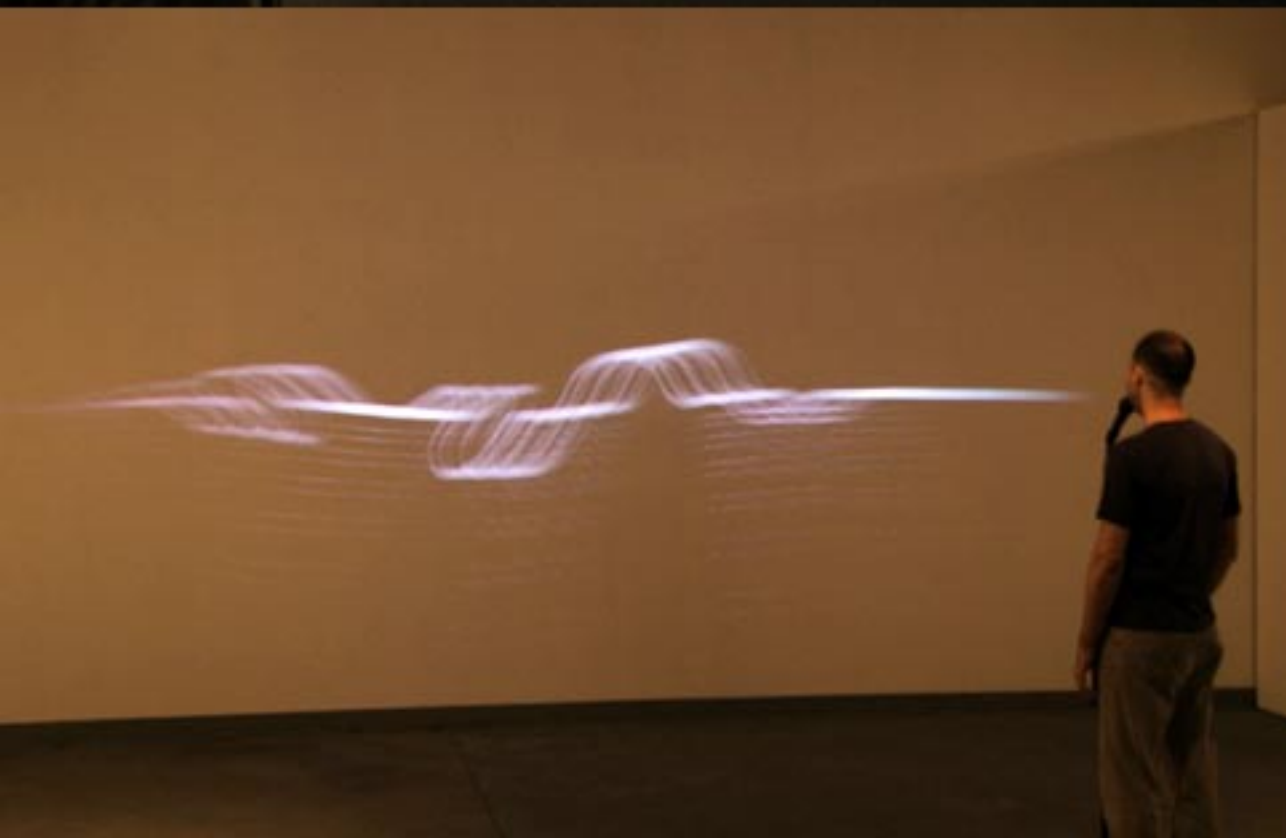


Peter Cho : Takeluma (2005),
Writing system for representing speech sounds (reactive installation).
http://www.pcho.net/takeluma



Peter Cho : Takeluma (2005),
Writing system for representing speech sounds (reactive installation).
http://www.pcho.net/takeluma





Interprétation du langage.

Avec l'aide d'un **interprète**, nous communiquons avec une chinoise sans connaître sa langue :

Bonjour Mademoiselle Li !

早，李太太，您早！

De la même manière, nous pourrons communiquer avec l'ordinateur sans connaître son langage. Nous allons employer l'ordinateur lui-même (ou plus exactement un programme appelé «**interprète**» ou «**compilateur**» selon le cas) pour effectuer la traduction du langage que vous utilisez dans celui de l'ordinateur :

Bonjour eMAC !

0101011000100010001
0100110011101110110
0100111011111011101...

...

NB : «communiquer avec l'ordinateur» est une notion toute relative : l'eMAC ne comprend rien à ce que je lui raconte. Il se borne à exécuter. Lui dire bonjour est donc inutile... Le test habituel est «Hello world !», formule consacrée qui démontre que nous allons communiquer *à partir de notre application* avec le monde, mais aussi que nous traitons notre programme comme le début d'une *entité autonome qui parlerait en son nom propre.*

Bonjour eMAC !

```
public class Hello {
      public static void main(String[] args)
      {
          System.out.println(«Bonjour eMac !»);
      }
}
```

```
0101011000100010001
0100110011101110110
0100111011111011101...
```

...

Bonjour eMAC !

Français :

Processing :

```
size(200, 200);
background(255);
PFont fontA = loadFont(«HelveticaNeue-
48.vlw»);
textFont(fontA, 14);
int x = 50;
fill(0);
text(«Bonjour eMac !», x, 100);
```

Java :

```
public class Hello {
      public static void main(String[] args)
      {
          System.out.println(«Bonjour eMac !»);
      }
}
```

Assembleur :

```
0101011000100010001
0100110011101110110
0100111011111011101...
```

Langage-machine :

```
0101011000100010001
0100110011101110110
0100111011111011101...
```

...

PROCESSING est un environnement qui simplifie la création de programmes dans un langage plus extensible, plus puissant, mais plus complexe : Java.

Si vous voulez savoir à quoi ressemblerait votre programme si vous l'aviez programmé directemment dans Java, ouvrez le fichier .java qui se trouvera à l'intérieur du dossier "applet" généré lorsque vous exporterez votre création dans PROCESSING afin de la publier en ligne (fonction "export", qui produit un ensemble de fichiers dont une page HTML). C'est une bonne manière de comprendre ce qui se passe.

Processing fonctionne grâce à Java et ses outils.

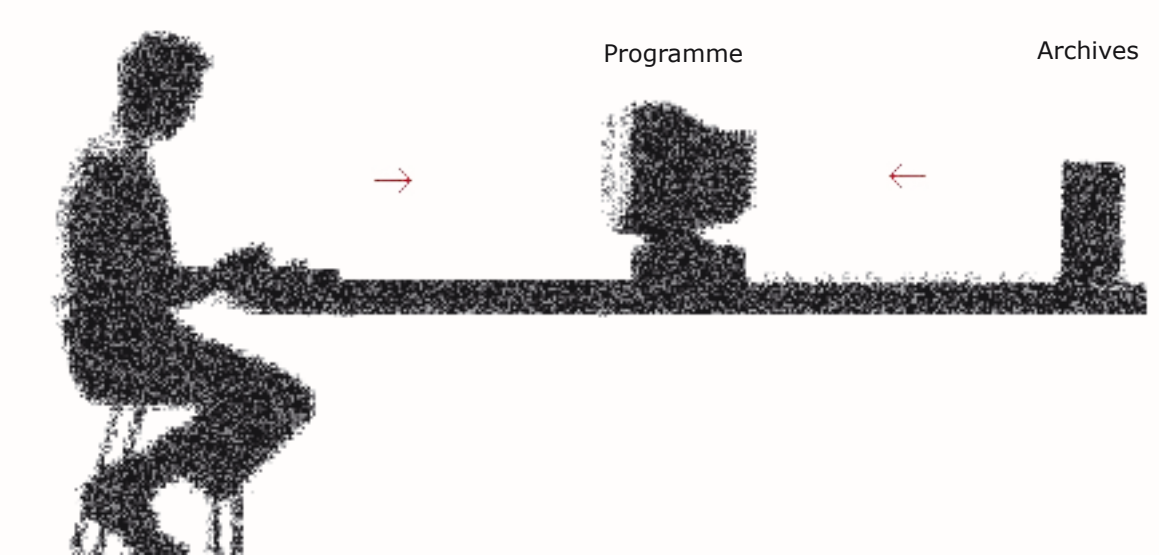
Concernant le schéma de la page précédente, j'ai volontairement simplifié les étapes qui mènent de **PROCESSING** au langage-machine. Java est un langage **compilé** : il est traduit en langage machine à l'aide d'un **compilateur**. Toutefois, contrairement aux langages compilés traditionnels, pour lesquels le compilateur crée un fichier binaire directement exécutable par un processeur donné (c'est-à-dire un fichier binaire contenant des instructions spécifiques à un processeur), le code source Java est compilé en un langage intermédiaire (appelé **pseudo-code** ou **bytecode**) dans un fichier portant le même nom que le fichier source à l'exception de son extension (.class).

Pour en savoir plus :

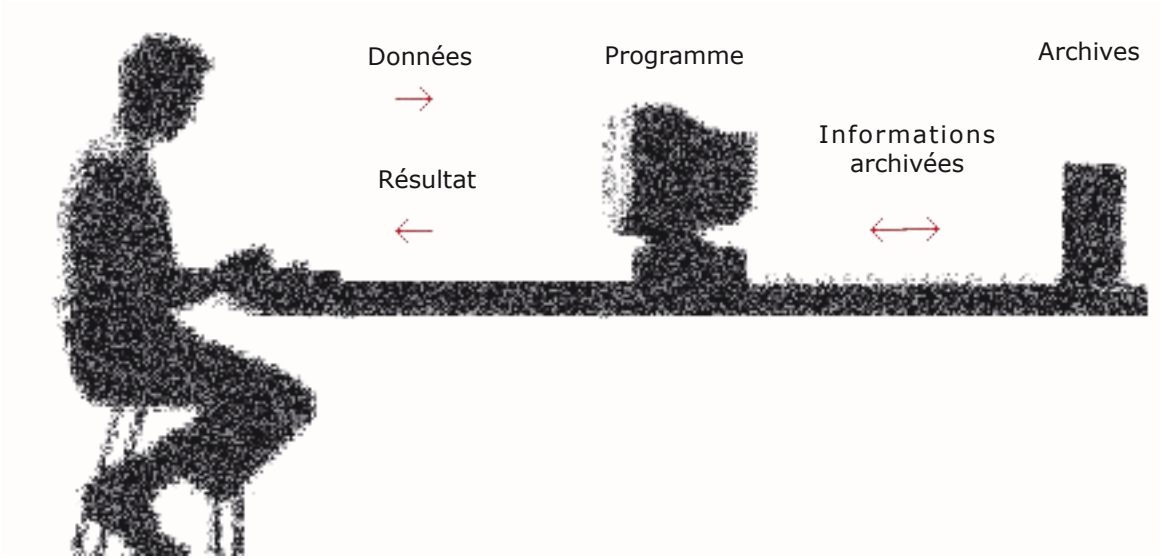
http://www.commentcamarche.net/java/javacarac.php3

Entrée/traitement/résultat.

La transmission du programme à l'ordinateur : programme fourni directement (via le clavier) ou recherché en archives (sur le disque dur).



L'exécution du programme et le résultat : le programme utilise des données et fournit des résultats. Là encore, les données peuvent être fournies par l'utilisateur (clavier, souris, interface, capteur) ou par des archives scrutées par le programme lui-même (base de données).

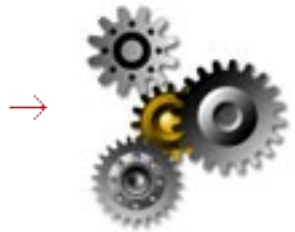


Un programme est donc un outil qui traite l'information selon ce schéma :

Entrée :

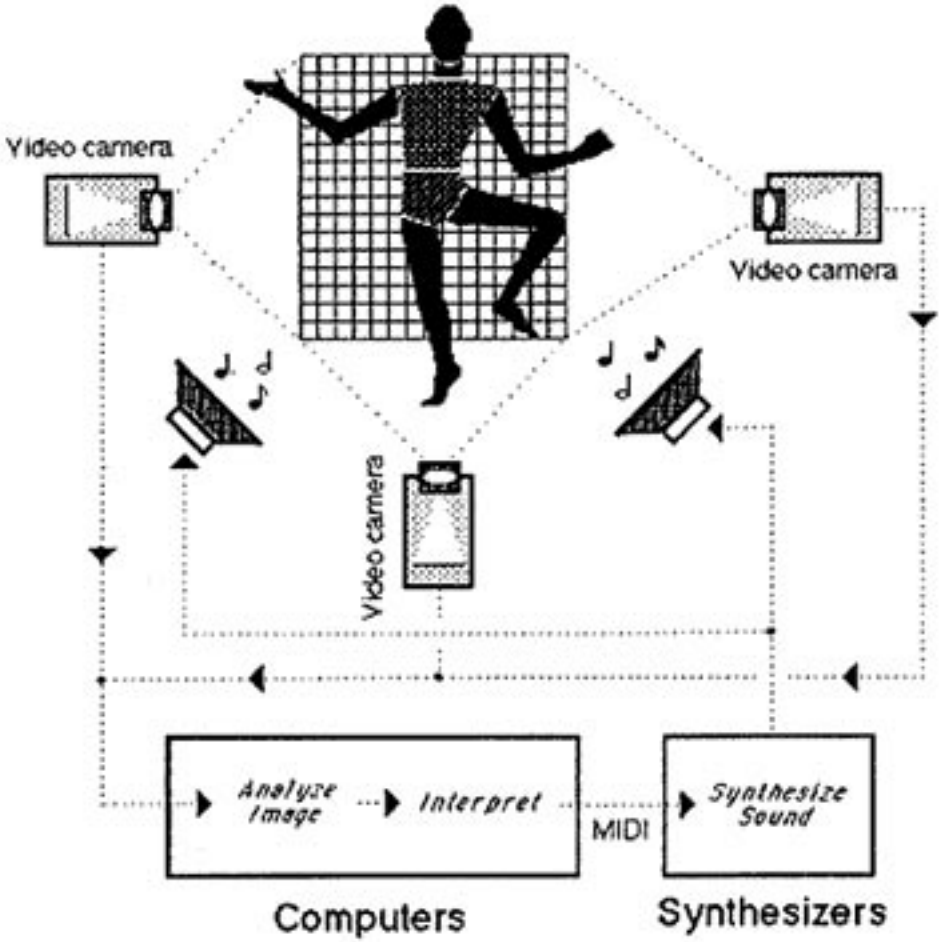
L'utilisateur **transmet** l'information à l'ordinateur par l'intermédiaire du clavier ou de la souris (ou bien les données sont saisies par le programme sur le disque dur). Cette transmission de données à l'ordinateur est aussi appelée **input** ou encore **saisie**.

Traitement :

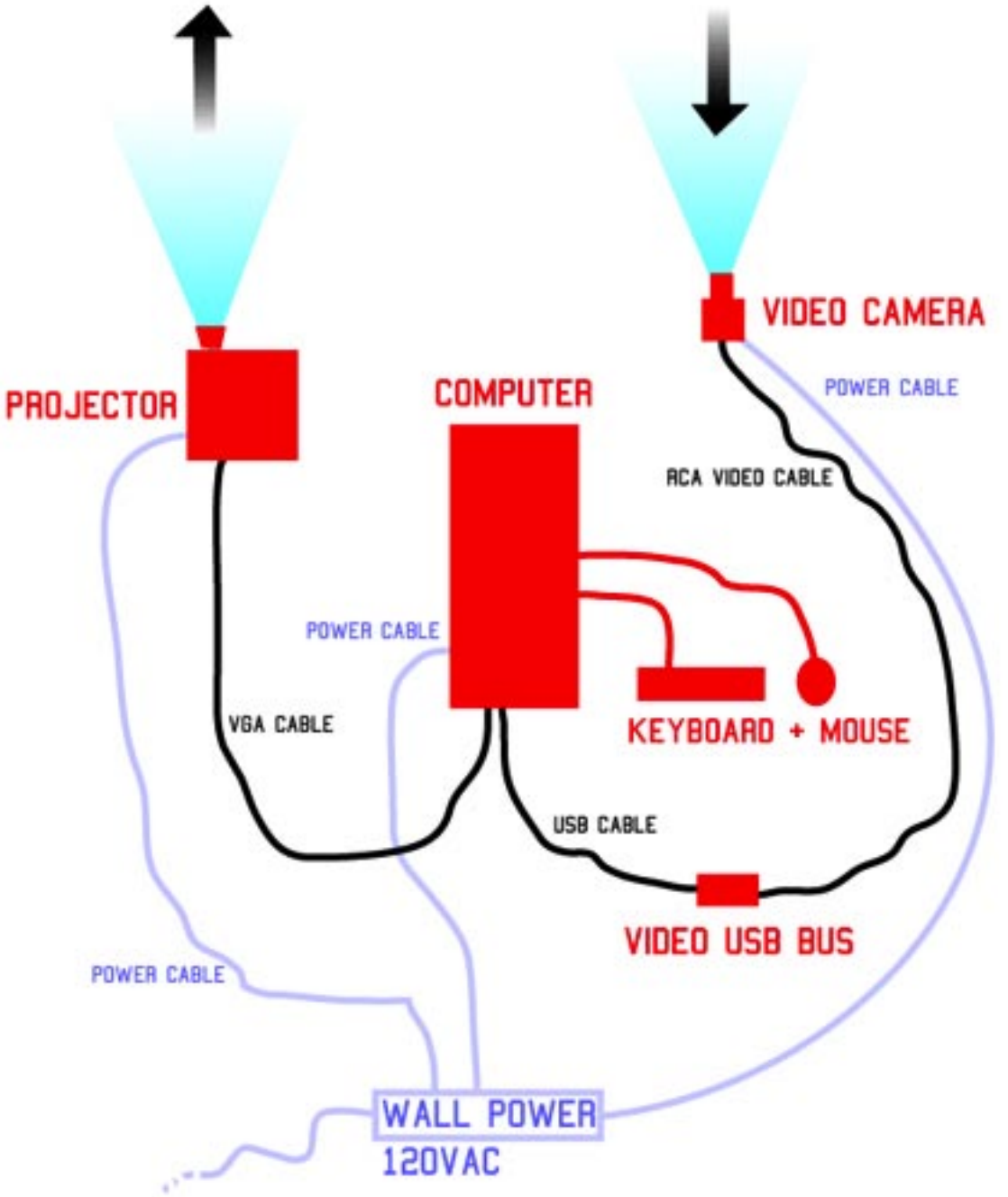


Résultat :

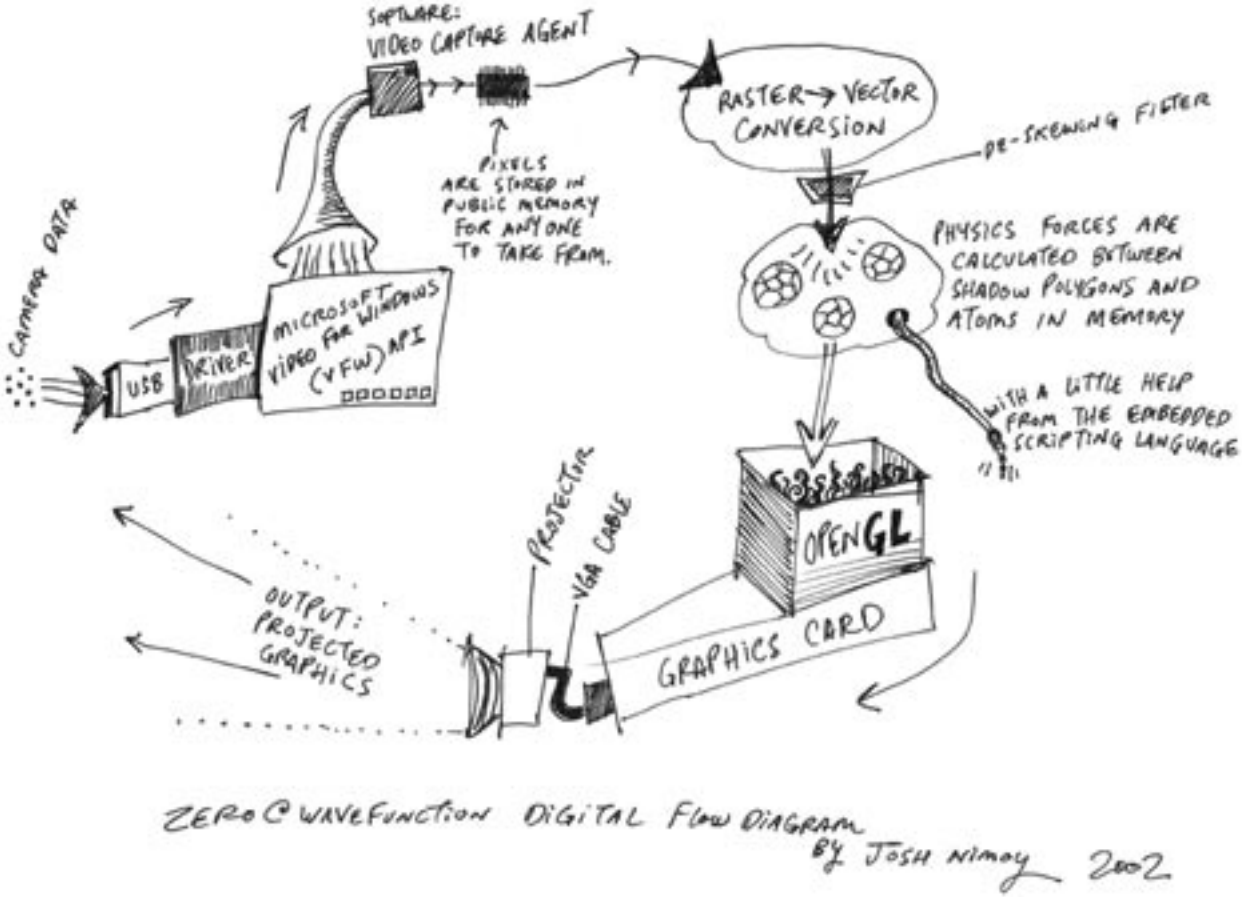
Après **traitement**, le programme fournit un **résultat** à l'utilisateur, soit par l'intermédiaire de l'écran, soit sous forme de fichiers. Ce retour de données (de l'ordinateur vers l'utilisateur) est aussi appelé **sortie**, **output** ou encore **affichage**.



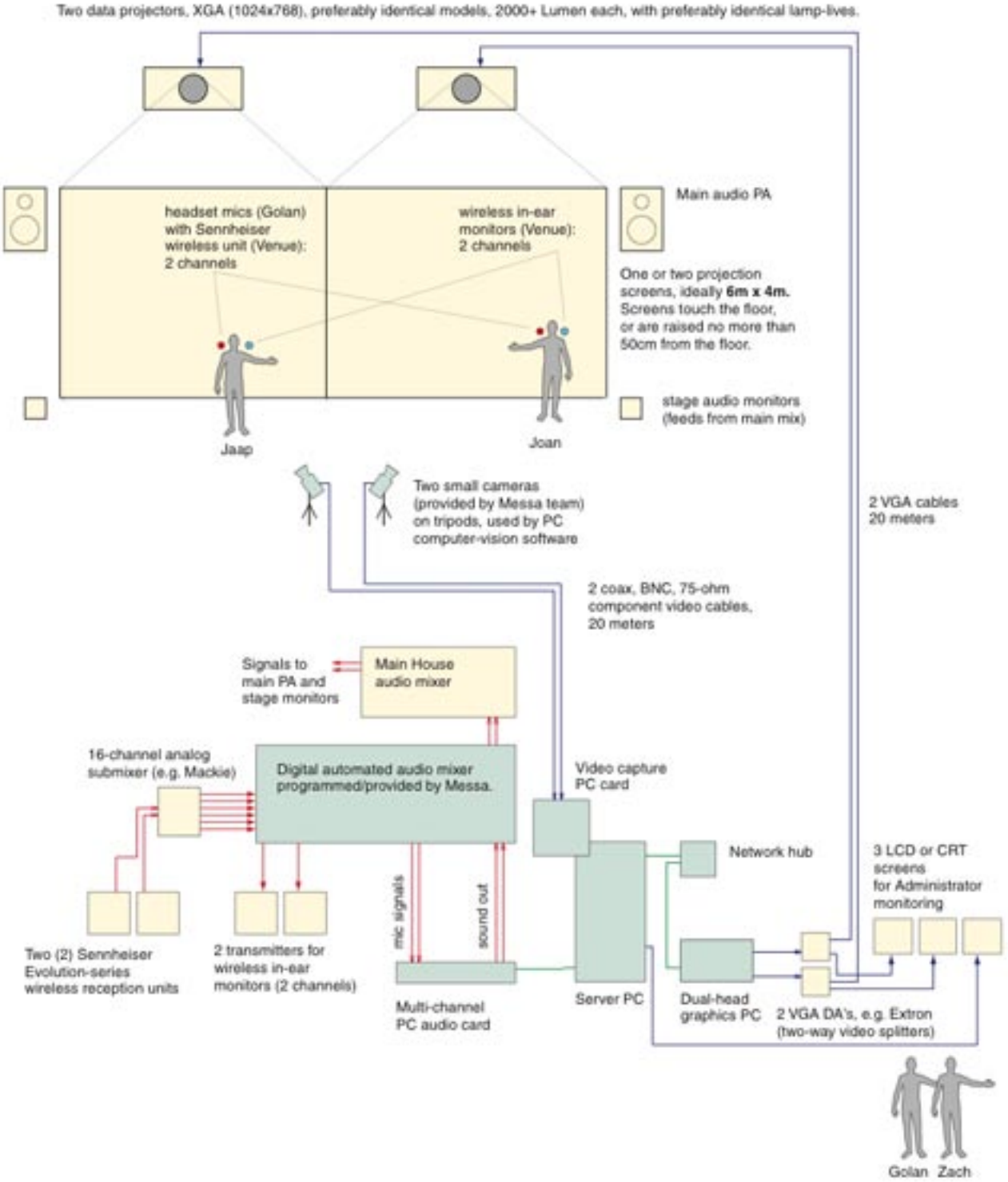
David Rokeby : softVNS (1986-1990),
logiciel et environnement interactif.
http://homepage.mac.com/davidrokeby/vns.html



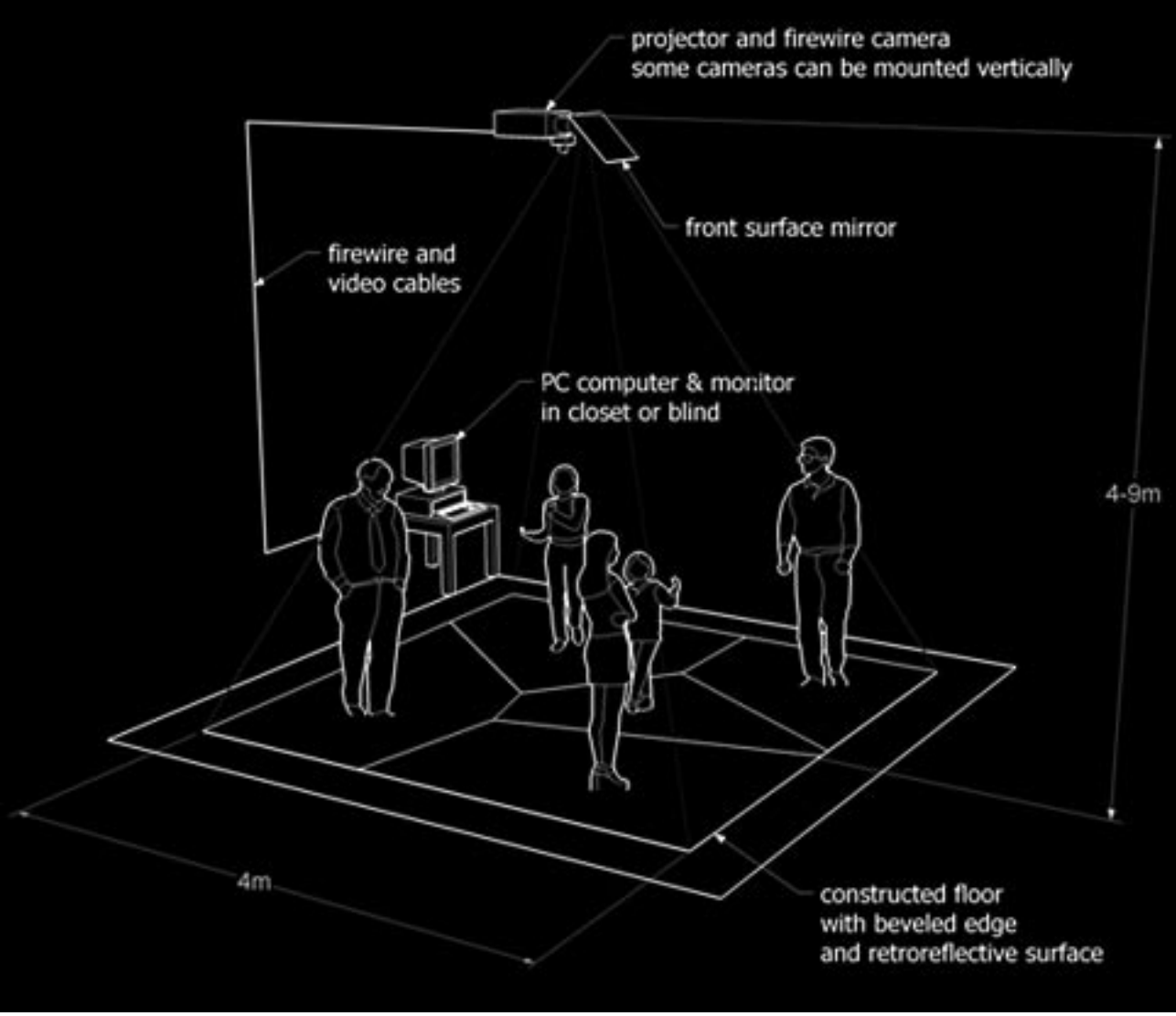
Josh Nimoy : Zero@wavefunction (1998),
logiciel et environnement interactif.
http://www.jtnimoy.com/zerowave/



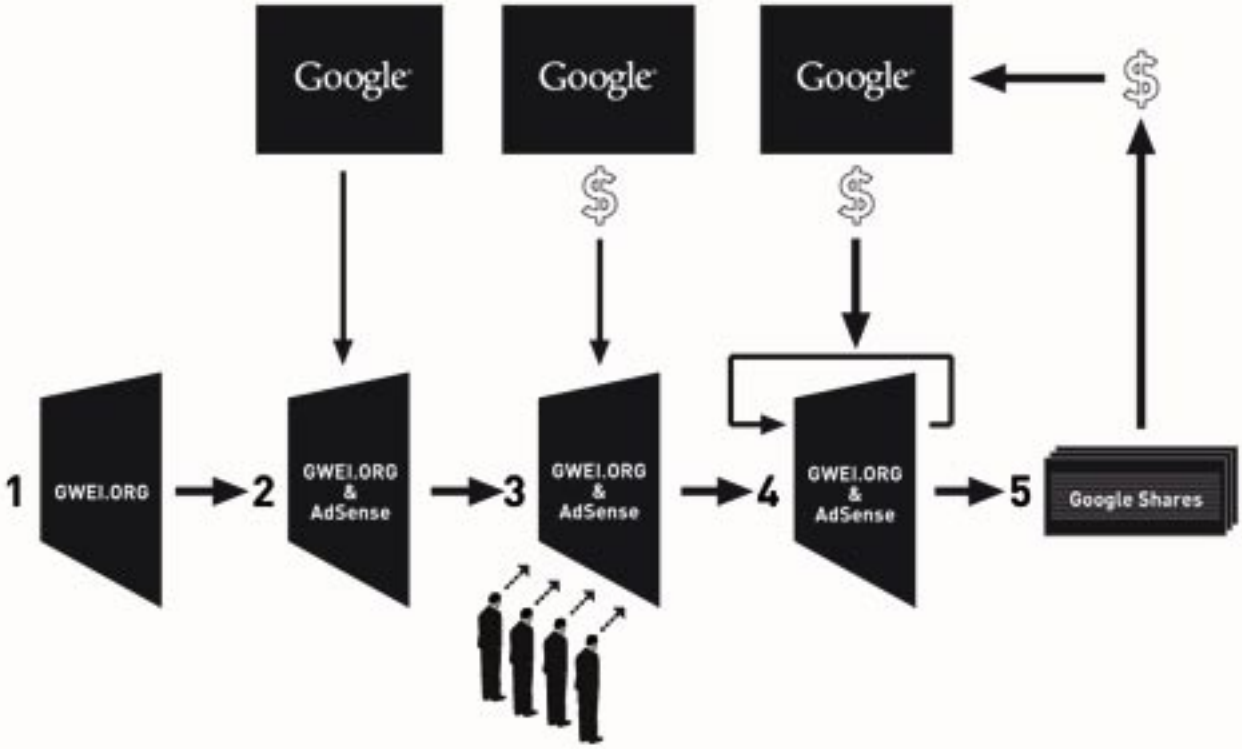
Josh Nimoy : Zero@wavefunction (1998),
logiciel et environnement interactif.
http://www.jtnimoy.com/zerowave/



Golan Levin & Zachary Lieberman : Messa di voce (2003),
logiciel, installation et performance interactive.
http://www.tmema.org/messa/



Scott Snibbe : Boundary functions (1998),
logiciel et installation interactive.
http://www.snibbe.com/



1- www.gwei.org website designed for Google advertismenl and for our clicking robot.

2- We set up an AdSense account with Google serving targeted ads on the web-marketing platform gwei.org

3- Our Ubermorgen.com and Neural.it network (community) is informed and invited to visit (create impressions for ads) and click trough : generating cash.

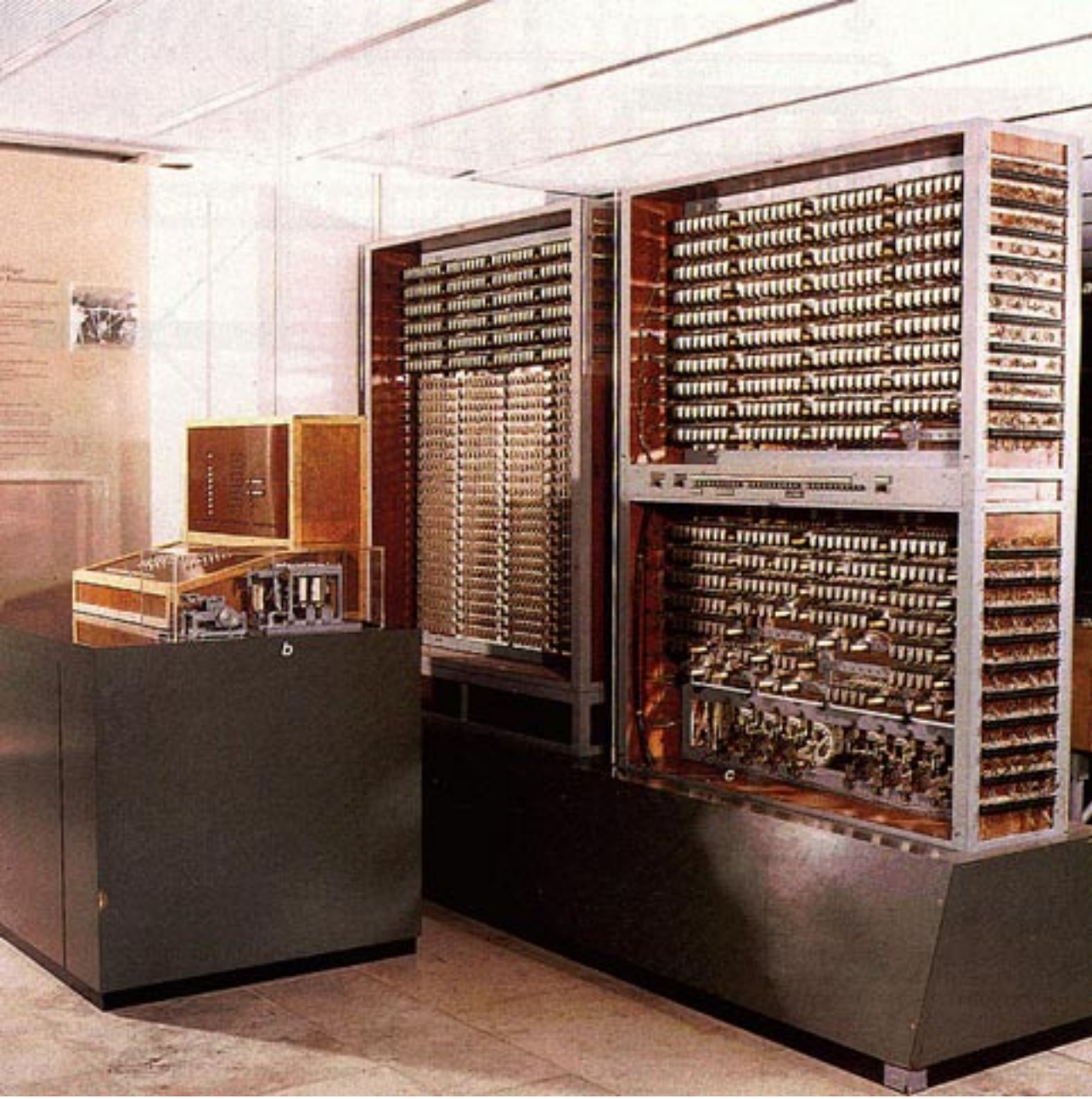
4- Additionally a program (php-robot) simulates visitation and clicks and therefore generates cash/income for GWEI.

5- With the cash we receive from Google we buy Google shares.

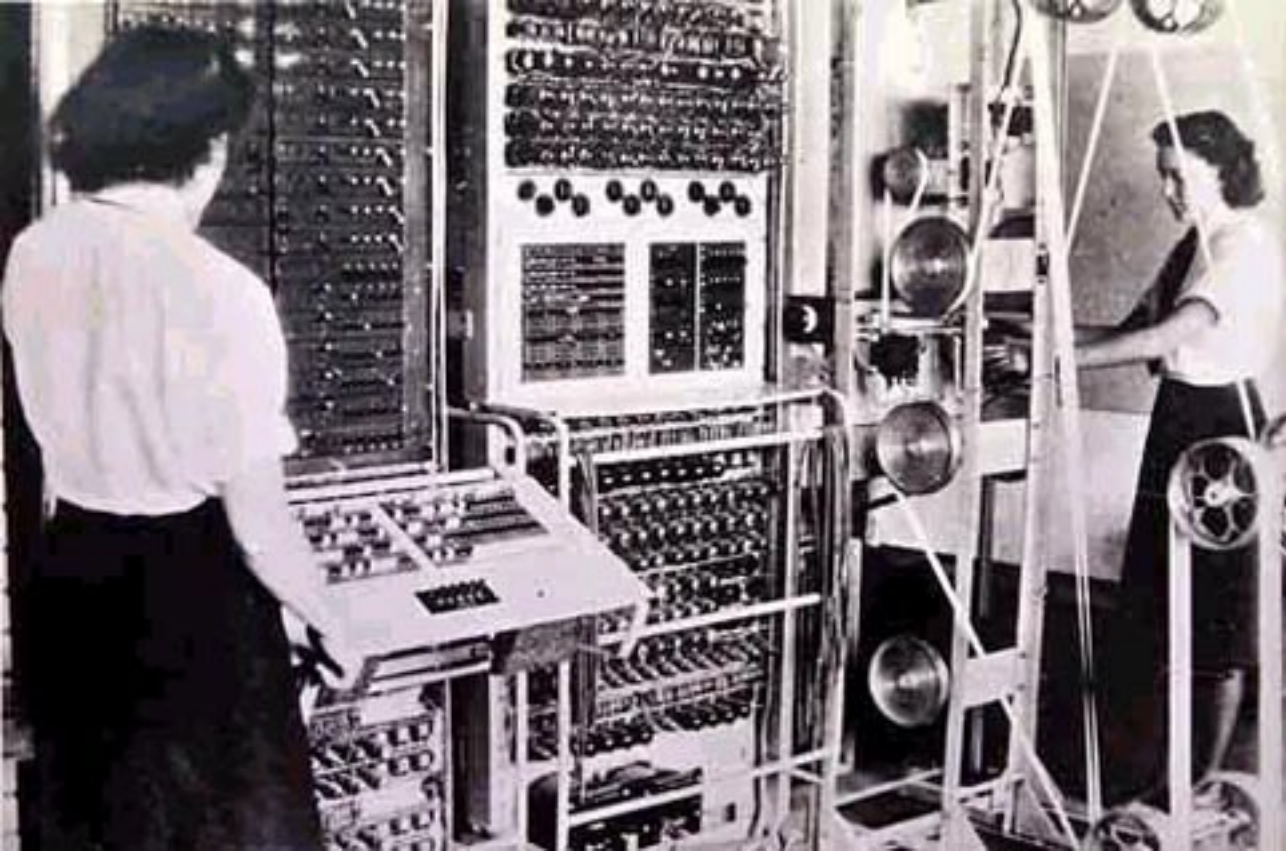
Google Will Eat itself thru advertisment.

Ubermorgen : GWEI - Google Will Eat itself (2005),
logiciel et website.
http://www.gwei.org/
http://www.ubermorgen.com/

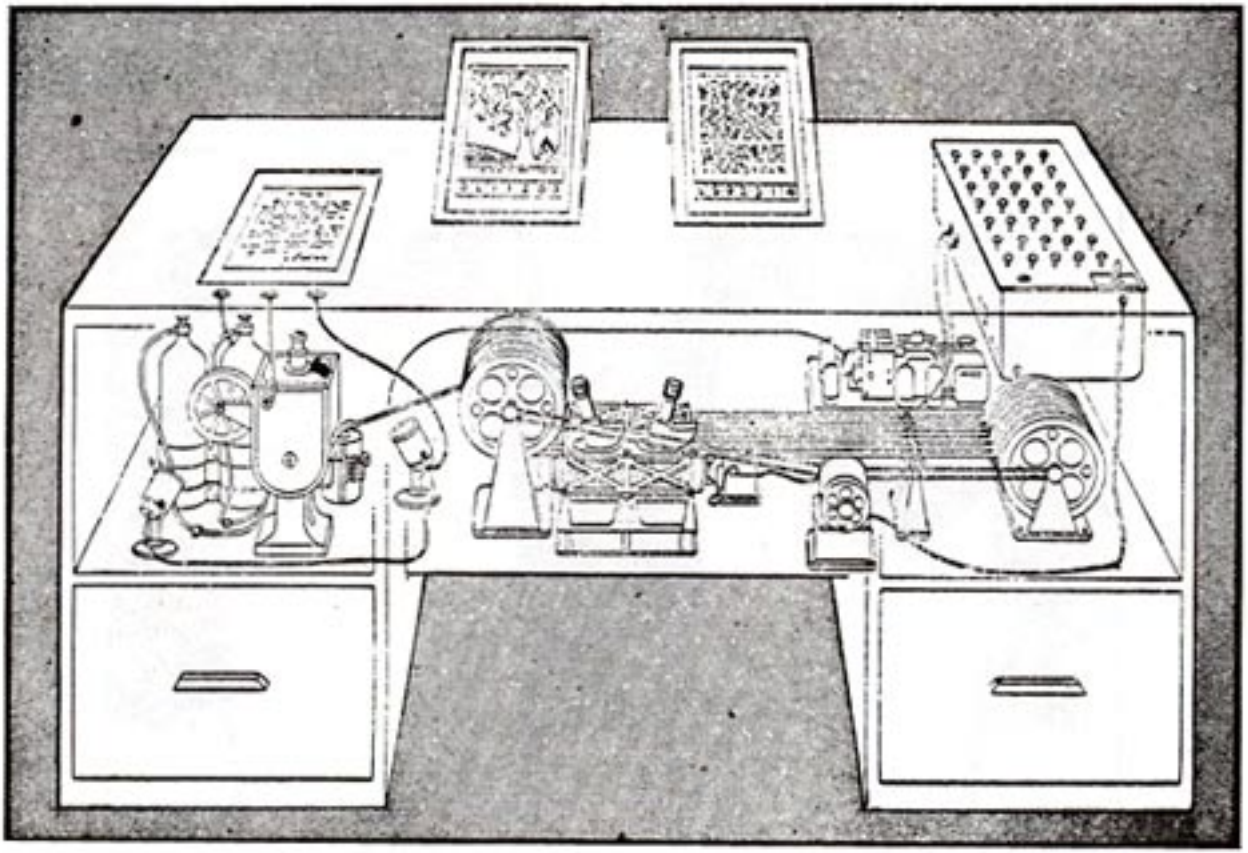
Fonctionnement d’un ordinateur.



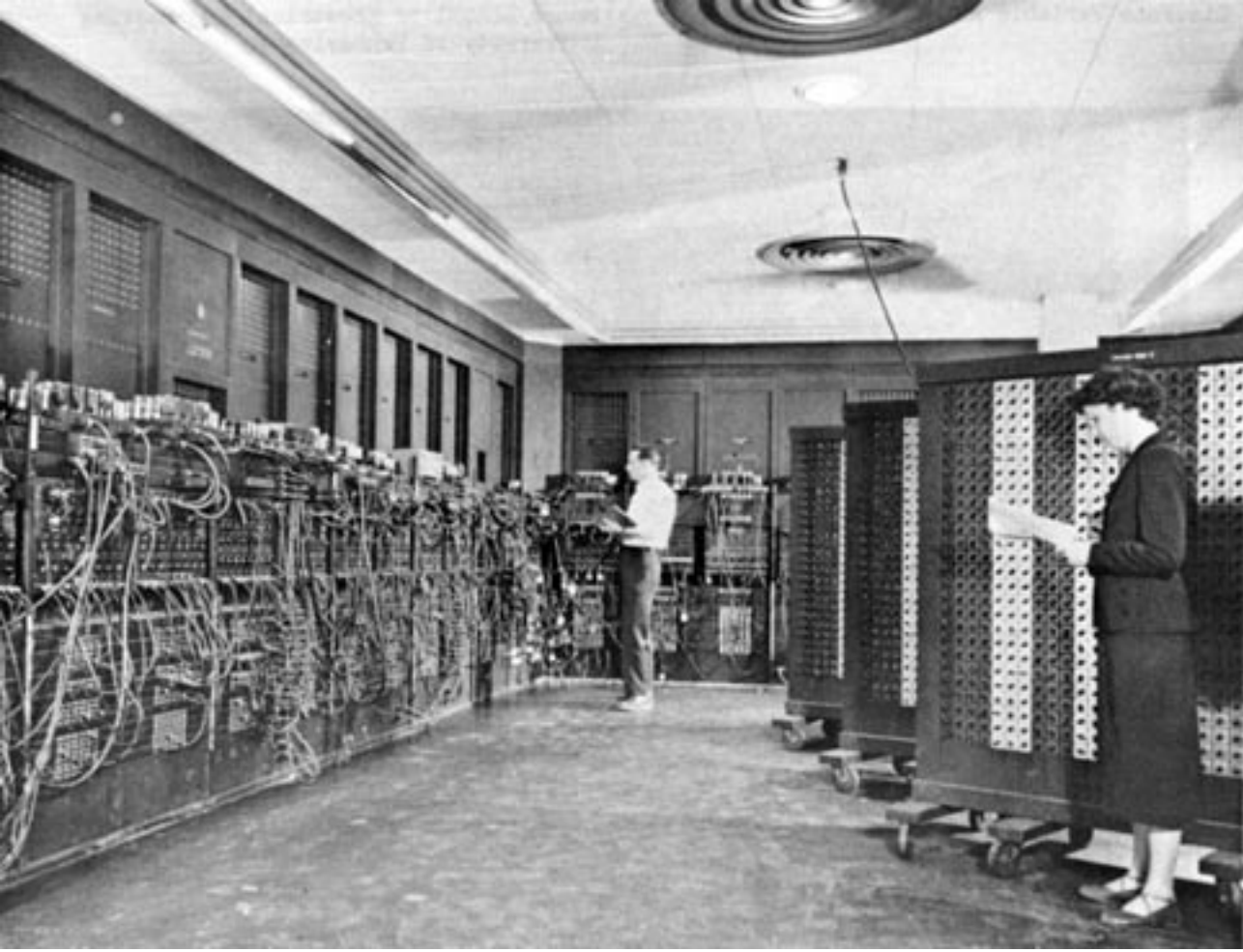
Konrad Zuse : Z3 (1941).
Premier ordinateur programmable mécaniquement
(ici une reconstruction vers 1960).
http://irb.cs.tu-berlin.de/~zuse/Konrad_Zuse/en/Rechner_Z3.html



Tommy Flowers : Colossus (1944).
Premier ordinateur électronique programmable.
http://en.wikipedia.org/wiki/Colossus_computer

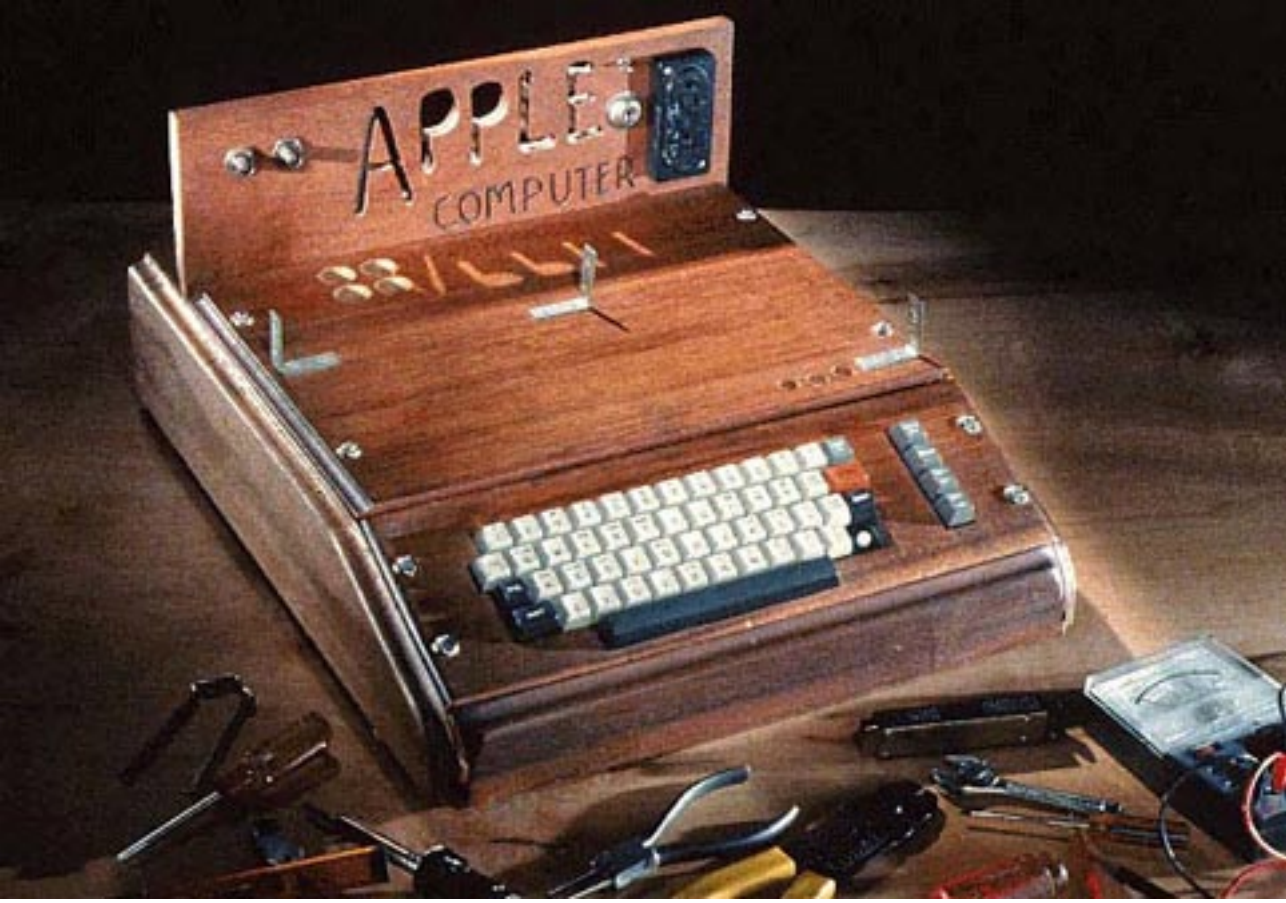


Vannevar Bush : Memex (Memory extender) (1945).
Ordinateur analogique théorique et visionnaire.
http://en.wikipedia.org/wiki/Memex



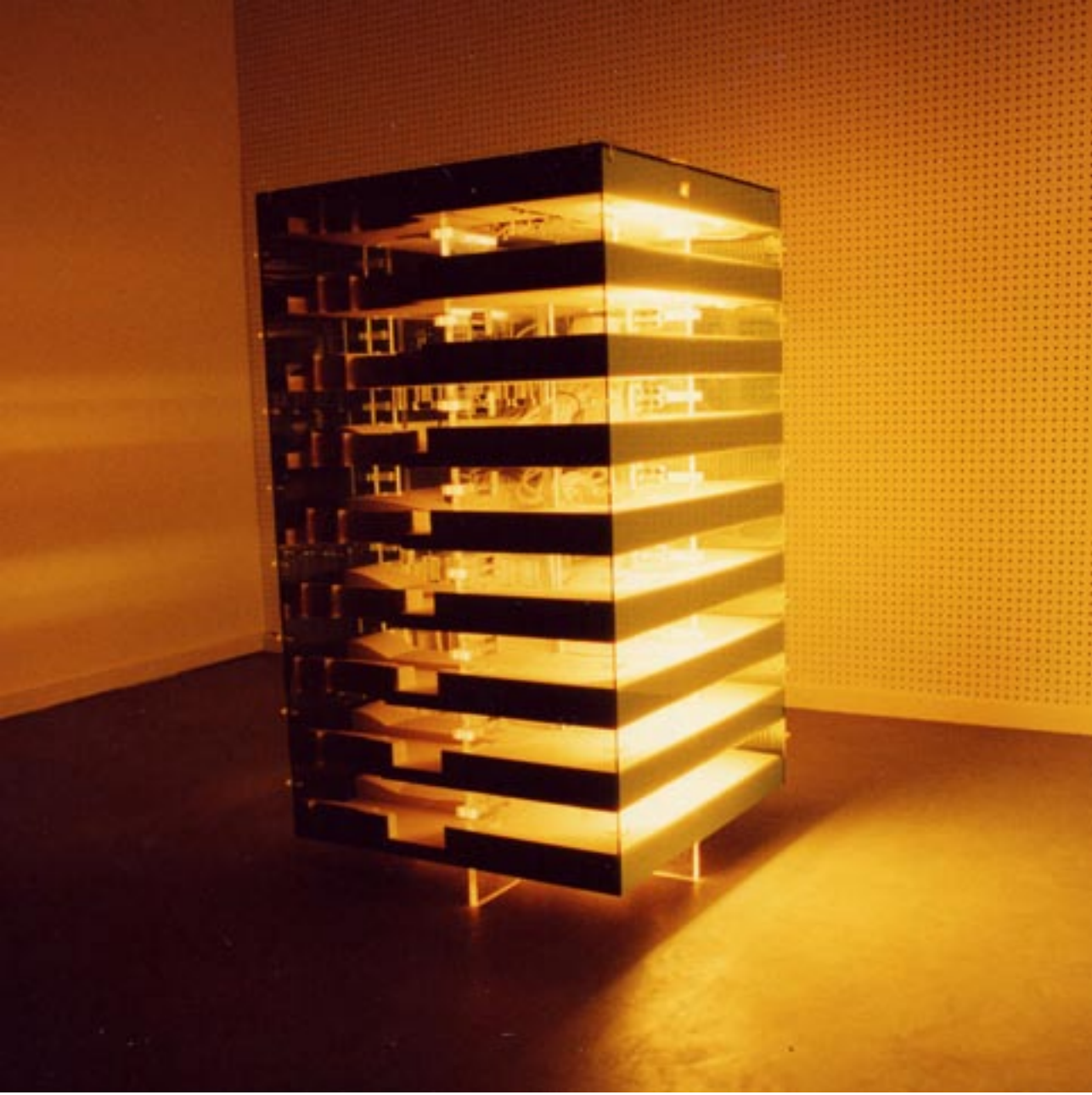
John William Mauchly & J. Presper Eckert : ENIAC
(Electronic Numerical Integrator And Computer) (1946).
Premier ordinateur entièrement électronique.
http://fr.wikipedia.org/wiki/ENIAC

John William Mauchly & J. Presper Eckert : UNIVAC I
(UNIVersal Automatic Computer I) (1951).
Premier ordinateur commercial.
http://fr.wikipedia.org/wiki/UNIVAC_I



Steve Jobs et Steve Wozniak : Apple I (1976).
Premier ordinateur individuel à être conçu
pour être combiné à un clavier et à un moniteur.
http://fr.wikipedia.org/wiki/Apple_I

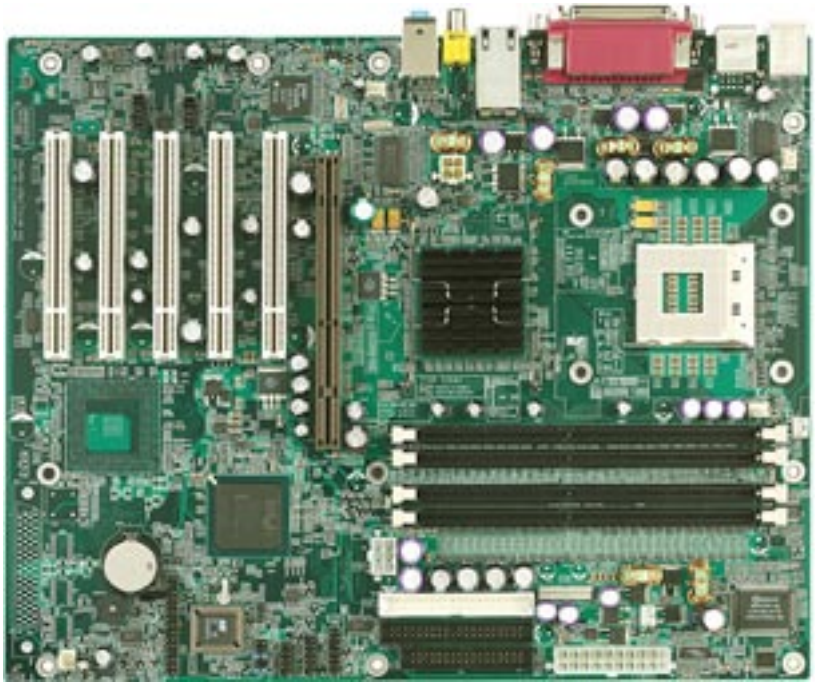
Steve Jobs et Steve Wozniak : Apple II (1977).
http://fr.wikipedia.org/wiki/Apple_Computer



Markus Popp : Oval Process Terminal (2004),
container, écran LCD, Mac G4, logiciel.
http://www.medienkunstnetz.de/works/oval-process/

Schématiquement, un ordinateur est composé de trois parties distinctes:

1- la **Mémoire Centrale**.
2- l'**Unité Centrale**.
3- les **Périphériques**.



← Processeur (unité centrale)

← Mémoire

La carte-mère permet la connexion des éléments essentiels.

Pour en savoir plus :
http://www.commentcamarche.net/pc/carte-mere.php3

1- la **Mémoire Centrale** permet de mémoriser les programmes pendant le temps nécessaire à leur exécution. On y trouve également les informations temporaires manipulées par ces programmes: données après leur introduction, résultats avant leur communication à l'extérieur, informations intermédiaires apparaissant pendant le déroulement d'un programme.

La mémoire centrale correspond à ce que l'on appelle la **Mémoire Vive** ou **RAM** (Random Access Memory, mémoire à accès direct).

Contrairement au stockage de données sur une mémoire de masse telle que le disque dur, la mémoire vive est volatile, c'est-à-dire qu'elle permet uniquement de stocker des données tant qu'elle est alimentée électriquement. Ainsi, à chaque fois que l'ordinateur est éteint, toutes les données présentes en mémoire sont irrémédiablement effacées.

Plus précisémment, on peut distinguer :

- les **Registres** : petites mémoires rapides de 8, 16, 32 ou 64 bits. Suivant le type de processeur, le nombre global de registres peut varier d'une dizaine à plusieurs centaines.

- la **Mémoire Cache** (mémoire tampon) : mémoire rapide permettant de réduire les délais d'attente des informations stockées en mémoire vive.

Pour en savoir plus :
http://www.commentcamarche.net/pc/memoire.php3

2- L'**Unité Centrale** (ou processeur, ou CPU - Central Processing Unit, ou UCT - Unité Centrale de Traitement) est la partie active de l'ordinateur. Elle est chargée de prélever une à une chaque instruction de programme située en mémoire centrale et de l'exécuter.

Plus précisémment, on peut distinguer deux sortes d'instructions:

- celles qui agissent sur des informations situées en mémoire centrale. Ce sont elles qui permettent de réaliser le traitement escompté.

- celles qui assurent la communication ou l'archivage d'information. Elles réalisent en fait un échange d'information entre la mémoire centrale et d'autres appareils nommés **Périphériques**.

Pour en savoir plus :
http://www.commentcamarche.net/pc/processeur.php3

3- Les **Périphériques** désignent tous les appareils susceptibles d'échanger des informations avec la mémoire centrale.

Ils sont de deux sortes :

- ceux qui assurent la communication entre l'homme et l'ordinateur (claviers, écrans, souris...)

- ceux qui assurent l'archivage d'information (disques durs, sticks USB...). Ces derniers ont un rôle de mémorisation d'information au même titre que la mémoire centrale, dont ils constituent ainsi une sorte de prolongement.

Pour en savoir plus :
http://www.commentcamarche.net/pc/peripherique.php3

Lorsqu'un ordinateur **exécute** un programme, son travail consiste en grande partie à **gérer la mémoire** :
- soit pour y **lire** une instruction.
- soit pour y **stocker** une information.

En ce sens, l'ordinateur peut être perçu comme un **robot** qui agit en fonction d'**ordres** qui lui sont fournis.

Ces actions sont en nombre limité :

- **Déposer** ou **lire** une information dans une case mémoire.
- **Exécuter** des opérations simples telles que l'addition ou la soustraction.
- **Comparer** des valeurs.
- **Communiquer** une information élémentaire.
- **Coder** l'information.

Déposer ou lire une information dans une case mémoire.

La mémoire est formée d'éléments, ou **cases-mémoire,** qui possèdent chacune un numéro (une **adresse**).

Chaque case-mémoire est en quelque sorte une **boîte aux lettres** pouvant contenir une information (une lettre). Pour y déposer cette information, l'ordinateur (le facteur) doit connaître l'adresse de la boîte.

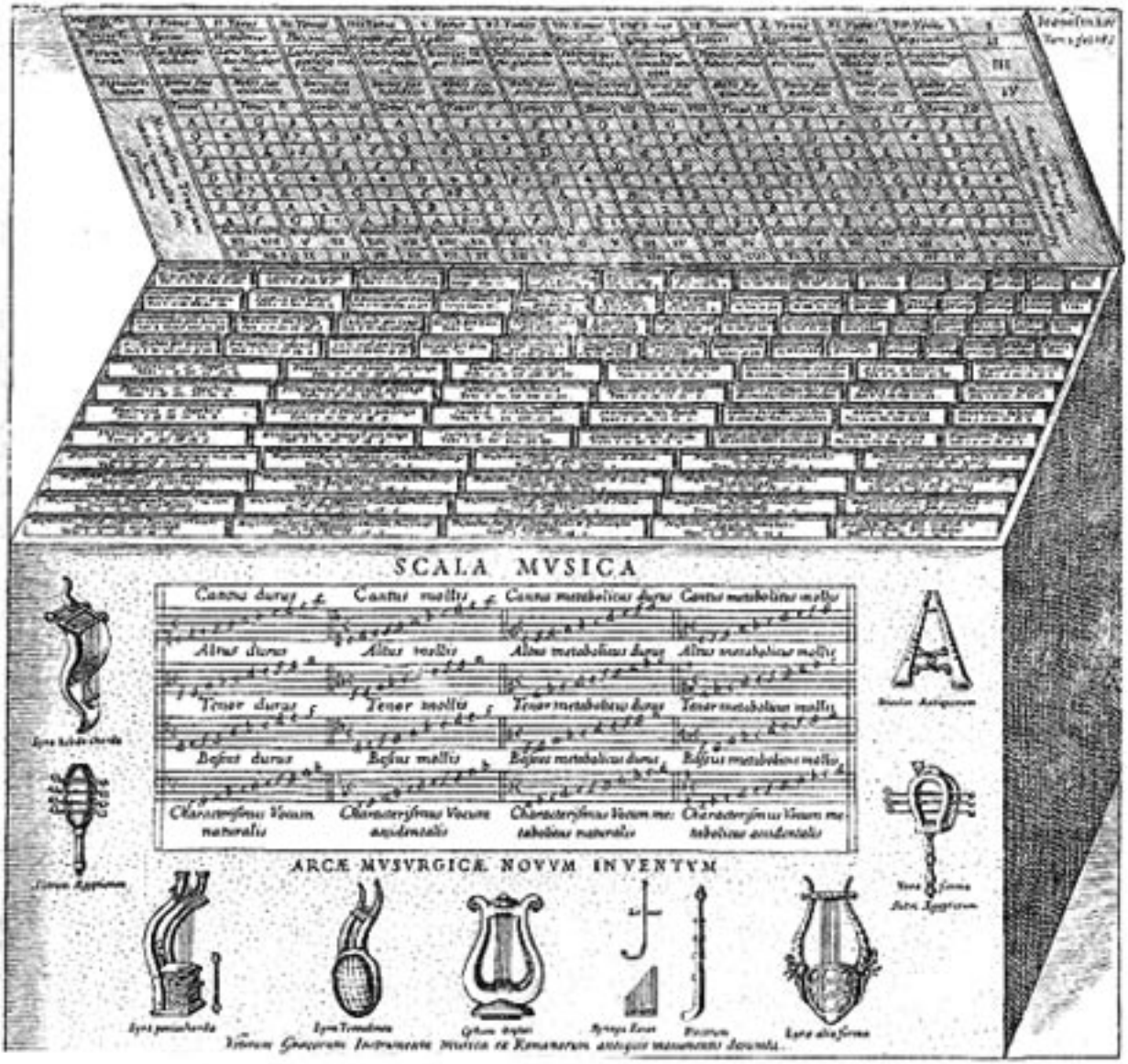
Lorsque le robot place une information dans une case mémoire, il mémorise l'adresse où se situe celle-ci afin de retrouver l'information en temps nécessaire. Le robot sait déposer une information dans une case, mais il ne sait pas la retirer (au sens de prendre un courrier déposé dans une boîte aux lettres).

Lorsque le robot prend l'information déposée dans une case mémoire, il ne fait que la lire. En aucun cas il ne la retire ni ne l'efface. L'information lue reste toujours dans la case mémoire.

Remarque : pour effacer une information d'une case mémoire, il est nécessaire de placer une nouvelle information dans cette même case. Ainsi, la nouvelle donnée remplace l'ancienne, et l'information précédente est détruite.

Les **cases-mémoires** (aussi appelées **Registres**) de la mémoire centrale peuvent être comparées aux tiroirs d'une armoire de bureau ou à des casiers à courrier.

Jacques Tati : Playtime (1967).
Une unité centrale de calcul ?
http://www.tativille.com/



Athanasius Kircher (1601-1680) a inventé un système destiné à engendrer des partitions musicales, ce qui fait de lui le père de la musique algorithmique générative. Il est également l'auteur de concepts d'instruments de musique automatisés.

http://en.wikipedia.org/wiki/Athanasius_Kircher
http://kircher.stanford.edu/

Exécuter des opérations simples
telles que l'addition ou la soustraction.

La machine **lit** et **exécute** les opérations dans l'ordre où elles lui sont fournies.

Pour faire une addition, il va chercher les valeurs à additionner dans les **cases-mémoire** appropriées (stockées, par exemple, aux adresses **a** et **b**) et réalise ensuite l'opération demandée. Il enregistre alors le résultat de cette opération dans une case d'adresse **c**.

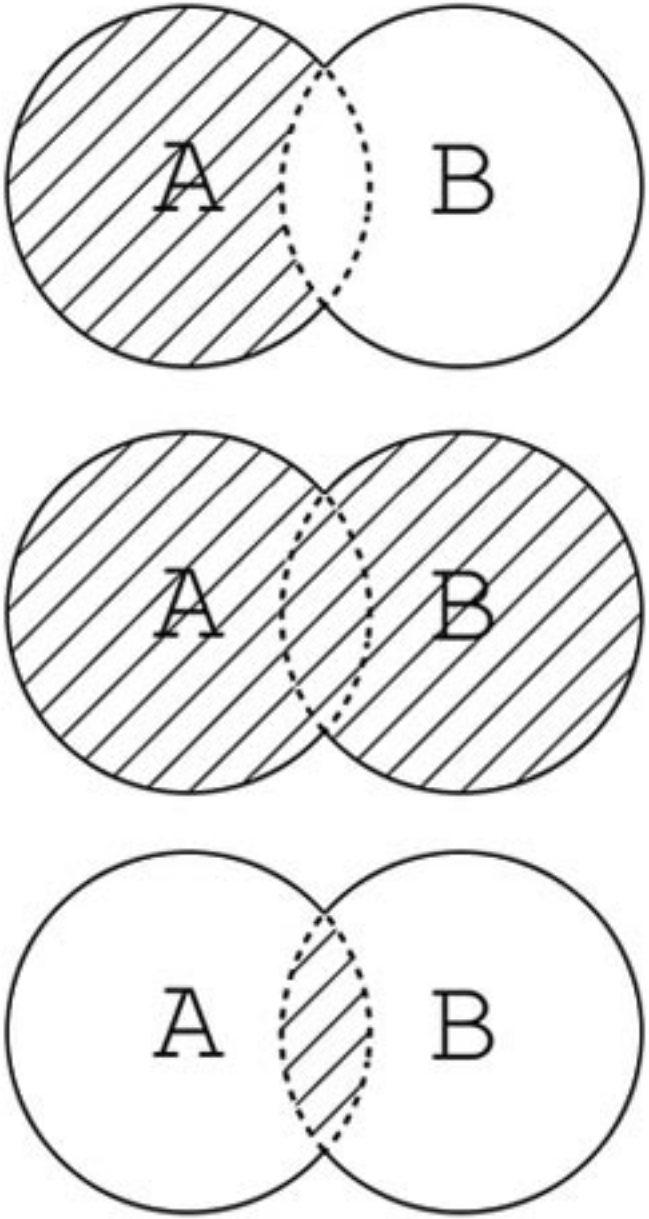
De telles opérations sont décrites à l'aide d'ordres, appelés aussi **instructions**.

Comparer des valeurs.

La machine est capable de **comparer** deux valeurs entre elles pour déterminer si l'une d'entre elles est plus grande, plus petite, égale ou différente de l'autre valeur. Grâce à la comparaison, la machine peut tester une condition et exécuter un ordre plutôt qu'un autre, en fonction du résultat du test.

La réalisation d'une comparaison ou d'un test fait que le robot ne peut plus exécuter les instructions dans leur ordre d'apparition. En effet, suivant le résultat du test, il doit rompre l'ordre de la marche à suivre, en sautant une ou plusieurs instructions. C'est pourquoi il existe des **instructions** particulières dites **de branchement**. Grâce à ce type d'instructions, le robot est à même non seulement de sauter des ordres mais aussi de revenir à un ensemble d'opérations afin de les répéter.

Pour en savoir plus :
http://www.commentcamarche.net/logic/intro.php3



Algèbre de Boole : fonctions logiques OU, ET, NON.
http://fr.wikipedia.org/wiki/Algèbre_de_Boole_%28logique%29

Communiquer une information élémentaire.

Un programme est essentiellement un outil qui traite l'information.

L'utilisateur **transmet** cette information à l'ordinateur par l'intermédiaire du clavier ou de la souris.

Cette transmission de données à l'ordinateur est appelée **entrée**, **input**, **saisie** ou encore **lecture de données**.

Après **traitement**, le programme fournit un **résultat** à l'utilisateur, soit par l'intermédiaire de l'écran, soit sous forme de fichiers.

Ce retour de données (de l'ordinateur vers l'utilisateur) est appelé **sortie**, **output**, **affichage** ou encore **écriture de données**.

Coder l'information.

De par la nature de ses composants électroniques, le robot ne perçoit que deux états: composant **allumé** et composant **éteint**. De cette perception découle le **langage binaire**, qui utilise par convention les deux symboles **0** (éteint) et **1** (allumé). Ne connaissant que le 0 et le 1, l'ordinateur utilise un code pour représenter une information aussi simple qu'un nombre entier ou un caractère. Ce code est un programme, qui différencie chaque type d'information et transforme une information (donnée numérique ou alphabétique) en valeurs binaires. À l'inverse, ce programme sait aussi transformer un nombre binaire en valeur numérique ou alphabétique. Il existe autant de codes que de types d'informations. Cette différenciation du codage (en fonction de ce qui doit être représenté) introduit le concept de **type de données**.

Toute information fournie à l'ordinateur est, au bout du compte, codée en binaire. l'information peut être un simple nombre ou une instruction de programme.

Exemple : Pour additionner deux nombres, l'ordinateur fait appel aux trois éléments qui lui sont nécessaires pour réaliser cette opération.
Ces éléments sont les suivants:
- le code binaire représentant l'opération d'addition (par exemple 0101) ;
- l'adresse de la case mémoire où est stocké le premier nombre (par exemple 011101) ;
- l'adresse de la case mémoire où se trouve la deuxième valeur (par exemple 010101).

Pour finir, l'instruction d'addition de ces deux nombres s'écrit en assemblant les trois codes binaires (soit, dans notre exemple, 0101011101010101).

Remarque :
Le code binaire associé à chaque code d'opération (addition, test, etc.) n'est pas nécessairement identique d'un ordinateur à un autre. Ce code binaire est déterminé par le constructeur de l'ordinateur. De ce fait, une instruction telle que l'addition de deux nombres n'a pas le même code binaire d'une machine à une autre. Il existe donc, pour un même programme, un code binaire qui diffère suivant le type d'ordinateur utilisé.

Pour en savoir plus :
http://www.commentcamarche.net/base/binaire.php3

La notion d'INSTRUCTION.

Les instructions exécutées par l'ordinateur sont des traitements élémentaires simples (comparaison de valeurs, addition, soustraction, multiplication ou division) opérant sur des données que l'ordinateur puise dans sa mémoire.

Ces données, préalablement transmises par l'utilisateur ou par le programme lui-même, sont des assemblages de bits (valeurs de 0 ou de 1).

Ces assemblages sont de deux types:
- l'**octet**, qui correspond à huit bits;
- le **mot**, terme plus général qui comprend un nombre de bits variant selon les ordinateurs (8, 16, 32, 60...)

Les instructions sont également représentées comme des suites de 0 et de 1 dans la mémoire de l'ordinateur.

Pour en savoir plus :
http://www.commentcamarche.net/langages/langages.php3

La notion de VARIABLE.

Une instruction machine effectue des opérations sur des **valeurs** repérées par leur **adresse**. Une telle adresse est représentée par un **nom**.

Une **variable** est précisémment ce nom qui sert à désigner un **emplacement** donné de la mémoire centrale.

Cette notion, simple en apparence, contribue considérablement à faciliter la réalisation de programmes. Elle vous permet, en effet, de manipuler des valeurs sans avoir à vous préoccuper de l'emplacement qu'elles occuperont effectivement en mémoire: il vous suffit simplement de leur choisir un nom.

Bien entendu, la chose n'est possible que parce qu'il existe un programme de traduction (compilateur ou interprète) de votre programme vers le langage machine : c'est lui qui attribuera une adresse à chaque variable.

Pour en savoir plus :
http://www.commentcamarche.net/langages/structure.php3

La notion de TYPE.

Les informations conservées en mémoire sont toujours codées en langage binaire. Cela est vrai en particulier pour le contenu des variables. Comme il est nécessaire de pouvoir conserver des informations de nature différente (par exemple des **nombres** et des **lettres**), il faut employer plusieurs codes différents. Dès lors, la connaissance du contenu binaire d'une variable ne suffit pas pour déterminer l'information correspondante. Il est nécessaire de savoir, en outre, comment la valeur qui s'y trouve a été codée.
Cette distinction correspond à la **notion de type**.

Ainsi, une variable destinée à recevoir des valeurs telles que **123** ou **12,45** est de type **numérique**. Une variable destinée à contenir des **lettres** sera dite de type **caractère**.

Le type limite les opérations : les opérations arithmétiques (addition, soustraction, multiplication, division), possibles avec des variables numériques, n'ont aucun sens pour des variables de type caractère.
Par contre, les comparaisons seront possibles pour les deux types.

En résumé, le type d'une variable définit:
- la **nature des informations** qui seront représentées dans la variable (numériques, caractères).
- le **codage** utilisé.
- les **limitations** concernant les valeurs qui peuvent être représentées.
- les **opérations** réalisables avec les variables correspondantes.

Pour en savoir plus :
http://www.commentcamarche.net/langages/structure.php3

Dans la vie courante, les êtres humains n’ont pas besoin de préciser le type des informations qu’ils échangent, car il est explicite : nous comprenons que 23 représente un nombre tandis que DUPONT est une suite de caractères. Ce n’est pas le cas pour l’ordinateur : il faut lui expliquer la différence.

Annexes.

Modèles et simulation.

«[...] Les matériaux et les outils numériques sont essentiellement d'ordre symbolique et langagier. On ne peut cependant les considérer simplement comme des «immatériaux» car les objets qu'ils produisent, si virtuels qu'ils soient, font bien partie du monde réel et agissent sur nos sens. Ce qui fait donc la spécificité des technologies numériques n'est pas leur immatérialité mais leur programmaticité, c'est-à-dire le fait qu'elles se réduisent à des programmes informatiques capables d'être traités automatiquement par la machine ordinateur. Il est vrai que le plus souvent les utilisateurs n'ont pas d'accès direct à la conception et à l'écriture de ces programmes, encore qu'ils soient obligés de donner leurs instructions à la machine sous la forme d'un langage intermédiaire (icônes, menus de fonctions, etc.) qu'ils doivent connaître, mais ce sont toujours des programmes qu'ils manipulent indirectement: quoique le langage de la programmation soit symbolique et abstrait, il reste cependant différent du langage naturel. En effet, tous les programmes sont élaborés à partir de modèles logiques et mathématiques issus des sciences les plus diverses. Le tracé d'une simple droite sur l'écran requiert un modèle géométrique déjà complexe. Le tracé des courbes est encore plus sophistiqué et emprunte à la géométrie analytique. La construction d'une forme en trois dimensions fait appel à des notions parfois très élaborées de géométrie dans l'espace. Les couleurs sont synthétisées grâce à des modèles colorimétriques. Les lumières, les ombres, les reflets, le rendu des surfaces, sont créés avec des modèles issus de l'optique. Le mouvement des objets emprunte à la physique, parfois à la mécanique des fluides. La production de formes végétales emprunte à la botanique. Les modèles intervenant dans les programmes interactifs sont aussi très complexes et peuvent s'inspirer des sciences du vivant et du connexionnisme (domaine de l'intelligence artificielle traitant des réseaux neuronaux) [...]».

***Edmond Couchot & Norbert Hilaire**,*
«L'art numérique», Paris, Champs-Flammarion, 2003, pp 25-26.

***Edmond Couchot** est théoricien de l'art et professeur à l'université Paris-VIII.*
***Norbert Hilaire** est historien d'art et professeur à l'université de Nice.*

La science comme présence efficiente.

«[...] Comme nous l'avons montré, techniquement, l'image numérique est étroitement dépendante des processus programmatiques qui la produisent. Or ces modèles de simulation numérique utilisés dans les programmes sont, comme tout modèle scientifique, déjà des interprétations formalisées du réel. Le modèle d'un cercle, par exemple, et sa visualisation informatique n'ont rien de commun avec un cercle dessiné au compas. Certains mouvements propres à des objets de synthèse sont réglés par des équations qui lient le temps et l'espace et empruntent à une physique qui n'existait pas avant Galilée. Il en résulte que sur un écran d'ordinateur on ne peut figurer, donner une forme visible, sensible, qu'à ce qui est déjà intelligible, déjà interprétation rationnelle du monde. Les artistes se trouvent alors dans la délicate nécessité de créer du sensible (des formes artistiques) avec de l'intelligible (des programmes informatiques), en quelque sorte des résidus applicatifs de la science [...]».

***Edmond Couchot & Norbert Hilaire**,*
«L'art numérique», Paris, Champs-Flammarion, 2003, pp 33-34.

***Edmond Couchot** est théoricien de l'art et professeur à l'université Paris-VIII.*
***Norbert Hilaire** est historien d'art et professeur à l'université de Nice.*

Calculs, algorithmes, machines universelles.
***Pierre Levy**, «La machine univers», Paris, Points-Sciences, 1987, pp 71-75.*

«[...] Une machine n'est pas nécessairement le moteur d'un véhicule ou bien une chose lourde et bruyante qui transforme la matière en lui appliquant avec violence une force mécanique. Radios, télévisions, centraux téléphoniques montres à quartz et ordinateurs nous ont habitués à l'idée qu'une machine pouvait *traiter de l'information*, c'est-à-dire transformer suivant une loi déterminée un message d'entrée en message de sortie.

La différence entre un traitement industriel, par exemple, et un traitement informationnel réside dans la faible quantité d'énergie mise en jeu dans la sphère de l'information. Rigoureusement parlant, un traitement industriel est aussi informationnel, puisqu'il s'agit d'un processus par lequel des différences engendrent d'autres différences (ou par lequel une forme est donnée à une matière brute). Mais on réserve la notion de traitement de l'information aux processus mettant en oeuvre de petites énergies et qui servent souvent à connaître, surveiller, contrôler, commander, directement ou indirectement, des processus de niveau d'énergie supérieurée.

Le traitement d'information par excellence est le calcul. Au sens mathématique restreint, un calcul est un ensemble d'opérations arithmétiques. Nous retrouvons ici la notion d'opération, c'est-à-dire d'action organisée, méthodique, en vue de la production d'un effet déterminé. L'opération mathématique est une combinaison effectuée suivant des règles données sur des êtres mathématiques (par exemple des nombres, des ensembles, etc.) et admettant comme résultat un être mathématique bien déterminé. Si les êtres mathématiques sont convenablement représentés par des éléments physiques, et les règles de combinaisons parfaitement précisées, nous apercevons immédiatement la possibilité de *mécaniser* et d'*automatiser* les calculs.

Nous pouvons donner au mot calcul une extension plus large que son sens mathématique strict. On appellera alors calcul des opérations de tri, de classement, de permutation, de combinaison, de comparaison, de substitution, de transcodification (traduction d'un code à l'autre). Cette extension du sens du mot calcul est parfaitement légitime puisque toutes les opérations précitées peuvent se ramener à la combinaison, plus ou moins complexe, de deux ou trois opérations mathématiques fondamentales. Cela ne nous est pas familier, car lorsque nous nous livrons à des classements, des tris, des traductions d'un code à l'autre, nous agissons, du moins consciemment, de façon directe ou globale. Mais il faut savoir qu'il est souvent possible de décomposer des actions globales en quelques opérations élémentaires répétées un très grand nombre de fois et appliquées dans un ordre déterminé aux objets sur lesquels on opère. C'est ainsi que calcule un

ordinateur; ses circuits de base ne peuvent effectuer que quelques actions très simples, mais ces actions sont combinées entre elles et répétées de telle sorte que des calculs très complexes sont finalement réalisés.

Les objets sur lesquels agissent les circuits de l'ordinateur sont des impulsions électriques. La présence d'une charge électrique représente (pour nous) le chiffre 1, son absence le chiffre 0. Du fait qu'il est possible d'affecter un nombre à chaque lettre de l'alphabet et que tous les nombres peuvent s'exprimer en numération binaire, il est possible de représenter à l'intérieur d'un ordinateur tout ce qui peut s'écrire dans un alphabet ou se traduire par des chiffres. Ainsi, des circuits électriques simples jouant sur la présence ou l'absence de charge électrique, parce qu'ils représentent des opérateurs mathématiques fondamentaux agissant sur des nombres exprimés en base deux, effectuent des calculs extrêmement élaborés, traitent de l'information.

L'exécution d'un travail peut généralement se décomposer en une suite ordonnée d'actes élémentaires. La réalisation d'une multiplication, le classement de mots par ordre alphabétique, tout ce que nous avons appelé calcul, et qui se réduit à des opérations d'arrangement, de combinaison et d'organisation d'un nombre fini d'éléments, peut être rangé dans la classe des travaux décomposables en sous-travaux plus simples.

Pour nous, l'instruction: «classez ces mots par ordre alphabétique», a un sens. L'être humain est un opérateur dont la compétence est extrêmement étendue. Mais imaginons un opérateur dont la compétence est si restreinte qu'il ne peut effectuer que deux ou trois opérations et seulement sur les chiffres 0 et 1 : c'est le cas du circuit de base d'un ordinateur. Pour que l'ordinateur puisse effectuer ce traitement, il faudra donc non seulement que tous les mots soient traduits en suites de 0 et de 1, mais que l'instruction globale: «classez ces mots par ordre alphabétique», soit décomposée en instructions élémentaires exécutables par l'ordinateur. Mais s'il fallait, à chaque fois qu'il y a un calcul à faire, le décomposer en opérations élémentaires convenablement disposées pour qu'il soit effectué par l'ordinateur, il serait plus simple de le faire à la main.

C'est pourquoi les opérations les plus courantes, addition arithmétique, multiplication, etc., sont la plupart du temps déjà câblées dans la machine, c'est-à-dire que les circuits de base sont disposés de telle façon qu'ils réalisent automatiquement l'opération voulue. C'est aussi pourquoi on établit des plans de calcul à l'usage de l'ordinateur non pour *un* traitement déterminé mais pour *un ensemble* de traitements similaires ou une classe de traitements. Ce sont donc les exigences du traitement automatique de l'information qui conduisent les informaticiens (ou les usagers de la micro-informatique) à élaborer des algorithmes.

Un algorithme est une suite finie (car il faut que le calcul ne soit pas infini, aboutisse à un résultat) et *ordonnée* (convenablement disposée de

façon à aboutir au résultat voulu) de *règles* (ou d'instructions, ou d'opérations) en vue de résoudre une classe de problèmes (de réaliser un certain type de tâches et non pas *un* problème ni *une* tâche).

On dit qu'un problème à résoudre, une tâche à accomplir ont été *formalisés* lorsqu'on a établi la liste des opérations élémentaires, celle des objets sur lesquels s'effectuent ces opérations élémentaires, et qu'on a déterminé précisément dans quel ordre et sur quels objets doivent s'effectuer les opérations. La formalisation d'une tâche, d'un calcul, nécessite l'explicitation de tous ses aspects. Il y a plusieurs algorithmes ou plusieurs formalisations possibles d'un même calcul. Les algorithmes varient en fonction de la compétence de l'opérateur (les instructions « élémentaires» ne sont pas les mêmes) et du facteur optimisé dans l'exécution: plus ou moins grande rapidité, plus ou moins grand degré de généralité, etc.

Les algorithmes sont souvent déconcertants: une fois formalisées, les actions les plus familières perdent leur apparence habituelle. La perception globale disparaît au profit d'une extrême rigueur dans l'explicitation et la description; le découpage de la réalité n'y est pas fait autour de pôles de signification, mais suivant une logique purement opératoire.

Un programme informatique est donc un algorithme, ou un ensemble d'algorithmes, destiné à conduire entièrement l'exécution d'une tâche, souvent très complexe, à l'intérieur d'un ordinateur ou d'un système informatique. Chaque mot d'un programme ne doit pouvoir être interprété que d'une seule manière par la machine. La syntaxe (c'est-à-dire l'ordre dans lequel doivent se trouver les mots et les signes de ponctuation) d'un langage de programmation est extrêmement rigoureuse. Une virgule en moins ou mal placée, et le programme ne fonctionne pas comme il faut. Les programmes sont stockés dans la mémoire de la machine en même temps que les informations qui font l'objet du traitement. Le programme et les données ayant été introduits dans la machine, le traitement va se dérouler d'une façon entièrement automatique, sans intervention humaine, ce qui lui permet de s'exécuter à une très grande vitesse. En informatique, les mots «machine» ou «automate» désignent moins le dispositif physique qui effectue la transformation d'un message d'entrée en message de sortie que la structure logique de ce dispositif. La même «machine» (à faire des additions, par exemple) peut s'incarner aussi bien dans une calculatrice à roues dentées, que dans un microprocesseur ou une liste d'instructions que devrait suivre à la lettre un esclave humain parfaitement) obéissant. En fait, une «machine» est un algorithme, un programme [...]».

Pierre Levy,
«La machine univers», Paris, Points-Sciences, 1987, pp 71-75.

Pierre Levy *est chercheur, conseiller scientifique et professeur à l'université d'Ottawa.*

Média de programmation.
***Casey Reas**, in «Code - The Language of our Time», Ars Electronica, 2003.*

La conception de logiciel est un facteur déterminant de notre culture moderne et devient de plus en plus un fondement de notre réalité.
Les citoyens du monde se ressemblent, passant leurs vies face aux surfaces réfléchissantes de leurs mobilophones ou de leurs ordinateurs de bureau.
Leurs esprits et leurs mains opèrent dans un espace contenu entre la réalité et les règles arbitraires des menus, cliquant et étirant des fenêtres.
Les artistes utilisent le logiciel pour commenter nos structures sociales et politiques de plus en plus numériques, et pour défier les acceptations fondamentales du code machine. Indépendamment du contenu ou de l'intention de leur travail, les artistes contemporains expriment leurs idées par l'intermédiaire du logiciel. Dans le bouillonnement continuel des différentes disciplines des arts électroniques (cybernétique, réalité virtuelle, CAVE [1], vie artificielle, Net-art, réalité augmentée), le logiciel fournit les fondations sur laquelle la signification et le contenu sont construits.
Avec la revitalisation du concept de «software art» dans des festivals tels que *Transmediale* et *READ_ME*, une discussion critique émerge autour du rôle du logiciel dans notre pratique en matière de culture et d'art.
Ce texte en est une extension et se concentre sur le concept de logiciel comme moyen capable d'expressions inédites, et de langages de programmation comme matériaux doués de propriétés spécifiques.

Qu'est-ce qu'un logiciel ?

Un logiciel est écrit grâce à des langages de programmation, des séquences de caractères alphanumériques et des symboles composés selon des règles syntaxiques rigides [2]. Si vous n'êtes pas un familier des programmes d'ordinateurs, en voici pour référence quelques fragments :

Perl

```
        opendir(DIR, $dir) || die $!;
        @files = readdir(DIR);
        closedir(DIR);
        foreach $file (@files) {
        if($file =~ «.xml») {
                handle(«$dir/$file»);
        }
}
```

[1] NdT: La technologie CAVE est un environnement immersif de réalité virtuelle en 3D.

[2] Il existe des exceptions de langages de programmation appelés «langages de programmation visuels» qui permettent à des structures d'être définies à l'aide de symboles graphiques.

C++

```
                main() {
        int c;
        c = getchar();
        while(c != EOF) {
        putchar(c);
                c = getchar();
        }
}
```

LISP

```
(define (square x)
        (* x x))
(define (sum-of-squares x y)
        (+ (square x) (square y)))
```

Par l'écriture de logiciel, les programmeurs décrivent les structures qui définissent des «processus». Ces structures sont traduites en un code qui est exécuté par une machine et les processus sont interprétés, impliquant activement les composants électroniques de l'ordinateur.
Harold Abelson, informaticien au MIT (Massachusetts Institute of Technology) explique : «Des processus manoeuvrent ces choses abstraites appelées «données». L'évolution d'un processus est dirigée par une succession de règles que l'on appelle un «programme». Les gens créent des programmes pour diriger des processus».
C'est ce processus actif de lecture, de manipulation, et de stockage de données qui offre au logiciel ses capacités uniques.

Le logiciel est un moyen.

Le logiciel a permis une manière de construire un pont entre l'art du passé et des arts électroniques du présent et futur. Comme le souligne Roy Ascott, nous avons établi une transition entre «contenu, objet, perspective, et représentation» et «contexte, processus, immersion, et négociation». L'aspect le plus singulier du logiciel comme moyen est qu'il permet une réponse. Un tel objet réactif a la capacité d'interagir avec son environnement. Myron Krueger, pionnier de la réalité artificielle, suggère un certain nombre de métaphores intéressantes concernant les interactions entre une personne et un logiciel : dialogue, amplification, écosystème, instrument, jeu, et récit. Je suis pour ma part intéressé au développement d'expressions logicielles plus fondamentales que celles évoquées par Ascott et Krueger. Ces expressions sont à la base du logiciel comme moyen et incluent la

forme dynamique, le geste, le comportement, la simulation, l'auto-organisation, et l'adaptation.

Chaque langage est unique.

Comme il y a beaucoup de différentes langages humaines, il existe beaucoup de différents langages de programmation.

De la même manière que différents concepts peuvent être véhiculés par des langues humaines diverses, différents langages de programmation permettent à des programmeurs d'écrire diverses structures de logiciel.

Comme certaines expressions sont intraduisibles d'une langue humaine à l'autre, des structures de programmation ne peuvent souvent pas être traduites d'un langage machine à un autre. Certains langages de programmation ont été établis spécifiquement pour des applications commerciales (COBOL), d'autres pour l'exploration de l'intelligence artificielle (LISP), ou encore la manipulation de données (Perl), et beaucoup de structures écrites dans ces différents langages ne peuvent être exprimées comme telles par d'autres langages. L'animateur abstrait et programmeur Larry Cuba décrit ainsi son expérience : «Chacun de mes films a été fait sur un système différent en utilisant un langage de programmation différent. Un langage de programmation vous donne la puissance d'exprimer certaines idées, tout en limitant vos capacités à en exprimer d'autres».

Les langages de programmation sont des matériaux.

Il peut être utile d'envisager chaque langage de programmation comme un matériau possédant ses moyens et ses contraintes propres. Les différentes langues sont appropriées selon le contexte. Certains langages sont faciles à employer mais obscurcissent le potentiel de l'ordinateur. D'autres sont très compliqués, mais permettent un contrôle total en offrant un accès complet à la machine. Certains langages de programmation sont flexibles et d'autres sont rigides. Les langages flexibles comme Perl et Lingo sont bons pour créer rapidement des programmes courts, mais ils deviennent souvent difficiles à entretenir et à comprendre quand les programmes deviennent longs.

La programmation à l'aide de langages rigides tels que le Assembly 68008 ou le C exige un soin extrême et une attention pénible jusqu'au moindre détail, mais les résultats sont efficaces et robustes. Autant les bois de sapin et de chêne ont une apparence et une utilisation différente, autant les programmes logiciel écrits en différents langages ont des formes esthétiques distinctes. Par exemple, des programmes semblables écrits en Java ou en Flash présentent de singulières différences qui n'échappent pas aux habitués des deux.

La programmation est exclusive.
Beaucoup de gens perçoivent les programmeurs comme des personnes d'un genre unique, différent de tous les autres.
Une des raisons pour lesquelles la programmation reste coincée dans les limites de ce type de personnalité est que les langages de programmation sont généralement créés par des gens aux esprits semblables.
Il est cependant possible de créer différents genres de langages de programmation impliquant des personnes dont l'esprit est visuel et spatial. Certains langages alternatifs ouvrent l'espace de programmation à des gens qui pensent différemment. LOGO était un de ces langages alternatifs précoces, conçu vers la fin des années 60 par Seymour Papert comme concept de langage pour enfants. Grâce à LOGO, les enfants peuvent programmer beaucoup de différents médias comprenant une tortue-robot et des images graphiques à l'écran. Un exemple contemporain est l'environnement de programmation MAX développé à l'IRCAM par Miller Puckette dans les années 80. Max a généré un enthousiasme auprès de milliers d'artistes qui l'emploient pour créer des applications audiovisuelles et des installations.
Les interfaces graphiques (GUIs - Graphic User Interfaces) ont rendus possible l'utilisation d'un ordinateur à des millions de personnes ; les environnements alternatifs de programmation contribueront à engendrer de nouvelles générations d'artistes créant des logiciels.

Expressions logicielles.
Quand des programmes sont exécutés par une machine, ils sont des processus dynamiques et non des textes statiques à l'écran.
Des noyaux d'expressions numériques incluant forme dynamique, geste, comportement, simulation, auto-organisation, et adaptation émergent de ces processus. De telles expressions, ou d'autres plus basiques, sont les fondements sur lesquels sont élaborées des idées ou des expériences plus complexes. Chacune de ces expressions est décrite ci-dessous et illustrée par un exemple provenant du «Aesthetics & Computation Group» du MIT (Massachusetts Institute of Technology). Ces exemples ont été créés par d'hybrides artistes/programmeurs entre 1998 et 2001 et fournissent des démonstrations claires de ces expressions logicielles.

Forme dynamique.
Une forme dynamique est une forme qui change au bon moment. Si cette forme réagit aux stimuli, elle est réactive. «Scratch» de Jared Schiffman (figure 1) démontre les qualités de base d'une forme dynamique. Dans ce logiciel, la position d'un cercle contrôlable affecte sans interruption la découpe de chaque élément visuel. «Scratch» augmente la communication

visuelle de la forme en ajoutant des couches de mouvement de manière réactive et fluide. En général, la forme peut réagir à n'importe quel signal de son environnement, incluant les dispositifs d'entrée de données habituels comme la souris, un micro ou une camera video, jusqu'à des interfaces plus surprenants comme un capteur de radiations ou encore un sonar.

Figure 1: Jared Schiffman : «Scratch».

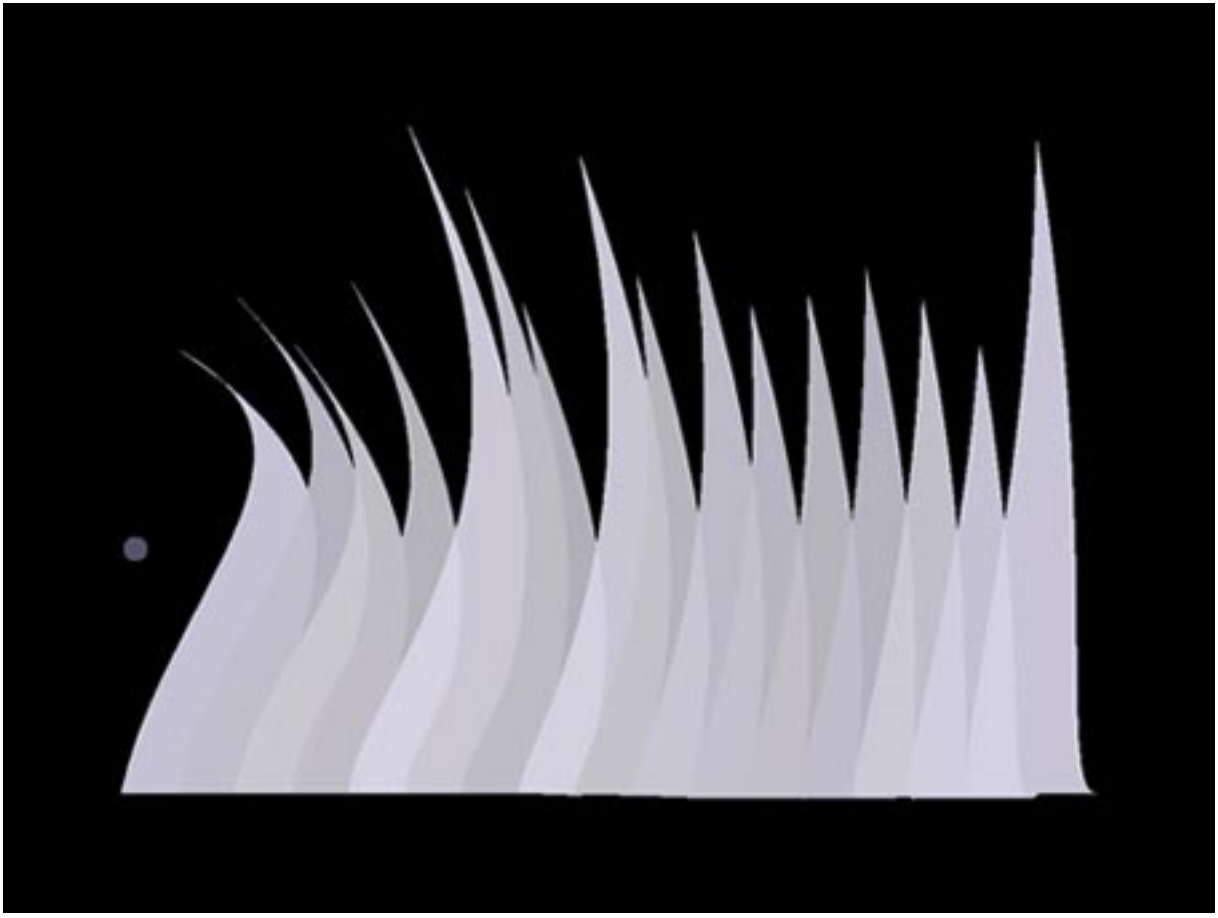
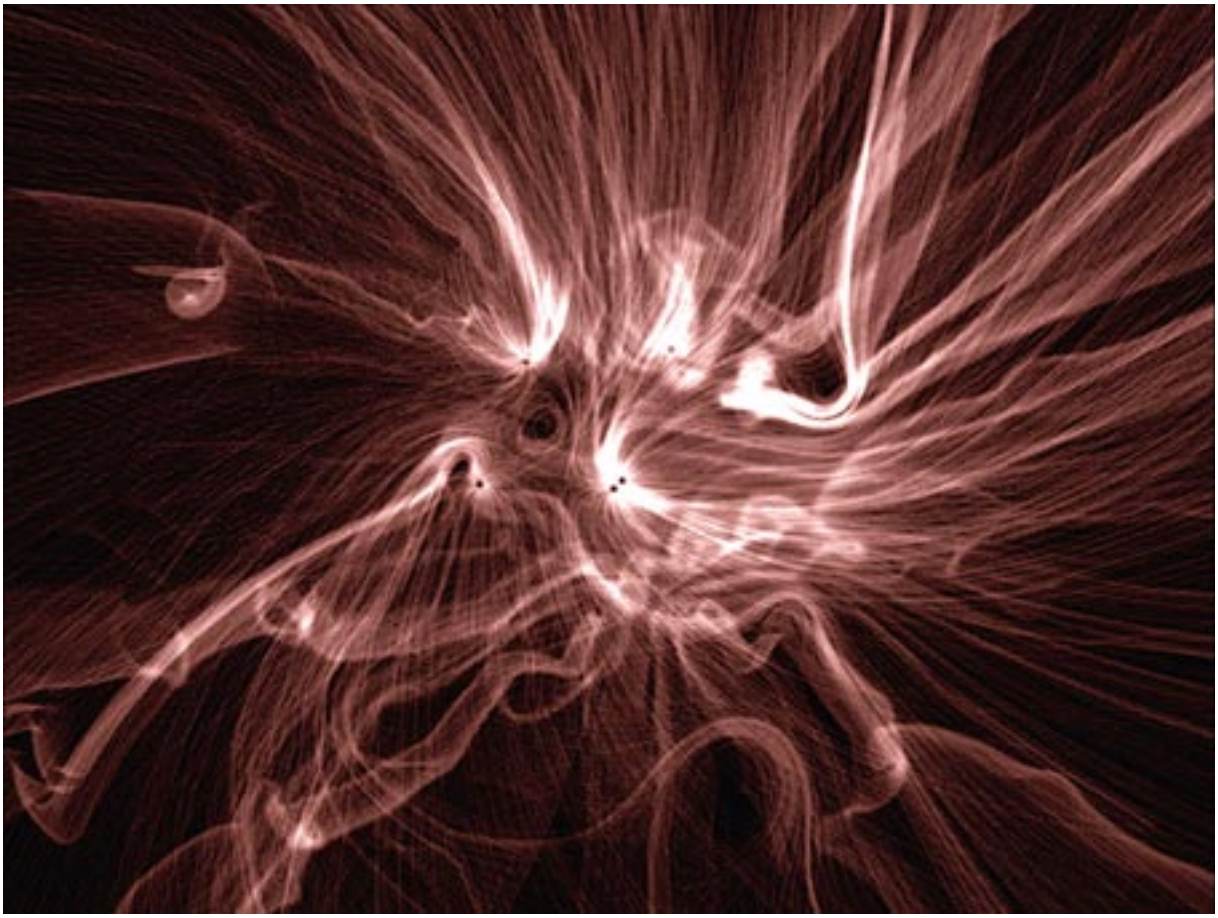


Figure 2: Golan Levin: «AVES (AudioVisual Environment Suite)».



Le geste.

Le geste est un composant important de tout medium continu et le logiciel possède la capacité de transcrire et d'interpréter un geste.

«Aves» de Golan Levin (figure 2) est un ensemble d'applications qui amplifient les gestes de la main en traduisant leurs données en son et en image. Une application transpose la structure de chaque geste en sons qui reflètent leur degré de courbure. Une autre empile les gestes pour créer des textures sonores graduelles qui s'activent et varient selon la présence du curseur.

L'interprétation des gestes est alors plus complexe, mais permet de nouvelles occasions pour provoquer l'interaction. La reconnaissance de l'écriture manuelle est une application d'interprétation de geste. Certaines installations ou jeux-video utilisent une forme plus basique d'identification de geste pour permettre de contrôler l'action avec des mouvements complexes.

Le comportement.

Le comportement est un mouvement possédant l'apparence d'une intention. La combinaison de comportements simples peut suggérer une personnalité ou une humeur. Le comportement peut être créé en écrivant intuitivement des programmes ou en mettant en application des modèles biologiques.

Dans le projet «Trundle» (figure 3), l'objet physique est géré par un programme qui détermine comment il devra se déplacer en présence de stimuli dans son environnement. «Trundle» scrute l'environnement à la recherche de personnes, mais quand il trouve quelqu'un, il tente de s'échapper. «Trundle» est curieux et timide. Une rangée de capteurs sur le corps de «Trundle» surveille continuellement son environnement immédiat et envoient des signaux au microcontrôleur qui détermine comment les moteurs devraient tourner. En général, le comportement peut être employé pour impliquer activement l'esprit par la personnification d'objets, le développement de caractères, la communication d'une humeur, et ajouter une épaisseur psychologique à une oeuvre logicielle.

La simulation.

La simulation du monde physique fournit un point d'accès facile à la perception des travaux logiciels. Nos sens ont évolué pour répondre aux règles de la nature. Un des premiers jeux d'ordinateur, Pong, était une simulation hautement abstraite du tennis. La technologie moderne et les communautés scientifiques utilisent des modèles inspirés de la réalité comme base de conception d'objets physiques et d'orientation de recherche.

«Floccus» de Golan Levin (figure 4) crée un groupe de lignes élégantes, connectées chacune à une liste de ressorts simulés. Le mouvement ondulant créé par cette simulation simple engendre une crispation chez les spectateurs quand elle est combinée avec la réponse. Les lignes s'étendent et se contractent selon la force, la masse, et l'accélération. Dans un logiciel, la simulation peut dépasser une simple imitation de la perspective, des matériaux, et des lois de la physique - les processus inhérents aux phénomènes naturels peuvent être aussi bien simulés.

Figure 3: Casey Reas: «Trundle».

Figure 4: Golan Levin: «Floccus».

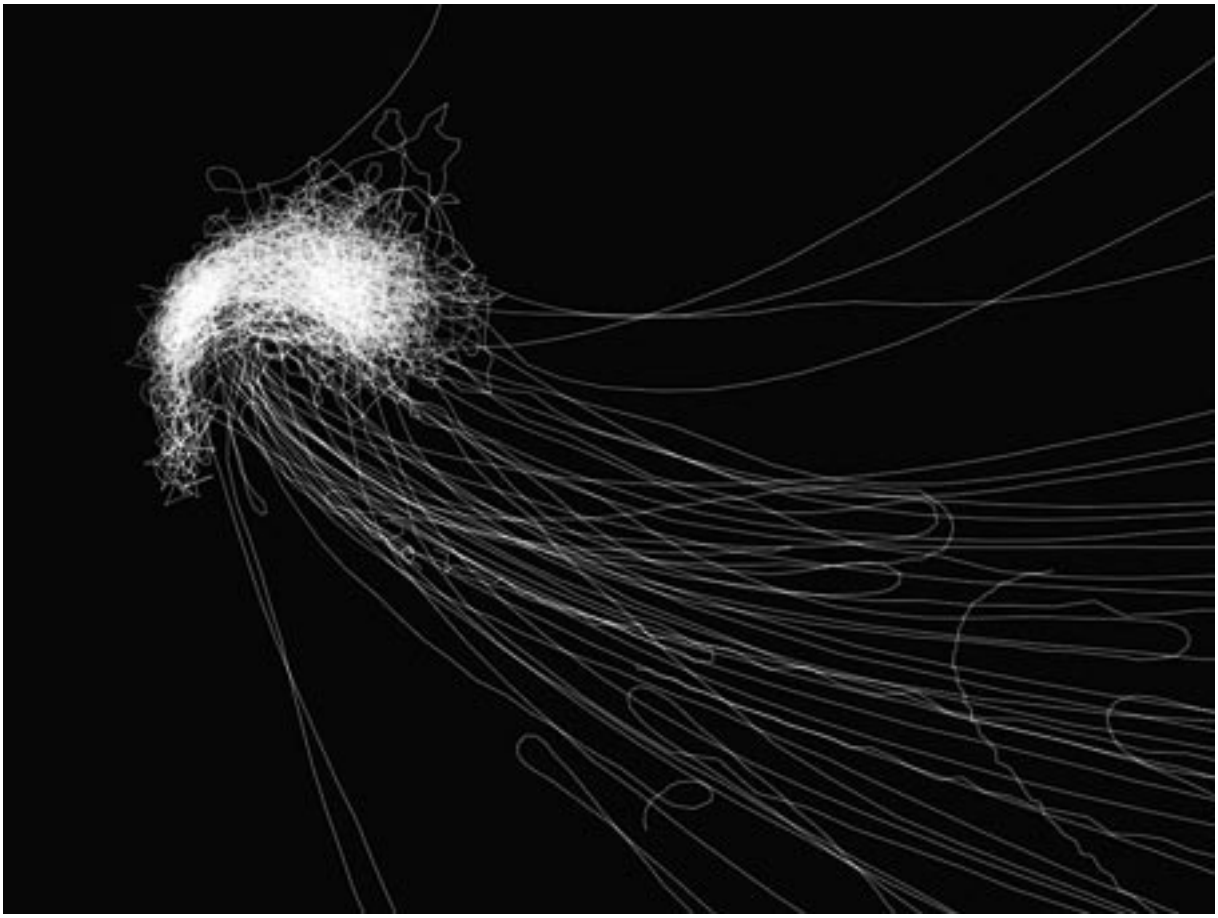
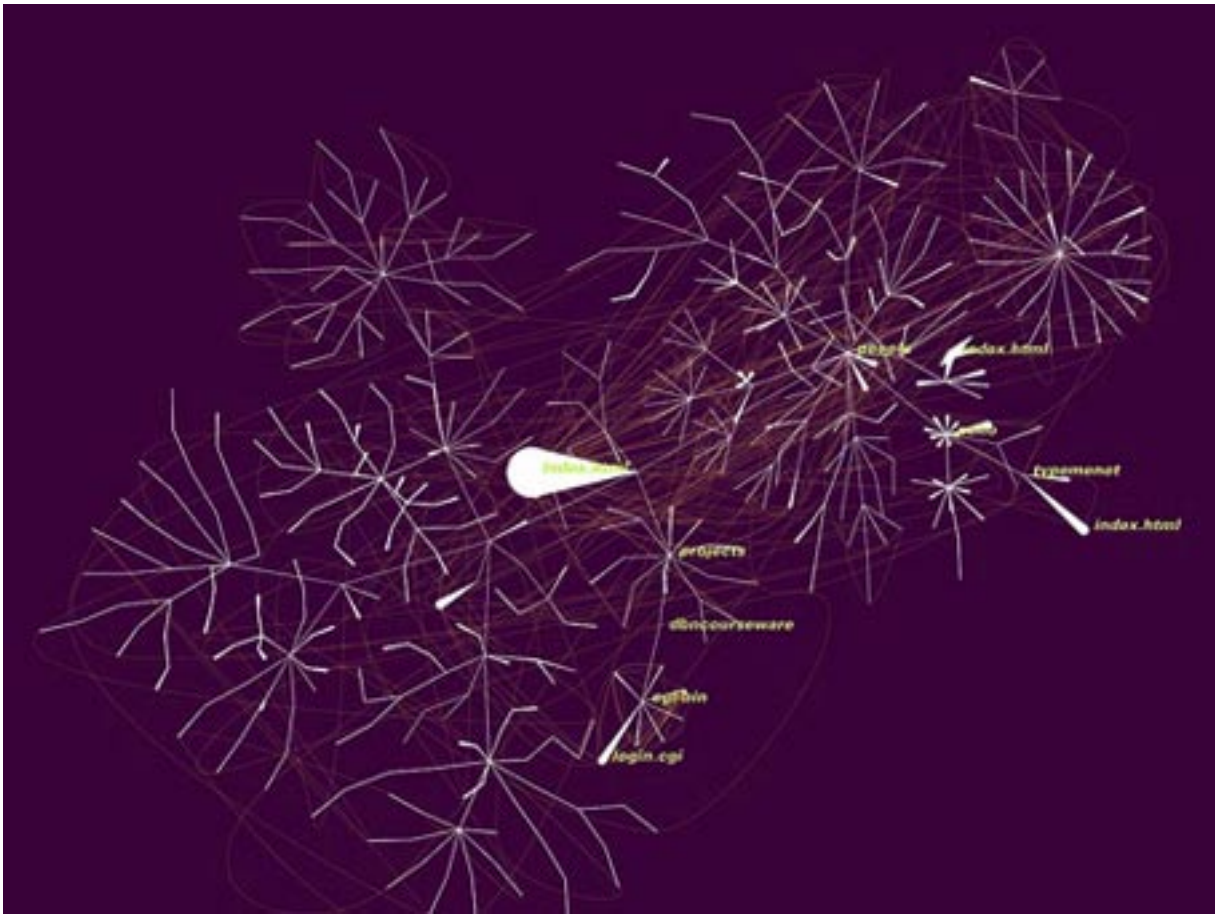


Figure 5: Ben Fry: «Valence».

Figure 6: Ben Fry: «Anemone».



Auto-Organisation.
La capacité des éléments à s'auto-organiser rend possibles les phénomènes d'émergence. La structure émerge des interactions entre de nombreux processus autonomes. «Valence» de Ben Fry (figure 5), lit le texte d'un livre mot après mot et l'organise dans l'espace selon un système de règles.
Un volume complexe émerge des relations entre divers mots dans le texte. De petites entorses aux règles de l'interaction peuvent avoir des effets potentiellement grands sur le traitement de la visualisation.

Adaptation.
L'adaptation est la capacité à changer.
Pour que le logiciel s'adapte, il doit posséder une représentation de lui-même et se rendre compte de son contexte. «Anemone» de Ben Fry (figure 6) peut surveiller sa densité et modifier sa structure pour garder l'équilibre.
«Anemone» est la visualisation d'un trafic sur un site web et, au fur et à mesure que les heures et les jours passent, le logiciel enlève des sections de sa masse pour tenir compte du développement de nouvelles sections sans empêcher la lisibilité de l'information. Écrire un logiciel qui s'adapte vraiment à son contexte est un vrai défi et les résultats pertients sont rares. En utilisant un interprète, il est possible pour un programme de s'auto-modifier en cours d'exécution.

Programmer.
Bien que les logiciels soit constamment utilisés dans les arts électroniques, des individus choisissent de construire leur code de manière radicalement différentes, allant de l'écriture dans des langages de bas niveau jusqu'à des collaborations avec des programmeurs. Certains artistes utilisent le code comme outil pour créer des oeuvres dans d'autres médias. Ils emploient des applications disponibles dans le commerce pour produire des impressions, des videos ou pour faire des croquis approximatifs qui sont exécutés dans des médias analogiques. D'autres collaborent avec des programmeurs professionnels, créant des caractéristiques qui sont ensuite mises en application par les programmeurs. Beaucoup d'autres créateurs utilisent des environnements de programmation développés pour des designers et des artistes. Ils travaillent avec des environnements de programmation de scripts et/ou visuels tels que Director, Flash, et Max qui facilitent la construction de logiciels pour les non-programmeurs. Un plus petit groupe d'artistes travaillant le logiciel utilise des langages de programmation développés pour des programmeurs professionnels. Ils utilisent en général des langages comme C, Java, et Peri et développent souvent leurs propres outils-maison pour travailler dans ces environnements. Il n'y a pas UNE

seule manière correcte de travailler avec du logiciel.
C'est un choix personnel, une mélange équilibré de contrôle et de simplicité. La maîtrise de la programmation demande beaucoup d'années de travail intensif, mais la compréhension des principes de base est à la portée de chacun. À mon avis, chaque artiste utilisant du code devrait y être instruit.
Que signifie l'instruction dans le contexte du logiciel ?
Alan Kay, innovateur en matière de réflexion sur le code comme matériau, écrit ceci: «La capacité à «lire» un medium signifie l'accès aux matériaux ET aux outils créés par d'autres. Vous devez posséder les deux pour être instruits. Dans l'écriture imprimée, les outils que vous produisez sont rhétoriques; ils démontrent et convainquent. Dans l'écriture pour ordinateur, les outils que vous produisez sont des processus; ils simulent et décident. Ces processus qui simulent et décident sont l'essence du logiciel et ils peuvent seulement être entièrement compris en les construisant. Les artistes capitalisent en écrivant leur propre logiciel. Avec la croissance du web, la popularité d'environnements de scripts comme Flash, et la chute des prix du matériel, beaucoup plus d'artistes explorent la programmation.Le champ de la programmation audio-visuelle est un excellent exemple de cette tendance. De petites compagnies de logiciel comme Cycle '74, développeur de Jitter, sont très sensibles à leur communauté d'artistes et stimulent le développement d'outils accessibles. Jitter est une bibliothèque sophistiquée de structures visuelles pour l'intégration de l'image avec le son. Beaucoup d'artistes ont été bien au-delà de compter sur les développeurs pour leurs outils. Les Pink Twins, un duo de musiciens/programmeurs de Helsinki, ont créé Framestein, un logiciel de calcul video lié à PD (Pure Data), logiciel open-source de traitement en temps réel. Le collectif d'artistes allemand Meso est même allé encore plus loin avec VVVV, une bibliothèque ambitieuse d'outils de synthèse visuelle en temps réel. Certains artistes développent leurs propres outils logiciels, et après une période d'amélioration, choisissent de le faire partager à la communauté.

Synthèse.
Au cours des trente dernières années, des artistes ont développé des pratiques innovantes à l'aide de moyens logiciels, mais ils n'ont seulement exploré qu'une petite gamme des possibilités conceptuelles. Historiquement, les langages et les environnements de programmation ont encouragé une méthodologie spécifique, qui n'a cependant pas engagé la majorité des artistes créant des oeuvres interactives ou logicielles. De nouveaux outils émergent et encouragent les artistes à commencer à travailler directement avec le médium logiciel. La prolifération de l'instruction au logiciel parmi les artistes augmentera une meilleure utilisation du logiciel et contribuera à de nouvelles formes de logiciels et d'environnements de développement. Ces matériaux et environnements ont le potentiel d'étendre la création de logiciel à une communauté créative et critique encore plus vaste.

Casey Reas,
«Programming media», in «Code - The Language of our Time», Ars Electronica, 2003.
(Traduit de l'anglais par Marc Wathieu).

- *http://www.aec.at/en/archiv_files/20031/FE_2003_reas_en.pdf*

Bibliographie :

Abelson, Harold, Gerald Sussman, and Julie Sussman :
«Structure and Interpretation of Computer Programs».
MIT Press, Cambridge, MA. 1985

Ascott, Roy :
«Moist Ontology».
Published in The Art of Programming.
Sonics Acts Press, Amsterdam. 2002

Cuba, Larry :
«Calculated Movements».
Published in Prix Ars EIectronica Edition '87: Meisterwerke der Computerkunst.
Verlag H.S. Sauer. 1987

Kay, Alan :
«User Interface: A Personal View».
In «The Art of Human-Computer Interface Design»,
edited by Brenda Laurel Addison-Wesley Publishing Co, Reading MA. 1989.

Krueger, Myron : «Responsive Environments».
Published in «Multimedia, From Wagner to Virtual Reality».
Edited by Randall Packer and Ken Jordan. W.W Norton & Co, Inc., New York. 2001.

Casey Reas *est artiste, co-créateur (avec Ben Fry) du logiciel Processing et professeur à UCLA (Los Angeles). Il est représenté par la galerie Bitforms à New-York.*

Entrevue avec Golan Levin.

*Entrevue réalisée par **Carlo Zanni**, extraite du Magazine électronique du CIAC (Centre International d'Art Contemporain) de Montréal, n°19, 2004.*

Carlo Zanni : Aléatoire, génératif, œuvres d'art générées à partir d'un logiciel (« software art »), caractère aléatoire (« randomness ») : souvent les gens utilisent ces termes pour décrire une variété très large d'œuvres numériques... Puisque vous êtes un artiste mais également un diplômé du MIT, il me semble que vous êtes la personne la mieux placée pour clarifier ces concepts pour nous.

Golan Levin : Merci. En fait, je suis maintenant en poste à la Carnegie Mellon University, mais sept ans passés au MIT laissent leur empreinte.

Proposer des définitions est toujours risqué - je suis certain que j'offenserai quelques personnes, mais j'essayerai quand même. Les œuvres d'art générées à partir d'un logiciel (« software art ») représentent une nouvelle forme de pratique créatrice où le médium artistique est le code informatique. Habituellement (mais pas toujours), le code lui-même n'est pas la partie de l'œuvre qui est présentée au public ; au lieu de cela, l'artiste compile le code dans un programme exécutable, comme pour tout autre logiciel. Les œuvres d'art générées à partir d'un logiciel sont accessibles au public de la même manière que les logiciels courants - soit par téléchargement, soit par l'achat d'un disque compact-ROM. Les intentions qui président à la création de telles œuvres sont diverses : parfois le but est simplement de produire un bel effet visuel, toujours changeant. J'ai moi-même fait certaines œuvres de ce type. Mais une grande partie de ces œuvres ont été créées pour remettre en question les concepts et les conventions du logiciel lui-même - certaines, comme Auto-Illustrator d'Adrian Ward, une parodie d'Adobe Illustrator, est même un outil utile en soi. Enfin, certaines de ces œuvres parmi les meilleures ne sont mêmes pas visuelles, elles peuvent même prendre la forme d'un virus, comme le virus Biennale.py créé par le collectif 0101.org. Florian Cramer et Andreas Broeckmann ont écrit des textes intéressants à ce sujet.

Les «œuvres d'art générées à partir d'un logiciel» («generative software art»), comme on les comprend habituellement aujourd'hui, sont le produit d'une pratique artistique qui utilise des algorithmes mathématiques pour produire automatiquement ou semi-automatiquement des expressions dans des formes artistiques plus conventionnelles. Par exemple, un programme pourrait produire des poèmes, des images, des mélodies, ou des animations. Habituellement, l'objectif d'un tel programme est de produire des résultats différents à chaque fois qu'il est exécuté. Et généralement, on espère que

ces résultats auront une valeur esthétique, et qu'ils seront distincts les uns des autres, d'une manière qui les rendra intéressants. Un certain art génératif opère de façon complètement autonome, alors que quelques œuvres génératives incorporent également les inputs d'un utilisateur, ou encore de l'environnement.

Le caractère aléatoire en tant que tel («randomness») est très difficile à définir, et en dépit de mon diplôme du MIT, je suis à peine qualifié pour le faire, étant donné que la «vraie» définition implique certaines notions de mathématiques extrêmement avancées. Une bonne définition générale est que, dans une séquence de bits aléatoires, chaque élément ne peut pas être prédit eu égard à ses voisins ou suivant une certaine règle de base. Une séquence de ce type résiste à la compression ou à la description compacte. Par exemple, le modèle «01010101010101010101010101010101» peut être décrit de manière très compacte en tant que «16 blocs de '01'», mais la séquence «01100101001001000001010011001110», qui a le même nombre d'éléments, ne peut pas être exprimé aussi simplement, et est donc considérée comme plus « aléatoire ».

Les œuvres d'art aléatoire («aleatoric artwork») incorporent ce caractère aléatoire dans leur exécution ou leur composition. Les artistes ont utilisé l'aléatoire comme outil artistique bien avant l'avènement des ordinateurs ! Toutes sortes de techniques ont été employées comme adjuvant dans la création de poèmes, de récits, de musique, et de dessins, et ce, pour de nombreux motifs. Quelques artistes, comme les Dadaïstes, entendaient se rebeller contre les restrictions des règles académiques, alors que d'autres, comme les Surréalistes, ou quelques artistes Zen, ont souhaité s'affranchir de la censure de la conscience.

Je ne suis pas sûr de savoir pourquoi les artistes numériques d'aujourd'hui sont aussi attirés par l'aléatoire. Probablement pour beaucoup de raisons différentes ; pour ma part, je trouve qu'une telle approche peut certes se révéler intéressante de temps en temps, mais je demeure un peu sceptique envers les artistes qui manquent de distance critique pour évaluer les résultats de telles applications. Au mois de mai dernier, j'ai assisté à une conférence intitulée User_Mode à la Tate Gallery de Londres, à l'occasion de laquelle un artiste du Web bien connu a fait une communication. Il y a présenté une œuvre d'art générée à l'aide de Flash dans laquelle, à chaque fois qu'il cliquait sur la souris, il obtenait un autre arrangement aléatoire d'images de fleurs. Il semblait littéralement fasciné par la richesse générative de son œuvre et il a continué de cliquer sur sa souris encore et encore, tout en s'exclamant : «Un autre ! Magnifique ! Encore un ! Superbe ! Et celui-ci,

étonnant ! Ah, merveilleux ! Miraculeux ! Vrai, je pourrais continuer comme cela toute la journée !». J'ai été écoeuré. Ses collages de fleurs étaient bons, mais ils étaient tous également bons - et il n'a pas vu que c'est justement ce qui les rendait tous également mauvais, tout aussi bien. C'est une chose d'être capable d'apprécier la beauté surgissant de résultats inattendus, mais nous devons aussi réaliser que nos algorithmes sont également capables de froideur et de laideur, ou nous n'apprendrons jamais rien.

Carlo Zanni : Pouvez-vous s'il vous plaît me parler davantage de vos projets Floccugraph et Yearbook Pictures ?

Golan Levin : Les deux projets sont des traitements aléatoires de documents photographiques. Dans les deux cas, la photographie d'un visage humain sert de point de départ, et ce visage est traité comme champ de probabilité pour des opérations aléatoires. Plus spécifiquement, dans Cellular Portraits, des points aléatoires sont dispersés à travers la surface de l'image, selon un champ probabiliste de densité régi par l'obscurité de la photographie du visage. En d'autres termes, plus une partie du visage est foncée, plus il y aura de chance que des points aléatoires s'y retrouvent. Ces points sont ensuite employés pour produire un effet de papier-bulle comme les diagrammes de Voronoi que vous apercevez. Dans Floccugraph, l'obscurité du visage est employée comme champ de probabilité pour produire des forces aléatoires d'attraction ou de répulsion sur une série de filaments semblables à des cheveux. Les deux processus produisent toujours des résultats différents, chaque fois qu'ils sont exécutés.

Carlo Zanni : Tandis que pour JJ, et Yearbook Pictures, vous semblez être plus en contrôle (c'est vous qui avez choisi les différents visages pour les deux œuvres), Alphabet Synthesis Machine (un logiciel géré en réseau permettant à l'utilisateur de générer une police pour les caractères d'un alphabet à partir d'une «graine» dessinée à la main) laisse apparemment aux utilisateurs plus de liberté pour adapter leur propre police (mais ils demeurent influencés par vos algorithmes). Pouvez-vous s'il vous plaît nous parler de votre processus de création ? Pouvez-vous aussi, brièvement, nous exposer une partie des algorithmes génératifs en jeu dans le logiciel ?

Golan Levin : Je pense que mes œuvres varient considérablement, dépendant du degré de contrôle que je peux choisir d'exercer moi-même sur leur apparence, ou au contraire que celle-ci résulte de la mise en œuvre d'algorithmes autonomes, ou de l'interaction d'un autre utilisateur. Les trois projets que vous mentionnez s'avèrent justement des exemples des trois extrêmes de cette gamme. Mais c'est une erreur de supposer que mon

contrôle, en tant qu'auteur, commence et finit avec la manière dont j'aurai prédéterminé la manifestation précise des résultats visuels de mon œuvre. Cette idée que l'on puisse choisir de déléguer ou au contraire de conserver le contrôle de son œuvre est en fait un leurre - car je peux ou peux ne pas avoir prédéterminé le résultat visuel, mais c'est quand même moi qui ait choisi et programmé les algorithmes qui font le lien entre l'input et l'output ! Le contrôle a été supplanté par le méta-contrôle.

En d'autres termes, une grande partie de la magie dans la pratique de l'art interactif ou génératif tient à la création d'une illusion de contrôle : celle qui fait croire que l'artiste a renoncé au titre d'auteur en faveur de l'utilisateur, ou d'un certain algorithme intelligent. En fait, c'est un mythe. Dans un sens, les artistes qui travaillent en art généré à partir d'un logiciel ou d'un ordinateur sont plus en contrôle que les artistes en général l'ont jamais été. Au lieu d'une écriture qui soit le fait d'une expression individuelle (la leur), ils écrivent maintenant des systèmes à exprimer. Mais avec ce pouvoir vient un plus grand fardeau critique. Les critères esthétiques traditionnels ne sont pas suffisants pour évaluer les œuvres résultantes. Le système lui-même doit en plus être évalué, suivant la gamme et la plasticité plus ou moins grandes de ses expressions réelles et possibles.

Alphabet Synthesis Machine est un système interactif qui permet aux utilisateurs de « faire évoluer » leur propre alphabet personnel. Je trouve qu'il est important de mentionner que les alphabets qui en résultent n'ont pas de valeur signifiante, c'est-à-dire qu'ils évoquent idéalement des alphabets de civilisations inconnues, et ils ne sont probablement pas lisibles d'aucune manière usuelle. Ces alphabets résultent d'un processus au cours duquel l'utilisateur fournit d'abord une «graine» dessinée à la main, comme vous l'avez mentionné, et guide ensuite l'évolution d'un algorithme génétique qui produit des lettres. C'est très proche de l'idée de Darwin de la «survie du plus apte» («survival of the fittess») : il existe au point de départ une population d'environ mille lettres, dont chacune est une petite simulation physique d'une marque griffonnée à la main. Ces griffonnages se reproduisent continuellement, en mutant et en permutant leur matériel génétique. Mais seulement les «plus aptes» survivent, et c'est l'utilisateur qui en stipule la forme physique avec mon interface graphique. Après un nombre suffisant d'itérations, l'alphabet apparaîtra si tout va bien comme tout à fait unique.

En ce qui concerne vos questions au sujet de l'aspect aléatoire dans les œuvres d'art générées à partir d'un logiciel, Alphabet Synthesis Machine est la combinaison d'un processus aléatoire et d'un processus guidé par

l'utilisateur : il y a un aspect aléatoire dans les mutations qui affectent la population, et dans leur processus sexuel de reproduction, mais en même temps, le système entier est (lentement) formé selon les buts de l'utilisateur. Cela ressemble beaucoup à la création d'une nouvelle race de chien : cela prend beaucoup de générations, et de la chance. Il est intéressant de mentionner que plusieurs autres personnes ont mis sur pied des projets semblables avec des techniques très différentes ; par exemple, Alphabet Soup de Matt Chisholm est un projet qui emploie un ensemble de composantes pré-dessinées, qui se relient ensemble pour créer des nouvelles formes de lettres. Son projet apparaît comme plus typographique, alors qu'Alphabet Synthesis Machine apparaît comme plus manuscrit, en raison des simulations physiques derrière les marques.

Carlo Zanni : Comment contrôlez-vous personnellement le caractère aléatoire ? J'aimerais vous entendre parler de votre «style» dans l'emploi de ce «matériau» peu commun.

Golan Levin : De tous les pièges où risque de tomber un concepteur de systèmes interactifs, je pense que l'utilisation de l'aléatoire généré par ordinateur est un des plus difficiles à éviter. L'attrait de l'aléatoire, en théorie, est qu'il peut être employé pour introduire de la nouvelle «information» afin de maintenir la fraîcheur dans l'interaction. Malheureusement, le problème avec l'aléatoire est qu'il ne contient aucune information du tout - et qu'il peut donc être mal interprété par un utilisateur sans méfiance, qui peut faire l'erreur de le prendre pour de l'information réelle. L'utilisateur, face à un système aléatoire, peut se demander : «Est-ce que je viens de faire X, ou cela s'est-il produit tout seul ?» - «Est-moi qui contrôle tel comportement, ou est-il l'effet d'une coïncidence ?» Dans de tels cas, l'aléatoire constitue une distraction génératrice de confusion qui rend plus difficile pour un utilisateur la compréhension et la maîtrise d'un système interactif. Cet état de choses peut se révéler plus particulièrement problématique pour les systèmes dont l'attrait premier est la promesse d'une relation cybernétique étroite entre l'homme et la machine. J'ai constaté que, dans plusieurs cas, le geste humain lui-même élimine le besoin du recours à l'aléatoire. Le geste humain, en particulier, est déjà une source si riche d'inputs, avec ses propres propriétés stochastiques et irrégulières, qu'un élément aléatoire additionnel n'est pratiquement jamais nécessaire ! Le recours à l'aléatoire peut également être évité par un examen plus profond du geste : l'utilisation des techniques d'analyse de signal, par exemple, peut permettre d'explorer les dimensions expressives additionnelles latentes dans les gestes de l'utilisateur, telles que leur périodicité, ou leur spectre de fréquence particulier.

À une époque, j'évitais presque religieusement de recourir à l'aléatoire dans mes œuvres, et j'en étais fier. Mais il s'est produit un tournant, un jouralors que j'étais étudiant au *Media Lab*. J'avais été invité à faire une démons-tration de mon logiciel devant le premier ministre de l'Autriche. C'était un homme énorme et très grand, avec une solide poignée de main et jouissant d'une indéniable autorité. Comme je lui faisais la démonstration de mon logiciel, j'ai insisté sur le fait qu'il n'y avait aucun caractère aléatoire dans les algorithmes : tout ce qu'il voyait était entièrement en fonction directe des mouvements qu'il imprimait à la souris avec sa main. Il s'est tourné vers moi et m'a demandé : « mais qu'y a-t-il de mal dans l'aléatoire ? La vie elle-même est aléatoire. » J'ai pensé que c'était d'une grande sagesse. Ainsi, après tout ce que j'ai pu dire, j'admets aujourd'hui de manière un peu embarrassée que j'emploie effectivement l'aléatoire, particulièrement sous la forme de bruits statistiques à haute fréquence et basse amplitude. C'est en fait essentiel pour générer toutes sortes de textures organiques. Sans cela, la plupart des systèmes génératifs sembleraient sans vie, trop réguliers, et ennuyeux.

De nos jours, mon point de vue au sujet de l'aléatoire rejoint celui de John Maeda, qui a écrit dans *Design By Numbers* que «si vous entendez recourir à l'aléatoire, vous devriez au moins en connaître l'origine». Les artistes et les concepteurs en art génératif doivent être conscients que l'aléatoire produit par un ordinateur N'EST PAS de l'aléatoire véritable ! Les générateurs de nombres aléatoires («random number generators») sont en fait des algorithmes totalement prévisibles, et ces jeux mathématiques ne sont jamais aussi parfaitement chaotiques que ce que l'on peut retrouver dans la nature. Les différentes imperfections et régularités des générateurs algorithmiques de nombres aléatoires, d'ailleurs, influent de manière dramatique sur le caractère de toutes les œuvres génératives qui les emploient. Malheureusement il semble que la plupart des artistes qui travaillent dans ce domaine ne s'en rendent pas compte. J'affirme que les artistes qui utilisent des générateurs de nombres aléatoires se doivent à eux-mêmes d'apprendre à connaître les propriétés des différents algorithmes générateurs d'aléatoire. Ils devraient savoir qu'il existe plusieurs types différents de générateurs de nombres aléatoires, tels que ceux de Cauchy, Pareto, Poisson, les invariants continus de Kotz et de Johnson, etc. Chacun de ces algorithmes produit des types complètement distincts de caractères aléatoires !

J'avoue que cela m'agace beaucoup quand les artistes en général se contentent de la fonction aléatoire standard fournie par Flash. C'est comme s'il y avait un rabais sur la peinture rouge au magasin, et qu'à cause de cela tout le monde se mettait à utiliser cette seule couleur pour peindre.

C’est une chose de l’employer parce que c’est bon marché et avantageux, mais nous en sommes arrivés au point où les artistes semblent ignorer totalement qu’il puisse exister d’autres couleurs dans la palette de l’aléatoire.

Entrevue avec Golan Levin *réalisée par Carlo Zanni, extraite du Magazine électronique du CIAC (Centre International d’Art Contemporain) de Montréal, n°19, 2004.*

(Traduit de l’anglais par Anne-Marie Boisvert).

- *http://www.ciac.ca/magazine/archives/no_19/entrevue.htm*

Golan Levin *est artiste et ingénieur, diplômé du MIT Media Laboratory (Aesthetics and Computation Group). Il est actuellement professeur auxiliaire en arts électroniques à Carnegie-Mellon University ; il est représenté par la galerie Bitforms à New-York.*

Carlo Zanni *est artiste.*

Entrevue avec David Rokeby.

*Invité par l'association iMAL (interactive Media Art Laboratory) pour le workshop «Contructing experiences in interactive installations», **David Rokeby** a accordé une interview le 14 décembre 2003 à Xavier Ess (RTBF) pour l'ex-émission CyberCafé21.*

Xavier Ess : David, est-ce que vous pouvez d'abord nous expliquer ce qu'est le soft que vous avez inventé, softVNS ?

David Rokeby : SoftVNS est une boite à outils logiciel que j'utilise pour traiter la vidéo en temps réel, que ce soit pour créer des installations interactives ou des installations vidéos. C'est très rapide et cela permet de manipuler la vidéo un peu comme une matière comme on fait avec de l'argile ou de la peinture, c'est très fluide et très facile à utiliser.

Xavier Ess : La chose principale dans ce soft c'est qu'on peut déterminer le mouvement et jouer avec le mouvement dans l'image ?

David Rokeby : Depuis presque 1981, j'utilise la vidéo comme un moyen de détecter et suivre les mouvements, de les comprendre, d'observer le monde et de faire que l'ordinateur comprenne quelque chose de ce monde. SoftVNS 2 contient principalement des fonctions pour détecter les têtes des personnes, pour trouver certaines couleurs, pour voir la qualité des mouvements, pour construire des boutons ou des zones virtuelles de déclenchement dans l'espace...

Xavier Ess : Depuis plus de 20 ans vous travaillez dans l'interaction entre l'homme et le corps humain et la machine. Qu'est-ce qui vous fascine dans ce rapport ? Est-ce qu'il y a une communication homme-machine ?

David Rokeby : Il y a 2 types d'interactions quand on parle d'interactions entre les homme et les machines. D'un côté, il y a les interactions entre les humains et les machines. Et de l'autre, il y a les interactions entre humains et humains à travers les machines. Et les deux types m'intéressent... mais d'une certaine façon, celles qui m'intéressent le plus ce sont quand les humains répondent à leur propres réflections, à leurs propres ombres à travers le dispositif interactif... Souvent, nous ne nous reconnaissons pas nous mêmes complètement. Nous avons des comportements étranges avec nos ombres interactives, et par exemple, nous confondons notre intelligence avec l'intelligence du système. Je trouve cette relation très intéressante parce que je pense qu'il est important de commencer à comprendre ce qu'est la nature de nos rapports avec les machines, à la fois leurs aspects positifs, et aussi ceux pas toujours positifs.

Xavier Ess : Cela veut dire qu'une machine qui est quelque chose de très très simple peut arriver à tromper un être humain qui est quelque chose de très très complexe ?

David Rokeby : Oui. C'est parce que l'homme est toujours plus complexe que la machine. Nous sommes bien plus complexes que n'importe quel ordinateur, et donc quand on réalise une boucle de feedback interactive, c'est la complexité de l'homme qui remplit tout le système. Et c'est elle qui vous revient en partie dans la réponse du système. Et plus particulièrement parce que nous ne nous connaissons pas bien. Nous ne sommes pas vraiment conscients de comment notre corps bouge. Nous croyons le connaître, mais nous ne le connaissons pas. Ainsi, mon système «Very Nervous System» qui traduit les mouvements du corps en musique est surprenant parce que nous ne nous rendons pas compte de comment nous bougeons. Nous sommes supris par les réponses, pas parce qu'elles sont inhabituelles, mais parce que nous découvrons que nos mouvements ne correspondent pas à l'idée que nous en avions.

Xavier Ess : Alors il y a aussi dans votre travail cet aspect où la technologie met en avant des sentiments humains, comme par exemple dans «Watch», il y a ce travail avec les gens qui restent sur place, les Homeless. Vous pouvez nous parler de cela ?

David Rokeby : La génèse de «Watch» est assez intéressante. J'ai construit une situation où pour la première fois je voyais ce que mon ordinateur voyait. Comme il voyait principalement les mouvements et que j'étais fatigué de gesticuler devant la caméra pour générer des mouvements, j'ai pointé la caméra par la fenêtre, directement dans la rue. J'habitais alors une rue très fréquentée. Et j'ai obtenu deux processus parallèles: un, d'un côté de l'écran, qui montrait les mouvements dans l'image (dans cette rue, c'était les voitures, les gens qui marchent,...); un autre, de l'autre côté, qui ne montrait que ce qui était fixe: les bâtiments étaient ainsi visibles au contraire des voitures ou des gens qui se déplacent. Dans la rue, il y avait aussi beaucoup de clochards, et les chochards en général restent sur leur coin, juste avec le bras tendu pour quêter... C'était les seules personnes fixes. Ainsi de ce côté de l'image, tous les gens actifs, importants, qui font du shopping ou vont travailler sont devenus invisibles, et les gens qui ne font rien sont visibles... C'était particulièrement intéressant parce que à force de vivre dans une telle rue avec autant de clochards, vous finissez par vous créer vos propres filtres qui vous rendent ces gens invisibles. Cela m'a intéressé de voir comment une technologie simple pouvait faire basculer ces filtres que nous développons mentalement.

Xavier Ess : Vous avez fait plusieurs travaux au départ de la vidéo de surveillance, qui deviennent des travaux qui sont politiques. Est-ce que l'interactivité dans l'art jusqu'à présent, c'était pas un truc rigolo où les gens veulent jouer avec une pièce interactive pour justement créer de la musique avec son corps par exemple. Et là tout d'un coup c'est plus rigolo du tout, vos dernières pièces ?

David Rokeby : Oui, c'est vrai. De mon expérience à montrer «Very Nervous System» - qui est un travail excitant et amusant - j'ai constaté que beaucoup d'idées ou de choses qui se passent dans VNS et qui étaient intéressantes n'étaient pas toujours drôles.
Le côté amusant de VNS cache parfois les autres choses intéressantes. Et ainsi d'une certaine façon j'ai essayé de faire évoluer mon travail pour qu'il soit plus ennuyeux. Cela peut sembler bizarre, faire quelque chose d'un peu ennuyeux pour vous aider à penser un peu plus aux questions qui sont soulevées. Et si nous devons nous poser ces questions c'est parce que les technologies de l'interactivité, de l'informatique, de la réalité virtuelle, toutes ces technologies font de plus en plus partie de notre vie quotidienne, de nos communications avec les autres.
Si en même temps elles sont un apport incroyable à notre culture, je pense qu'elles ne sont pas toujours une bonne chose. Et nous devons nous poser des questions quand nous adoptons ces technologies, pour être sûrs que nous les introduisons dans nos vies de la meilleure manière qui soit, et pas simplement de la manière la plus facile qui soit... Donc mes dernières pièces sont peut-etre plus difficiles parce que je sens qu'il est nécessaire face à la guerre et au terrorisme avec tous ces systèmes de surveillance, de «détection automatique» de terroristes par exemple, de dire que la technologie n'est plus si innocente. En tant qu'artiste, je dois rendre compte de cela.

Xavier Ess : Cela c'est le propos exact de votre dernière pièce «Sorting Daemon». C'est vraiment inspiré de l'après 11 septembre ?

David Rokeby : Oui, oui... En réalité cela m'a été très dur de faire cette pièce parce que je la faisais en même temps que se préparait la guerre en Irak au printemps 2003. Il m'était bien difficile de rester concentré sur cette pièce, une pièce que j'allais montrer dans une galerie d'art, pour juste quelques personnes, au moment où des évènements importants se déroulaient dans le monde. J'étais très distrait mais je suis heureux d'avoir pu finalement terminer ce travail. Et je pense que c'était intéressant de le faire. Dans ce travail, les gens sont divisés, partagés; certaines personnes sont terrifiées de voir la façon dont les gens sont séparés de leurs vêtements, et triés suivant la taille de leur têtes ou de leurs couleurs de peau. Et d'autres

personnes trouvent juste que c'est beau et pas du tout effrayant. La question que je voulais me poser était: étais-je là pour effectivement poser la question critique ou bien la technologie était-elle si intéressante au point de distraire du débat politique? C'est le genre de questions que je veux continuer à me poser, pour continuer à être un meilleur artiste parce qu'il y a toujours des choses à apprendre, et en particulier dans ce champ des nouveaux médias où les règles ne sont pas établies.

Xavier Ess : Peut-être qu'il faut expliquer comment fonctionne cette pièce «Sorting Daemon», quel est son principe ?

David Rokeby : «Sorting Daemon» est installé dans une galerie avec une très longue fenêtre le long d'une rue. Une rue intéressante où l'on trouve aussi bien des hommes d'affaires (elle se trouve dans le quartier d'affaires), des cinémas, et aussi des dealers de crack sur les coins,... C'est une culture très mélangée. Une caméra regarde par la fenêtre; elle peut tourner et zoomer, elle cherche des personnes. Et pour elle, une personne c'est simplement quelque chose qui bouge et qui est allongé et fin (contrairement aux voitures qui sont larges et basses). Quand elle trouve une personne, elle cherche les couleurs de sa peau. Une fois trouvées, elle les enlève et sépare la tête, puis elle cherche les autres couleurs. Par exemple, la couleur de votre chemise ou de vos pantalons. Et elle trie tout cela sur un autre écran où l'on voit d'un côté, une série de têtes et de l'autre, on voit des morceaux de jeans, de manteaux, de cravates, de vestes, séparés suivant leurs couleurs et triés dans la gamme d'un arc-en-ciel. D'une certaine façon, cela ressemble à une peinture, une peinture à l'huile, avec des couches épaisses et denses. Mais cet acte de prendre le corps humain et de le découper, cette analyse très active qui prend les parties des choses et les trie..., qui extrait les personnes de leur vie dans la rue et les trie dans un espace mathématique arbitraire... il y a là une sorte de beauté mais aussi de violence. C'était bien ce que je cherchais en final, avec ce mélange d'obscur et d'attraction sombre vers cette beauté dans le contexte de la guerre et des questions soulevées autour de la vie privée et de la liberté.

Xavier Ess : Dans le titre «Sorting Daemon», c'est le démon qui trie tous les êtres humains ou bien chaque être humain est-il un démon potentiel selon certains pouvoirs ?

David Rokeby : Ici, le terme «démon» et l'expression «démon qui trie» ont été inventés par Quick Maxwell, un médecin, il y a 2 siècles. Il imaginait une petite créature mythique, qui n'existe pas, mais qu'il a proposé comme mécanisme dans une expérience mentale où ce démon trierait les molécules

en fonction de leur température: il enverrait les chaudes d'un côté et les froides de l'autre. Pour moi, c'est un peu comme le garde-frontière qui décide qui peut rentrer dans un pays et qui ne peut pas. Quand on sait que des caméras automatiques sont installées tout le long de la frontière US pour détecter les gens à interroger et séparer les terroristes potentiels, pour moi ce démon qui trie est vraiment sinistre.

Xavier Ess : C'est vrai ce que vous dites ?

David Rokeby : C'est vrai. Que cela marche ou pas, c'est une autre question. Mais c'est certainement quelque chose qu'ils essaient de faire. Si vous allez sur le site web du département de la défense US, vous trouverez le projet «Human Id» qui concerne l'identification des personnes à distance. L'idée est de pouvoir, avec une caméra à grande distance, identifier une personne pour décider soit de vous en protéger, soit de la capturer sans avertissement.

Xavier Ess : Pour terminer, on peut dire un mot d'une autre installation, «n-chant», qui est une installation de machines intelligentes. En regardant, je ne sais pas si j'étais effrayé ou fasciné parce qu'elles chantent toutes en même temps, etc. Peut-être le principe d'abord ? Comment cela fonctionne ?

David Rokeby : Dans n-chant, il y a 7 ordinateurs, qui chacun se promène dans sa mémoire, allant d'idées en idées, par exemple de l'idée de la voiture, puis l'idée de conduire, puis l'idée de route, et de là l'idée du voyage, du besoin de rentrer chez soi,... Et ce courant d'idées, ce courant de «conscience» - j'utilise des guillements parce que ce n'est pas ce que je veux dire au sens littéral - l'ordinateur l'exprime par un flot de phrases en anglais, de fragments d'anglais. C'est comme s'il rêvait éveillé, imaginait, en disant tout haut ce à quoi il pense au fur et à mesure de son cheminement dans sa mémoire. Les ordinateurs sont connectés les uns aux autres et ils s'échangent entre eux leur sujets d'intérêts du moment, et ainsi ils s'encouragent à penser aux mêmes choses. Et après quelques minutes, vous entendez que ce que cet ordinateur-ci raconte, ce que cet autre là et celui-là au fond disent tournent autour de la même chose. L'un va commencer à parler de nourriture, l'autre dire qu'il a faim, puis parler de poulet, de poulet rôti, et tout doucement ils vont converger jusqu'à ce que soudain ils disent les mêmes mots, à peu près de la même manière, et vous allez vraiment vous croire dans une église remplie de gens qui prient... Chaque ordinateur est aussi équipé d'une fonction de reconnaissance vocale. Ainsi si vous approchez d'un ordinateur pour lui parler, il tentera de vous comprendre, et vos paroles le stimuleront, lui susciteront de nouveaux centres d'intérêts qu'il élaborera. En quelque sorte, cet ordinateur devient

un dissident au milieu de la communauté. Ces nouvelles informations vont se répandre d'ordinateur en ordinateur, et le chant - la parole commune - s'écroulera dans un chaos de voix discordantes. Puis si vous les laissez seuls, ils se remettront lentement ensemble.

Xavier Ess : Et donc à la fin la communauté gagne toujours ? C'est pas possible d'être unique, un seul individu ?

David Rokeby : Je ne pense pas que le message est si noir. Il est intéressant qu'il y aie une tension entre l'individu et la communauté. Il y a des moments où le chant ne vient pas. Il y en a d'autres où l'on assiste à des glissements d'alliance, avec ces 2 qui parlent ensemble, ces 3 autres et celui-là qui va vers ce groupe... et tout d'un coup, il y en a un qui parle tout seul. Ce n'est pas vraiment rigide. Cela ne se veut pas comme un commentaire dépressif sur la nature de la société humaine, mais plus comme un moyen de regarder nos oscillations entre ces sentiments de groupe et ceux de l'individu. Et puis ce n'est pas un groupe d'humains, mais un groupe de machines. C'est important de se rappeler qu'il s'agit d'un groupe d'ordinateurs prétendant être humains. D'ailleurs les ordinateurs sont bien visibles: ils sont accrochés dans l'espace. Je suis en fait plus intéressé par des ordinateurs essayant d'être humains, que par ce que les humains font vraiment, parce que, en concevant, créant et exposant des ordinateurs tentant d'être humains, j'apprends énormément sur nos différences avec les machines, par la différence entre mes systèmes et nous, êtres humains. C'est étrange de penser que l'on peut utiliser l'ordinateur ironiquement, comme un instrument d'apprentissage de nous-mêmes parce que l'on peut tester ces idées sur ce que l'on pourrait être, et en voir certaines échouer. Je pense que les idées que nous avons sur ce que nous sommes, comment nous fonctionnons sont souvent simplistes, et l'ordinateur, d'une manière étrange, nous aide à comprendre que nous ne nous comprenons pas vraiment.

Xavier Ess : Pour terminer, vous êtes un artiste interactif optimiste ?

David Rokeby : Hey...(smile).
J'ai certainement été un artiste interactif optimiste à beaucoup de périodes de ma vie. Je suis en général une nature optimiste. De temps en temps, je suis l'artiste interactif pessimiste, parce qu'il y en a trop d'optimistes, et que parfois, par contrariété, je pense qu'il faut un autre discours. Dans le milieu de la culture en général, il y a beaucoup de voix pessimistes par rappport à la technologie, mais il n'y en a pas assez dans le milieu de la technologie. Les voix pessimistes sont utiles. Pas pour dire que la technologie est mauvaise, mais pour dire que nous devons être attentifs à la technologie

que nous prenons très vite à bras ouverts. Et parfois, dans ce processus, nous prenons les mauvaises décisions.

Entrevue avec David Rokeby *réalisée par Xavier Esse (RTBF) le 14 décembre 2003. (Traduit de l'anglais par Yves Bernard).*

- *http://www.imal.org/drokeby/archives/interviewfr.html*

David Rokeby *est artiste, pionnier des technologies interactives, diplômé du Ontario College of Art (Experimental Art). Il est le créateur de softVNS.*

Xavier Esse *est journaliste à la RTBF (Radio Télévision Belge Francophone).*

Langage dur, langage mou.

«[...] On voit qu'il existe ainsi une véritable échelle des langages, que j'appelle l'échelle de l'information synthétique parce qu'à sa base, se tiennent les données les plus dures, les plus analytiques, et à son sommet les données les plus molles, les plus synthétiques. En bas, c'est pauvre et indéformable. En haut, c'est riche et ambigu. En bas le langage de la machine, en haut la plus subtile expression humaine [...]».

«[...] Des domaines très inattendus peuvent, eux aussi, être analysés en fonction de cette «échelle de l'information synthétique». Ainsi, notre code génétique comme celui de tous les organismes vivants est «écrit» dans un langage très dur, qui ne comprend pas deux éléments, mais quatre - ce qui reste très peu. Les gènes, portés par les fibres très longues des molécules d'ADN, sont des successions de combinaisons entre ces seuls quatre éléments. Cette pauvreté extrême explique que l'écriture d'un message génétique soit interminable. Mais, un langage dur étant difficilement déformable, le code génétique se montre très résistant; les gènes sont très difficiles à modifier, et les mutations restent l'exception.

Langage pauvre, dur, mais communicable; langage riche, mou, mais intransmissible: l'homme ne trouve aucun confort dans l'un ou l'autre de ces infinis. À l'une des extrémités, grâce à l'ordinateur, l'observation scientifique permet d'engranger et de traiter un nombre gigantesque d'informations. La physique des particules élémentaires est aujourd'hui tellement formalisée que nous communiquons admirablement avec la nature; mais nous ne comprenons plus très bien ce que nous découvrons. C'est là le gros problème de la science moderne: elle arrive à des conclusions idéalement claires mais dont le sens nous échappe de plus en plus. À l'autre extrémité; nous l'avons vu, l'information est tellement riche et subtile qu'on finit par renoncer à la communiquer... L'homme préfère se tenir dans les régions médianes. Quant à l'ordinateur, qui ne s'épanouit qu'au pied de l'échelle, on voit que, malgré tout ce qu'on a pu dire, il ne sera jamais substituable à l'homme. Il restera toujours étranger au monde de l'humain trop humain, du génie, de l'intuition, des hauts niveaux de synthèse, de l'éthique - a fortiori de l'illumination [...]».

***Bruno Lussato**,*
«Le défi informatique», Paris, Fayard, 1981, pp 32-33.

***Bruno Lussato** est Professeur au Conservatoire National des Arts et Métiers et à la Wharton School, University of Pennsylvania.*

Le jeu de la vie.

*Un article de **Jean-Claude Heudin**, paru dans «Sciences et avenir» hors-série n°25, juillet 2005.*

Les algorithmes graphiques baptisés «automates cellulaires» témoignent de ce que la complexité peut émerger à partir d'une règle simple à la frontière de l'ordre et du chaos. Quelle est la portée de ces modèles informatiques ?

[...] Un système complexe est caractérisé par ce qu'il convient d'appeler des propriétés émergentes. On utilise généralement ce terme pour qualifier un phénomène qu'on ne parvient pas à expliquer par une combinaison simple des propriétés des éléments qui composent le système. Un exemple de propriété émergente est la fluidité de l'eau. Bien que la molécule d'eau soit globalement neutre, les deux atomes d'hydrogène qui la composent forment une région plutôt chargée positivement, alors que son atome d'oxygène forme une région plutôt chargée négativement. De cette dissymétrie résulte une forte polarisation de la molécule, qui est alors capable d'établir des liaisons électrostatiques avec trois autres molécules. Une coopérativité grâce à laquelle les molécules d'eau constituent, de proche en proche, un réseau tétraédrique dans l'espace. À basse température, les molécules s'écartent et se conforment à cette structure régulière pour créer des réseaux ordonnés: les cristaux de glace. Lorsque la température augmente, les liaisons électrostatiques tendent à disparaître. Les molécules d'eau s'assemblent et se désassemblent au gré des fluctuations. Il émerge alors une propriété singulière: l'eau devient fluide. Cette propriété, aucune des molécules d'eau ne la possède. Elle est le résultat du nombre gigantesque d'interactions dynamiques qui se produisent entre les molécules.

Tous les domaines de la science renferment des exemples de propriétés émergentes. Dans sa forme la plus générale, l'émergence rassemble toutes les questions fondamentales sur l'apparition de la vie et sur l'accroissement de la complexité. Cette notion est souvent résumée par la formule: «Le tout est plus que la somme de ses parties.» L'émergence marque en fait les limites de l'approche analytique. En effet, en essayant de réduire un phénomène à ses constituants les plus élémentaires, le réductionnisme sur lequel est fondée la science est essentiellement analytique. Au sens fort, il élimine les propriétés apparentes d'un système pour les remplacer par les processus élémentaires sousjacents. Cette méthode a largement prouvé son efficacité, notamment en physique. Elle permet d'expliquer n'importe quel phénomène par les propriétés physico-chimiques de ses composants élémentaires. Si bien établie soit-elle, cette approche ne parvient pas à expliciter nombre de phénomènes que l'on observe à des échelons supérieurs

de complexité. La difficulté principale vient du fait que, lorsqu'on dissocie les éléments qui constituent le tout, les propriétés émergentes disparaissent.

Il semble donc important de développer des approches complémentaires à l'analyse afin d'étudier ces phénomènes. Une classe de modèles particulièrement bien adaptés est représentée par les automates cellulaires. L'idée centrale consiste à créer, à synthétiser, à partir d'unités élémentaires et de leurs interactions, les conditions propices à l'émergence d'organisations d'un niveau supérieur. Ces expérimentations permettent d'étudier les phénomènes d'auto-organisation et de dynamique non linéaire qui caractérisent les phénomènes émergents. D'un formalisme mathématique rigoureux, les automates cellulaires ont le mérite d'être facilement programmables sur un ordinateur. Les plus célèbres utilisent une matrice de «cellules» caractérisées par deux états, 0 et 1. Sur un écran d'ordinateur, il est dès lors aisé de représenter cette matrice par une grille à deux dimensions constituée de petits carrés, par exemple blancs pour ceux dont la valeur est 0 et noirs pour ceux dont la valeur est 1. Dans l'univers virtuel de la matrice cellulaire, le temps s'écoule de façon discrète, par intervalles réguliers appelés «générations». À partir d'une configuration donnée, on calcule la génération suivante en appliquant sur toutes les cellules une même règle de transition. Celle-ci exprime le nouvel état d'une cellule en fonction de son état précédent et de celui de ses voisines.

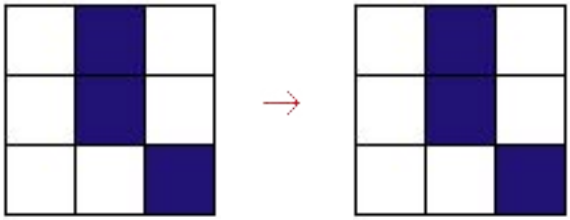
En 1970, un mathématicien de l'Université de Cambridge, John H. Conway, publie un divertissement insolite dans les colonnes de la revue «Scientific American ». Baptisé «Jeu de la vie» (Game of life), cet automate cellulaire devient rapidement la vedette des récréations informatiques. Il doit une bonne part de son succès à sa règle de transition, qui, malgré sa simplicité, produit une grande variété de structures et des dynamiques complexes évoquant l'émergence du vivant dans une soupe primitive. La règle de Conway s'exprime de la façon suivante:

1- si une cellule à l'état 1 («vivante») est entourée par deux ou trois cellules à l'état 1, alors elle conserve son état;

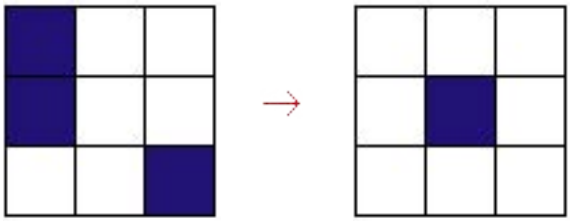
2- si une cellule à l'état 0 («morte») est entourée par trois cellules à l'état 1, alors elle passe à l'état 1;

3- dans tous les autres cas, elle passe à l'état 0 (voir schéma page suivante). En d'autres termes, la création d'une cellule vivante a lieu lors de la rencontre de trois congénères ; la mort provient pour cause d'isolement (moins de deux cellules vivantes) ou de surpopulation (plus de trois cellules vivantes).

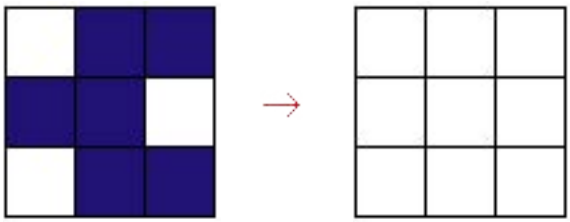
1- Si une cellule vivante (ici en bleu) est entourée de deux ou trois voisines vivantes, alors elle reste vivante.



2- Si une cellule morte est entourée d'exactement trois voisines vivantes, elle devient vivante (elle naît).



3- Dans tous les autres cas, la cellule meurt.



Une des caractéristiques de la règle de Conway réside dans son déterminisme. En effet, elle ne fait intervenir aucun aléa ni un quelconque tirage au sort. À partir d'une configuration initiale donnée, on obtient donc toujours la même succession de configurations de la matrice cellulaire. De ce point de vue, le comportement du jeu de la vie est totalement prédictible.

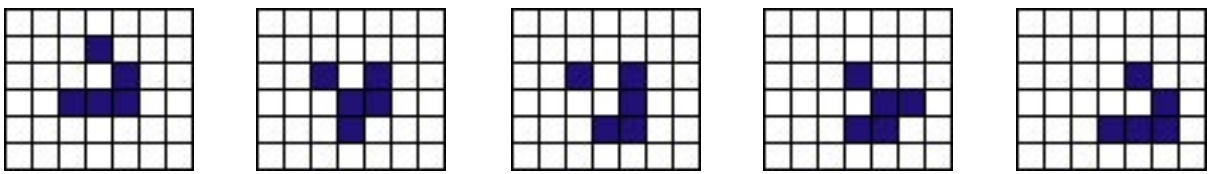
En partant d'une configuration arbitraire de cellules à l'état 1 ou 0, après seulement quelques dizaines d'itérations, la population des cellules vivantes diminue sensiblement. L'écran se peuple alors de structures étranges, de motifs géométriques qui oscillent et changent de forme. Certains se déplacent diagonalement et heurtent d'autres configurations qui disparaissent ou se transforment. Après quelques minutes d'observation, l'écran devient

pratiquement vide, à l'exception de quelques structures fixes ou oscillant entre plusieurs configurations. Le fourmillement des formes initiales a laissé la place à une certaine stabilité. Restent l'étonnement des yeux et une multitude d'interrogations qui assaillent l'esprit: comment cette complexité peut-elle provenir d'une règle aussi simple ? Quelles sont ces structures qui se déplacent comme des sortes d'organismes unicellulaires ? Le jeu de la vie aurait-il un nom prédestiné ?

L'étonnement provient en partie de l'évolution des configurations, qui défient nos prédictions malgré le déterminisme de la règle de Conway. En effet, au niveau de chaque cellule, tout est entièrement prévisible.

Par contre, si nous considérons la matrice cellulaire dans sa globalité, la combinatoire devient écrasante et il est alors très difficile de prévoir son développement sans effectivement le calculer sur un ordinateur. Néanmoins, il ne faudrait pas confondre la surprise de l'observateur avec la véritable nature de l'émergence. Celle-ci est mieux mise en évidence si l'on considère une des structures parmi les plus étranges du jeu de la vie : le «glisseur» (glider). C'est une configuration dissymétrique de cinq cellules qui se déplace diagonalement en changeant de forme, un peu comme une amibe. Son évolution consiste en une succession de quatre phases, dont deux sont des images inversées des deux autres.

Après quatre itérations, le motif original se retrouve à l'identique, mais décalé diagonalement d'une cellule (voir l'illustration ci-dessous). Sur l'écran d'un ordinateur, les glisseurs ressemblent à de petits organismes surgissant du chaos qui se dirigent frénétiquement vers un objectif inconnu et qui disparaissent lorsqu'ils heurtent un autre objet. Le glisseur de Conway est un exemple caractéristique d'émergence car il met en jeu au moins deux niveaux de complexité : celui de la matrice cellulaire et celui du glisseur proprement dit. En effet, le glisseur est constitué par un assemblage de cellules, mais celles-ci sont «remplacées» au fur et à mesure du déplacement. Sa configuration dynamique de cellules à l'état 1 évoque une forme primitive et non aboutie de «métabolisme» auto reproducteur entouré par une «membrane» de cellules à l'état 0 qui définit la frontière avec son environnement.



LE GLISSEUR DU JEU DE LA VIE : aprés quatre itérations, le motif original du glisseur se retrouve à l'identique, mais décalé d'une cellule en diagonale. Cette structure est un exemple caractéristique d'émergence mettant en jeu deux niveaux de complexité: celui de la matrice et celui du glisseur.

Une unité cohérente a donc émergé des interactions cellulaires et elle semble revendiquer son autonomie - au sens de l'*autopoïèse*[1] définie par Francisco Varela et Humberto Maturana, bien que le glisseur ne soit pas robuste face aux fluctuations de son environnement - vis-à-vis du milieu dont elle provient. Il est difficile de prédire l'apparition de ce comportement en ne considérant que le niveau cellulaire. De même, l'observation d'un glisseur en tant qu'un tout qui se distingue de son environnement ne peut, à elle seule, nous renseigner sur la nature des lois qui régissent la dynamique cellulaire. On est donc bien en face d'une propriété émergente, et non d'un simple effet de surprise qui résulte de la difficulté à appréhender la combinatoire des interactions cellulaires malgré leur déterminisme [...].

Jean-Claude Heudin,
«Sciences et avenir» hors-série n°25, juillet/août 2005.

[1] l'autopoïèse, du grec «autos» (soi) et «poiein» (produire) : «Un système autopoiétique est organisé comme un réseau de processus de production de composants qui (a) régénèrent continuellement par leurs transformations et leurs interactions le réseau qui les a produits, et qui (b) constituent le système en tant qu'unité concrète dans l'espace où il existe, en spécifiant le domaine topologique où il se réalise comme réseau. Il s'ensuit qu'une machine autopoiétique engendre et spécifie continuellement sa propre organisation. Elle accomplit ce processus incessant de remplacement de ses composants, parce qu'elle est continuellement soumise à des perturbations externes, et constamment forcée de compenser ces perturbations. Ainsi, une machine autopoiétique est un système ... à relations stables dont l'invariant fondamental est sa propre organisation (le réseau de relations qui la définit).»

- http://www.exobio.cnrs.fr/article.php3?id_article=41

Jean-Claude Heudin *a obtenu un doctorat à l'Université de Paris XI. Ses travaux portent sur les architectures et les algorithmes inspirés par le vivant, notamment les algorithmes génétiques. Il est professeur au Pôle Universitaire Léonard de Vinci où il dirige le laboratoire de recherche de l'Institut International du Multimédia http://www.virtual-worlds.net. Il est l'auteur de nombreux ouvrages et publications dans le domaine des sciences de la complexité et plus particulièrement sur les automates cellulaires, les mondes virtuels, le calcul génétique et la vie artificielle.*

Les nouveaux habits de la copie,
création en réseau et collaboration dans le monde digital.

*Extrait d'une conférence de **Nicolas Malevé** dans le cadre de «Couper/Copier/Coller#3» le 18 mars 2006 à La Vénerie (Watermael-Boisfort).*

L'informatique et les réseaux.

Contrairement aux technologies dominantes comme la télévision ou la radio (ce qu'elle est devenue), internet permet avec des moyens relativement réduits de recevoir de l'information (lire du texte, écouter de la musique ou de la video), d'en diffuser, de créer des liens avec d'autres (sites, personnes) et de collaborer sur des ressources partagées. Internet contrairement au canal audiovisuel ne vous considère pas comme une patate de canapé mais comme une personne dont la participation, la contribution est d'emblée possible.

La copie est dans le coeur du réseau.

La société de l'information qui a récemment émergé est construite conformément aux principes mis en place par les scientifiques qui ont inventé le net. Subventionnés par l'état et les fondations, les académiciens collaborent en disséminant leur savoir dans des journaux et lors de conférences. Les scientifiques n'avaient pas besoin d'une infrastructure pour vendre de l'information car leur intérêt est d'être cités et donc de voir leurs communications diffusées le plus largement. Ils ont construit le net à l'image de l'économie du don qui régit les circuits académiques. Conçu pour leur usage, ils ont inventé une forme de communication informatique pour partager l'information dans un espace virtuel: une communauté intellectuelle.
(paragraphe adapté sauvagement de http://www.constantvzw.com/copy.cult/texts/reg_liberty3.html)

Une conséquence importante de ceci est que l'ensemble du réseau internet fonctionne techniquement sur le principe de la copie. Pour que l'information soit acheminée d'un ordinateur à l'autre, pour que je puisse lire une page web, un grand nombre de copies doivent s'opérer. Caches, copies temporaires, sites miroirs sont autant de mécanismes nécessaires pour le fonctionnement optimal du réseau.

Copier, c'est apprendre.

Lorsque notre browser/navigateur lit une page internet, il ne reçoit pas seulement une image, il reçoit une série d'instructions qui lui expliquent comment afficher l'information. Ainsi chaque fois que nous visitons une page internet, nous en recevons à la fois l'apparence qui nous permet de la lire, mais aussi la source, c'est-à-dire comment elle est faite. En sélectionnant la commande «view source - afficher la source», nous pouvons lire le code informatique (HTML) qui la compose. En véhiculant toujours à la fois l'information sous sa forme consommable et sous sa forme d'instructions, internet est son propre manuel. Incitant toujours à sa propre consommation ET à sa compréhension, transformation.

Le copyleft et la General Public License.

La GPL (Licence Publique Générale) a été créée par Richard Stallman, de la Free Software Foundation, en 1983 et adoptée par les informaticiens du logiciel libre. Celle-ci garantit sans équivoque:

- le droit d'utiliser un logiciel sans contrainte
  (on peut utiliser un programme pour n'importe quel usage),

- le droit d'étudier (on peut apprendre comment le programme fonctionne),

- le droit de copier, de modifier et de distribuer des copies gratuitement
  ou commercialement.

Ces caractéristiques en font une licence copyleft, c'est-à-dire une licence "tous droits renversés" (all rights reversed).
La GPL implique une généalogie. Pour comprendre le sens du mot "généalogie" dans ce contexte, il est nécessaire de savoir que le copyleft n'est pas la négation du droit d'auteur. Il est une reformulation de la manière dont le droit d'auteur est appliqué. Parce que je suis l'auteur d'une oeuvre, je puis attribuer par contrat à mes utilisateurs plus de libertés que la loi ne leur en donne par défaut. Pour pouvoir autoriser des usages supplémentaires sur une production, il faut en avoir la propriété. Et dans le domaine des biens intellectuels, cela signifie en être l'auteur(ou avoir les droits équivalents à ceux de l'auteur). Ces droits supplémentaires sont attribués à une seule condition: que la même liberté d'utilisation soit garantie pour toute oeuvre dérivée de celle qui est mise sous copyleft.

On ne peut donc pas mettre sous copyleft une oeuvre dont on n'a pas les droits (on ne peut pas «blanchir» une oeuvre) et on ne peut pas restreindre les autorisations qui ont été données à l'usage d'une oeuvre libre, ni pour elle, ni pour les oeuvres dérivées (on ne peut pas mettre sous copyright une oeuvre sous copyleft). Une oeuvre généalogique en informatique est donc un ensemble de petits programmes combinés pour créer un nouveau programme. La contrainte à laquelle on est tenu lorsqu'on fait ce genre d'oeuvre est de mentionner les auteurs des différents petits programmes (pour que l'on puisse remonter jusqu'à ces auteurs) et de donner l'adresse web à laquelle on peut trouver ces différents petits programmes.

Dans le contexte où le copyleft a vu le jour, le monde de l'informatique, la réutilisation du code est un enjeu fondamental. Les programmeu/r/se/s écrivent du code générique sur lequel les autres peuvent construire des applications de plus haut niveau. Sans cela, il faudrait à chaque nouveau programme réinventer la roue. Ainsi offrir une base de code ouvert représente un énorme avantage, celui de permettre aux informaticien/ne/s de consacrer leur temps à l'écriture de ce qui reste à écrire plutôt qu'à ce qui est déjà écrit.

Voilà comment des programmes aussi raffinés que le système d'exploitation Linux, le server web Apache ont vu le jour. D'ailleurs, la machine sur laquelle j'ai réalisé cette présentation fonctionne uniquement sous logiciels libres. Le copyleft tel qu'il est défini par la GPL recouvre donc un ensemble de choses : à la fois une technique juridique (réappropriation du droit d'auteur), et une méthode de dissémination (la généalogie).

La culture libre.

Pour bon nombre d'artistes/acteurs culturels qui ont cherché à articuler une proposition différente à la table rase, au culte de l'individualisme, l'existence des réseaux, de logiciels libres et les pratiques d'échange qu'ils supposaient ainsi que la licence GPL a constitué une inspiration formidable. Au début des années 2000, le monde de la culture et des arts a commencé à vouloir transposer, adapter les idées et les outils juridiques à ses pratiques. Des outils gratuits et libres pour la création interactive ont vu le jour à un rythme sans cesse accéléré. Des réseaux de distribution alternative ont vu le jour. Et des créations qui empruntent le modèle de la généalogie, qui reposent sur la copie volontaire et les réseaux ont commencé à se répandre. Ce mouvement a été nommé par Lawrence Lessig, la *Free Culture*, culture libre.

Avant d'aller plus avant dans sa découverte, je voudrais insister sur plusieurs choses. Ce mouvement est décentralisé et pluraliste. Il n'est pas gouverné par un leader quelconque, même si des voix influentes se font entendre. Il n'est pas limité géographiquement ni politiquement. On peut trouver son discours relayé par la voix d'un milliardaire africain, des institutions muséales, le ministre de la culture brésilien, des entreprises cotées en bourse, des activistes altermondialistes, et des milliers d'internautes de tout genre et de toutes races.
Enfin, les projets qui naissent en son sein ne retiennent pas souvent la distinction entre un art savant ou cultivé et un art populaire. Ils supposent une participation plus ouverte et débouchent peu souvent sur une unité de bon goût. Leur intérêt principal étant la dynamique culturelle et sociale qu'ils génèrent.

Les Creative Commons.

Créées en 2001, par une équipe essentiellement académique (juristes, scientifiques, entrepreneurs et un cinéaste documentaire) regroupée autour du juriste Lawrence Lessig, soutenues par une fondation et plusieurs universités, les CC sont des licences inspirées de la GPL en y apportant de nombreuses nuances. Les CC se présentent comme «les garants de l'équilibre, du compromis et de la modération», comme «l'étendue des possibilités entre le tout-copyright — tous droits réservés — et le domaine public — sans droits réservés». Ces licences vous aident à conserver votre droit d'auteur tout en autorisant par avance certains usages de votre travail — un copyright — certains droits réservés.

- http://creativecommons.org/

Contrairement à la GPL qui octroie par défaut à l'utilisateur une série de droits (la modification, la redistribution, l'usage commercial), les CC permettent à l'auteur de faire un choix parmi les droits qu'il/elle octroie aux utilisateur/trice/s. On peut ainsi:

- octroyer le droit de modification d'une oeuvre ou non,
- l'usage d'une oeuvre dans un cadre commercial ou non
- et imposer ou non que les oeuvres dérivées soient distribuées sous les mêmes conditions.

Les licences CC ont un succès mondial retentissant. On compte par millions les productions publiées sous ces licences. CC a investi énormément de temps et d'effort pour convaincre des compagnies, des institutions, des

groupes de mettre en place des services qui permettent d'ientifier les oeuvres qui sont publiées sous ces licences et quelle utilisation on peut en faire. Des sites internet célèbres comme archive.org ou flickr.com mettent à disposition des oeuvres visuelles, sonores, animées, interactives. Les licences CC attachées à ses oeuvres s'adressent à vous, visiteur/euse, non comme consommateur/trice passif/ve, mais comme potentiellement acteur/trice. Voici ce que vous pouvez faire avec l'oeuvre. Les CC ont aussi promu et encourager des sites internet qui permettent d'échanger des composantes sonores pour intégrer à vos montages, comme le projet freesound... Enfin, les CC ont aussi convaincu les moteurs de recherche principaux comme yahoo, google d'avoir une option de recherche pour les contenus ouverts. On peut ainsi chercher sur le web des images, des pages web, etc qui correspondent aux permissions dont nous avons besoin: on peut chercher une image de «bison» que l'on peut modifier dans un cadre non-commercial, etc.

- Recherche d'image sur creativecommons.org
- Creative Commons search on google
- Creative Commons search on yahoo

Projets artistiques et la Licence Art Libre.

Plus près de nous, la licence Art Libre a été conçue par un collectif, Copyleft Attitude, composé plus d'artistes que de juristes. Cette licence très proche dans sa philosophie de la GPL a été appliquée depuis 2000 dans de nombreux projets collaboratifs en France, en Belgique, dans la francophonie en générale et depuis peu dans les pays latins. La licence Art Libre autorise la copie, la redistribution, la modification même commerciale pour autant que les oeuvres originales soient mentionnées et leurs auteurs crédités. Les oeuvres dérivées doivent accorder les mêmes libertés.

- http://artlibre.org/

Copyleft Attitude cherchait dans la GPL un outil de transformation culturelle plutôt qu'un moyen commode pour faciliter la diffusion d'une oeuvre. Le monde de l'art (la diffusion de la culture) était perçu comme entièrement dominé par la logique marchande, les monopoles et les diktats émis depuis des cercles fermés. Copyleft Attitude voulait renouer avec une pratique artistique qui n'était ni centrée sur l'auteur, qui encourageait la participation plutôt que la consommation, et qui brisait le mécanisme de la rareté qui est à la base des processus d'exclusion dans le monde artistique en fournissant le moyen d'encourager diffusion, multiplication, etc.

Les auteurs sont invités à créer des matériaux libres sur lesquels d'autres sont invités à travailler. Les auteurs sont invités à recréer une origine artistique à partir de laquelle une généalogie d'oeuvres libres peut se déployer. Plutôt que d'épiloguer sur la LAL, je préfère présenter quelques projets.

Peinture de peintres.

Le premier de ces projets est réalisé par Antoine Moreau, initiateur du collectif Copyleft Attitude. Le projet s'appelle «Peinture de peintres». Il le décrit comme ceci :
«Je propose à des peintres de rencontre de se peindre les uns par dessus les autres. Une peinture en recouvre une autre. Ce travail a été commencé il y a quelques années, il se fait sans pression. Cette peinture n'aura pas de fin. Pas d'image arrêtée. La peinture a une dimension de 88,5 cm/101,5 cm. Ce n'est pas une compilation de mes goûts en peinture, ni une collection de peintres. La qualité photo de ces peintures est approximative et inégale. Sur le web, c'est juste un coup d'oeil.»

- http://antoinemoreau.org/g/category.php?cat=4&start=15

On pensera immédiatement au chemin parcouru entre la confrontation de Kooning/Rauschenberg *. Le principe n'étant plus ici d'effacer la présence de l'autre, mais de composer avec lorsque l'inspiration le permet. L'oeuvre produite ne se résume pas à un tableau effacé, ni à la dernière surface peinte. L'oeuvre est le processus et le medium électronique est le support/témoin des modifications, témoin de l'évolution de la matière picturale qui ne peut qu'exclure un état pour un autre. Mais comme le souligne AM, sur le web c'est juste un coup d'oeil, une copie de basse qualité. L'oeuvre n'est nulle part, ou plutôt elle est virtuelle.

** En début de texte, Nicolas évoque cette anecdote:*
En 1959, Robert Rauschenberg, jeune artiste à l'époque, demanda à Willem de Kooning de participer à un projet artistique. De Kooning, plus agé et plus célèbre, accepta de lui donner du crédit et donna à Rauschenberg un dessin qu'il considérait important. Le dessin était constitué de matériaux gras et les gestes rapides et puissants de de Kooning avaient pénétré la feuille de papier profondément. Rauschenberg passa un mois à effacer complètement le dessin. Ensuite, il l'encadra et écrivit comme légende:»Dessin de de Kooning effacé, 1953». Aujourd'hui ce dessin fait toujours partie de la collection privée de Rauschenberg.

ref: http://www.csulb.edu/~jvancamp/multi/discuss.html

Le projet Sheep's Parade.

Le projet «Sheep's Parade» a été lancé en Mars 2005 à l'occasion d'un appel à participation formulé comme ceci: «Fall and Spring Sheep's Parade» (parade printemps/automne des moutons), une allusion aux collections de mode féminine lancées chaque saison. Cette parade est liée aux revues collaboratives d'un musée en ligne, le «Musée de l'essentiel et au-delà». Ces revues se proposent de faire l'analyse des travaux «web.art» de 3 femmes-artistes habitant 3 continents différents: Europe(France), Amérique du Nord (USA) et Amérique Latine (Chili). Afin de promouvoir le rôle essentiel que prennent les femmes artistes dans l'art sur le réseau. Sheep's Parade est un projet initié par Isabel Saij, la première artiste invitée.

Le premier appel à participation pour la «parade des moutons» était humoristique et jouait d'emblée la carte de la polysémie: envoyez votre représentation d'un mouton, dessin, peinture, photo, animation,.. n'importe quel genre de mouton: Dolly (tendance clonage...), blanc, fashion, rose bonbon, vert électrique, enfin tout mouton qui vous viendrait à l'esprit, même le fameux mouton noir serait le bienvenu!

Les commentaires du site internet témoignent assez bien de la vie d'un tel projet et de son évolution. Les premiers moutons sont drôles colorés et suscitent des commentaires amusés. Et puis progressivement, les moutons prennent la couleur des préoccupations de chacun et chacune. Le mouton devient un animal sacrificiel, puis une métaphore politique. «Le mouton noir était devenu en France une sorte de symbole du «non» de gauche dans la campagne référendaire sur le traité constitutionnel européen avant le vote du 29 mai 2005. C'est ainsi que la parade des moutons est devenue une parade historique !

Le projet connaît à présent une nouvelle mutation, il entre dans sa deuxième phase, celle d'une oeuvre généalogique: le «BIG SHEEP». C'est une proposition de clonage faite par Isabel Saij et Regina Célia Pinto (auteures du projet). Toutes les deux semaines, elles commentent deux des créations du premier troupeau de moutons et mélangent visuellement leur ADN. Les créations originales sont disponibles sur le blog afin d'être copiées et modifiées. Les mutants seront expédiés sur le site web du musée virtuel et iront former le troupeau «BIG SHEEP» qui sera exposé en permanence à la galerie du clonage du «Musée de l'Essentiel et Au-delà».

Comme on le voit, le rôle des artistes dans ce projet est de créer un contexte participatif, d'encourager avec humour les dérives enrichissantes et de rebondir sur celles-ci pour engager de nouvelles étapes. Le rôle de la licence Art Libre sert ici à clarifier l'usage que les participant/e/s peuvent faire des oeuvres et les usages qu'ils et elles doivent autoriser pour les

créations qu'ils/elles y envoient.

- http://bigsheep.blogspot.com/

Adam project.

Adam Project est un site web qui recense des journaux d'images réalisés par différent/e/s participant/e/s. Initié lors d'une journée Copyleft, ce site permet à différent/e/s utilisateur/rice/s de lier leurs journaux par mots-clés. Une consigne est donnée aux participant/e/s, celle de respecter ou d'interpréter une règle exprimée sous la forme d'une liste de 12 conseils (prenez une photo au lever du lit, ne pas faire de jolies photos, etc). Des rapprochements se créent entre les différentes journées et des parcours se tissent entre les différentes intimités. Les règles ont été définies par l'auteur et chaque participant les interprète avec plus ou moins de distance ou de créativité. Une poésie attachante se dégage de l'ensemble, une approche de l'intimité sans voyeurisme, à la fois célébration tranquille de la banalité et un air de fluxus, d'art par instruction. Si l'Adam Project est montré dans des galeries d'art (c'est de l'art libre), il est d'abord une création pour internet.

Si on peut le voir comme une création artistique, il est aussi la construction d'une communauté non-réciproque. On participe à ce projet et des liens se créent entre les images, les fils linéaires se croisent, mais rien ne se crée nécessairement entre les gens.

Ils/elles ne doivent pas se répondre ni s'interpeller. Le dispositif se charge de les relier. Si les liens se créent, ce n'est pas seulement que les photos sont mises en ligne, mais c'est aussi qu'elles sont décrites et que le dispositif encourage les participant/e/s à regarder les images. Comme l'espace dans un tableau traditionnel est organisé par des lignes de fuite, l'espace virtuel de cette communauté de journaux est organisé par le schéma d'une base de données.

Comme la perspective était la forme symbolique de la peinture renaissante, la base de données est la forme symbolique des systèmes collaboratifs, la grammaire avec laquelle s'écrive les liens, se définissent les proximités et les éloignements. La navigation par mots-clés loin d'être simplement une classification utilitaire est un élément narratif doublé d'une relecture assez fine des images. Une voiture aperçue par une fenêtre devient le lien qui conduit à une autre image, un détail sorti d'un contexte devient une porte dérobée qui mène à l'instant d'une autre vie. Adam Project possède beaucoup des caractéristiques que l'on retrouve dans l'idéologie/idéal attaché à la LAL. La volonté de redéfinir un cadre pour un art qui se partage, qui s'échange et se réinterprète. La licence sert ici à souligner une attitude plus ouverte quant au droit moral attaché aux contributions,

elle souligne l'accord de chaque participant que les liens faits entre une image et une autre, et éventuellement les glissements de sens que cela génère sont autorisés/souhaités. La présence de la licence au début de la journée de chaque participant au-dessus du bouton «entrez» fonctionne plus comme un signe de bienvenue que comme la marque d'une frontière.

- http://www.adamproject.net/

En conclusion.

Pour conclure, j'aimerais une remarque extraite d'une conférence de Larry Wall, le créateur du langage informatique Perl. Il résume à mon sens une préoccupation qui traverse les communautés de logiciels libres, de la culture ouverte et de l'art libre. «Vous avez déjà entendu le dicton: si vous avez un marteau en main, tout commence à ressembler à un clou. C'est en fait un dicton moderniste. La réponse postmoderne est de vous donner du tape, de la toile isolante». Nous avons hérité d'une culture qui assène la rupture et qui fragmente la tradition en ilôts propriétaires. La culture «libre» tente de recoller les morceaux, réparer les accrocs, tisser des liens. Et ce sont ces liens tissés qui constituent les nouveaux habits de la copie.

*Extrait d'une conférence de **Nicolas Malevé** dans le cadre de «Couper/Copier/Coller#3» le 18 mars 2006 à La Vénerie (Watermael-Boisfort).*

Texte complet en ligne :

- *http://www.stormy-weather.be/wiki/index.php/Les_nouveaux_habits_de_la_copie*

***Nicolas Malevé** est un membre actif de l'association Constant depuis 1998. Il a pris part à l'organisation de différentes activités liées aux alternatives au droit d'auteur, comme Copy. cult & The Original Si(g)n, en 2000. Il a développé des applications web pour des organisations culturelles. Sa recherche actuelle est orientée vers les strucures de l'information, les metadata et le web sémantique.*

La technologie comme modèle idéologique (de la trace au programme).

Extrait de "Esthétique relationnelle"
Nicolas Bourriaud *(Les presses du réel, Paris, 1998), pages 71-73.*

La technologie, en tant que productrice de biens d'équipement, exprime l'état des rapports de production: la photographie correspondait jadis à un stade de développement donné de l'économie occidentale (caractérisé par l'expansion coloniale et la rationalisation du processus de travail), stade de développement qui appelait, d'une certaine manière, son invention. Le contrôle de la population (apparition des cartes d'identité, des fiches anthropométriques), la gestion des richesses d'outre-mer (l'ethno-photographie), la nécessité de maîtriser à distance l'outillage industriel et de se documenter sur les sites à exploiter, donnèrent à l'appareil photo un rôle indispensable dans le processus d'industrialisation. La fonction de l'art, par rapport à ce phénomène, consiste à s'emparer des habitudes perceptives et comportementales induites par le complexe technico-industriel pour les transformer en possibilités de vie, selon l'expression de Nietzsche. Autrement dit, à renverser l'autorité de la technique afin de la rendre créatrice de manières de penser, de vivre et de voir. La technologie qui domine la culture de notre époque est bien entendu l'informatique, que l'on pourrait diviser en deux branches: d'une part, l'ordinateur lui-même et les modifications qu'il entraîne dans notre mode de sentir et de traiter l'information. D'autre part, l'avancée rapide des technologies conviviales, du minitel à internet, en passant par les écrans tactiles et les jeux vidéo interactifs. La première, qui touche au rapport de l'Homme aux images qu'il produit, contribue prodigieusement à la transformation des mentalités: en effet, avec l'infographie, il est désormais possible de produire des images qui sont le fruit du calcul, et non plus du geste humain. Toutes les images que nous connaissons sont la résultante d'une action physique, de la main qui trace des signes jusqu'à la manipulation d'une caméra: les images de synthèse, elles, n'ont nul besoin pour exister d'un rapport analogique à son sujet. Car «*la photo est l'enregistrement travaillé d'un impact physique*», tandis que «*l'image numérique, elle, ne résulte pas du mouvement d'un corps, mais d'un calcul* *». L'image visible ne constitue plus la trace de quoi que ce soit, sinon celle d'un enchaînement de nombres, et sa forme n'est plus le terminal d'une présence humaine: les images «*fonctionnent désormais toutes seules*» (Serge Daney), à l'instar des *Gremlins* de Joe Dante qui s'auto-reproduisent par pure contamination visuelle.

* Pierre Lévy, «La machine Univers - Création, cognition et culture informatique», Points Seuil, Paris, 1987, p50.

L'image contemporaine se caractérise précisément par son pouvoir générateur; elle n'est plus trace (rétroactive), mais programme (actif). C'est d'ailleurs cette propriété de l'image numérique qui informe l'art contemporain avec le plus de force: déjà, dans une grande part de l'art d'avant-garde des années soixante, l'oeuvre se donnait moins comme une réalité autonome que comme un programme à effectuer, un modèle à reproduire (par exemple, les jeux inventés par Brecht et Filliou), une incitation à créer soi-même (Beuys) ou à agir (Franz Erhard Walter). Dans l'art des années quatre-vingt-dix, alors que les technologies interactives se développent à une vitesse exponentielle, les artistes explorent les arcanes de la sociabilité et de l'interaction. L'horizon théorique et pratique de l'art de cette décennie se fonde en grande partie sur la sphère des relations inter-humaines. Ainsi les expositions de Rirkrit Tiravanija, Philippe Parreno, Carsten Höller, Henry Bond, Douglas Gordon ou Pierre Huyghe construisent-elles des modèles de socialité aptes à produire des relations humaines, comme une architecture «produit» littéralement les itinéraires de ceux qui l'occupent. Il ne s'agit toutefois pas de travaux sur la «sculpture sociale» au sens où l'entendait Beuys: si ces artistes prolongent bel et bien l'idée d'avant-garde, jetée avec l'eau du bain moderne (insistons sur ce point, encore qu'il faudrait trouver un terme moins connoté), ils n'ont pas la naïveté ou le cynisme de «faire comme si» l'utopie radicale et universaliste était encore à l'ordre du jour. On pourrait parler à leur, sujet de micro-utopies, d'interstices ouverts dans le corps social.

Ces interstices fonctionnent à l'instar de programmes relationnels: économies-mondes où se renverseraient les rapports du travail et du loisir (exposition *Made on the 1st of May* de Parreno, Cologne, mai 1995), où chacun pourrait rentrer en contact avec les autres (Douglas Gordon), où l'on réapprendrait la convivialité et le partage (*les cantines nomades* de Tiravanija), où les rapports professionnels feraient l'objet d'une célébration festive (*Hôtel occidental*, vidéo d'Henry Bond, 1993), où les gens seraient en contact permanent avec image de leur travail (Huyghe). L'oeuvre propose donc un modèle fonctionnel, et non pas une maquette; c'est-à-dire que la notion de dimension n'entre pas en ligne de compte, exactement comme dans l'image numérique dont les proportions peuvent varier selon la taille de l'écran, qui - à l'inverse du cadre - n'enserre pas les oeuvres dans un format préétabli mais matérialise des virtualités en x dimensions. les projets des artistes d'aujourd'hui possèdent la même ambivalence que les techniques dont ils s'inspirent indirectement: écrits dans et avec du réel comme les oeuvres filmiques, ils ne prétendent cependant être la réalité; d'autre part, ils forment des programmes, à l'instar des images numériques, sans toutefois garantir l'applicabilité de ceux-ci, pas plus que l'éventuel transcodage dans d'autres formats que celui pour lequel ils ont été conçus.

En d'autres termes, *l'influence de la technologie sur l'art qui lui est contemporain s'exerce dans les limites que circonscrit celle-ci entre le réel et l'imaginaire.*

L'ordinateur et la caméra délimitent des possibilités de production, qui eux-mêmes dépendent des conditions générales de la production sociale, des rapports concrets existant entre les Hommes: à partir de cet état des choses, les artistes inventent des modes de vie, ou bien rendent conscient un moment M de la chaîne de montage des comportements sociaux, permettant d'imaginer un état ultérieur de notre civilisation.

Nicolas Bourriaud *est critique d'art, directeur de la rédaction de la revue «Documents sur l'art», commissaire d'expositions et ancien co-directeur du Palais de Tokyo, site de création contemporaine, à Paris.*

Sa définitin de l'art:
«L'art c'est la production d'un rapport au monde à l'aide de signes, de formes ou de gestes; il s'agit d'une économie dans laquelle l'artiste crée des relations entre des gens ou entre des choses, relations qui échappent aux circuits de communication institutionnels».

Bibliographie.

Initiation à la programmation
Claude Delannoy (Eyrolles, 2002).

Le livre de Java : Premier langage
Anne Tasso (Eyrolles, 2005).

Code de création
John maeda (Thames & Hudson, 2004).

L'art numérique
Edmond Couchot & Norbert Hillaire (Champs/Flammarion, Paris 2005).

L'art numérique (Digital art)
Christiane Paul (Thames & Hudson, 2004).

L'art Internet
Rachel Greene (Thames & Hudson, 2005).

Les Nouveaux Médias dans l'art
Michel Rush (Thames & Hudson, 2004).

Logic & design in art, science & mathematics
Krome Barratt (Lyons Press, 1993).

Comment choisir un langage de programmation
Thomas Pornin (H&K, Paris, 2005).

Une histoire de l'informatique
Philippe Breton (Points sciences/Seuil, Paris 1990).

La machine Univers (Création, cognition et culture informatique)
Pierre Lévy (Points Sciences/La Découverte, Paris, 1987).

Ressources en ligne.

Le site de PROCESSING.
- http://processing.org/

La page dédiée à PROCESSING sur multimedialab.
- http://www.erg.be/multimedialab/cours/logiciels/processing.htm

Le cours de Douglas Edric Stanley, prof à Aix-en-Provence.
- http://www.ecole-art-aix.fr/rubrique.php?id_rubrique=81

Le cours de Emmanuel Lestienne à la HEAJ à Namur.
- http://www.bugsbusy.be/processing/index.html

Remerciements.

Pour d'excellents moments et d'enrichissantes discussions :

- Jérôme Decock, Manuel Abendroth et Els I.R.L. Vermang
du collectif LAB[au] architecture and urbanism.
  - http://www.lab-au.com/

- Yves Bernard, de l'association iMAL (interactive Media Art Laboratory),
professeur d'arts numériques à l'Erg
(École de Recherche Graphique) à Bruxelles.
  - http://www.imal.org/

- Emmanuel Lestienne,
professeur de programmation à la HEAJ
(Haute École Albert Jacquard) à Namur.
  - http://www.bugsbusy.be/

- Stéphane Noël,
professeur d'arts numériques à l'Erg
(École de Recherche Graphique) à Bruxelles.
  - http://www.lescorsaires.be/

- Michel Cleempoel,
professeur d'arts numériques à l'ENSAV
(École Supérieure des Arts Plastiques et Visuels) à Mons.
  - http://www.esapv.be/

- Nicolas Malevé, de l'association Constant VZW,
professeur de techniques infographiques et multimedias à la HEAJ
(Haute École Albert Jacquard) à Namur.
http://www.stormy-weather.be/wiki/

- Eric Angenot, Marcel Berlanger, Frédéric Gaillard, Juan d'Oultremont
et mes collègues de l'Erg, ainsi qu'Yvan Flasse, Directeur.

- Alain Debaisieux, Compositeur et producteur.

- Rudy Léonet, journaliste et producteur.

- Yves Bigot, journaliste, écrivain, réalisateur et producteur.

- Dieudonné Leclercq, professeur
et Brigitte Denis, chercheur et chargée de cours
à la Faculté de Psychologie et Sciences de l'Éducation de l'Université de Liège.
  - http://www.fapse.ulg.ac.be/

à Charlotte, Juliette et Maxime...