Performing Process
the Artist Studio as Interactive Art

Luis Blackaller
Hon.B.Sc. Mathematics
Universidad Nacional Autonoma de Mexico
September, 1997

Submitted to the Program of Media Arts and Sciences,
School of Architecture and Planning
in partial fulfillment of the requirements for the degree of
Master of Science in Media Arts and Sciences at the
MASSACHUSETTS INSTITUTE OF TECHNOLOGY
May 2008

© Massachusetts Institute of Technology 2008.
All rights reserved.

Author **Luis Blackaller**
Program of Media Arts and Sciences,
School of Architecture and Planning
May 18, 2008

Certified by **John Maeda**
Associate Professor of Design and Computation
E. Rudge and Nancy Allen Professor of Media Arts and Sciences,
MIT Media Laboratory
Massachusetts Institute of Technology
Thesis Advisor

Accepted by **Deb Roy**
Chairman, Department Committee on Graduate Students

Performing Process
the Artist Studio as Interactive Art

Luis Blackaller
Hon.B.Sc. Mathematics
Universidad Nacional Autonoma de Mexico
September, 1997

Submitted to the Program of Media Arts and Sciences,
School of Architecture and Planning
in partial fulfillment of the requirements for the degree of
Master of Science in Media Arts and Sciences at the
Massachusetts Institute of Technology
May 18, 2008

Abstract

Digital technology has radically changed human communication in the last few decades. The digital medium has pushed cultural artifacts towards forms defined by interaction, participation, and social systems. This thesis looks at recent changes of artistic practice in the realm of digital visual arts. In order to study the intersection between interactive art and the digital image, I will describe the design of an online participatory studio system, where an artist can perform the creative process and respond to remote feedback from audience participants at the same time. Such a system opens a conversation between artist and audience that will shed new light on how we can learn, understand and communicate boundaries in digital space and digital interactive art.

Thesis Advisor **John Maeda**
Associate Professor of Design and Computation
E. Rudge and Nancy Allen Professor of Media Arts and Sciences,
MIT Media Laboratory
Massachusetts Institute of Technology

Performing Process
the Artist Studio as Interactive Art

Thesis Reader **Casey Reas**
Associate Professor
UCLA Department of Design — Media Arts
University of California

Performing Process
the Artist Studio as Interactive Art

Thesis Reader **Nick Montfort**
Assistant Professor of Digital Media
MIT Program in Writing and Humanistic Studies
Massachusetts Institute of Technology

Acknowledgments

First, I'd like to thank John Maeda for his encouragement and inspiration, and because he gave me the opportunity to learn from him and MIT.

I also want to thank my thesis readers Nick Montfort and Casey Reas for their patience and their valuable feedback.

Abhimanyu Das and Amy Strong, thanks for the long nights proofreading my english.

Takashi Okamoto and Kyle Buza, my fellow crew members, thanks for all the knowledge you've shared with me, and mostly, thanks for making the PLW feel like home, and making 5:00 am feel like noon.

Amna Carreiro, thanks for everything. I especially appreciate the way you drive me crazy. I love you.

Brent Fitzgerald and Amber Frid-Jimenez, the ones that were there before. Thanks for your welcoming spirit, your advice, and, especially Brent, thanks for your support. Kate Hollenbach, thanks for the inspiration. Sanghoon Lee, thanks for your support. Mariana Baca, thanks for your insight.

I also want to thank a few professors that helped me learn, focus, and remember what it is all about:

- Gerald Jay Sussman, thanks for teaching me how to look at programming as a philosophy of life.
- Henry Jenkins, thanks for giving me back my strength.
- Frank Espinosa, thanks for making me pick up the paintbrush again. You alone turned MIT into a different place.

- Junot Diaz, thanks for trusting me, and for sharing your own process with me.
- Antoni Muntadas, thanks for São Paulo, and for making me fall; It's made me stronger.

I want to thank Michele Oshima, because she introduced me to some to my best friends in Cambridge.

Gabriella Gomez-Mont is responsible for sending me to meet John in 2003, so this whole thing would have no happened without her.

Most importantly, I want to thank John and Mary Watkins, whose hospitality and friendship welcomed me when I arrived to Boston.

And finally, I want to thank my muse, Amy Strong, for her love and friendship, and my parents Eduardo Rogelio Blackaller Villareal and Maria Bages Martinez, for partially making me what I am.

Contents

1 Introduction **19**

1.1 Motivation 19

1.1.1 Pictures and Interaction 19

1.1.2 Art and Participation 21

1.1.3 The role of Digital Technology 22

1.2 The Studio as the space for process 23

1.2.1 Natural or Artificial 24

1.2.2 Boundaries 25

1.3 Problems 25

1.4 Thesis Overview 27

1.5 Contribution 27

2 Background **29**

2.1 Cybernetics 29

2.2 Performance, Process, Feedback 31

2.3 Digital Art 34

2.3.1 Computer Art 36

2.3.2 Communication Art 38

2.3.3 Code as a material 40

2.3.4 Commenting Code 43

2.3.5 Programs as tools 45

2.3.6 Programs as art 49

2.3.7 Participation and Communication 54

2.3.8 Performance and Participation 59

3 System **63**

3.1 Foundations 63

3.1.1 Natural Interactions 64

3.1.2 Artificial Interactions 67

3.1.3 Tools: Drawing and Animation 70

3.1.4 Social Systems: Anonymous Participation and Public Space 72

3.1.5 Code as a material: E15, a new world 78

3.2 Scope 80

3.2.1 Trust 81

3.2.2 Limits 82

3.3 Approach 83

3.3.1 Goals and Expectations 83

3.3.2 Design Alternatives 84

3.3.3 Synchrony or Asynchrony 87

3.3.4 Comments and Version Control 90

3.3.5 Sessions 91

3.4 Design 92

3.4.1 Methodology 95

3.4.2 Resources 98

3.4.3 Implementation 101

4 Analysis **105**

4.1 Conceptual Analysis 106

4.1.1 Artist and Participants 106

4.1.2 Time over space 108

4.1.3 Computational art 110

4.2 Analysis of Process 112

4.2.1 Initial conditions 114

4.2.2 Feedback 116

4.2.3 Axes of evaluation 118

5 Conclusion **121**

5.1 Result 121

5.2 Future Work 122

List of Figures

3-1 Luis Blackaller, oGFx: blood, 2007. The coding session that produced this capture was the result of several hours manipulating sensitive parameters, such as OpenGL blending functions, CoreImage image processing filters, and the variables necessary to move and draw the original elements in the scene. Like with many other experiments of its kind, keeping track of the state that produced a specific result becomes hard, suggesting the need to consider an image based code versioning system, to help the programmer roll back to particular versions of the code by browsing through an image log. 81

3-2 E15:chat mockup scenario. In this model, programmers can have a coding conversation, looking synchronously at each other's code as it is edited, and choosing to evaluate between their own code and their partner's. The option to edit each other's code is discarded because it leads to confusion; Only your own script should be edited by you. Personal communication can happen in the form of embedded comments within the code. A web based logging application can still be used for optional persistency of important states of the code. This is a perfect "future work" model to explore. 89

3-3 Studio Interaction diagram. At the same time the artist is performing in E15, participants have the option to review the performance from other instances of E15, or from a process-log web application. The artist and the participants can access and publish feedback about the performance during or after process time. 92

3-4 E15 custom components. 1: Code, snapshots and comments are sent to the studio log using buttons or custom methods embedded in the script. The same resources are used to check for feedback on specific items. 2: Feedback is printed in the console. Additional commands allow the artist to fetch feedback about specific items and post in the console. 98

3-5 E15 custom interpreter window, detail. The background window is part of the OpenGL view where the graphics specified by the script are generated. . . 99

3-6 Web components. 1: Scripts and pictures are logged together, making it easy to keep track of the exact code that produced a specific effect. 2: Comments are attached next to the published content. Comments can be posted from the web browser or directly from e15. 3: Form for comments. Authorship is optional. 100

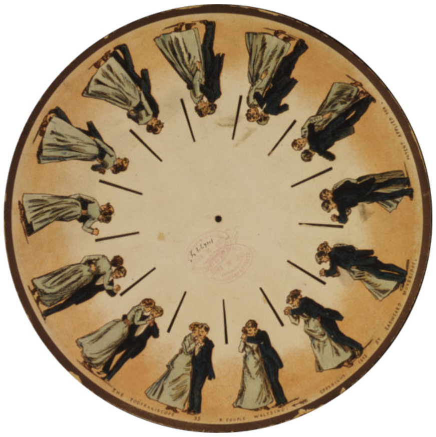
4-1 Time over space: First 110 submissions in the online studio snapshot log. The picture component becomes a sort of visual timeline. Different sessions can be identified by looking at the rate of change between consecutive pictures. Like in a movie, cuts from one scene (or session) to the other differ from smoother transitions, where analogies of shape or color maintain a visual coherence. However, visual jumps are present even within the same session. . . . 109

4-2 History command: Pencil test frames from 1943 animation classic “Red hot riding hood”, animated by Preston Blair and directed by Tex Avery. The drawings were extracted from Blair’s animation recipe books [30], loaded into E15, and projected in motion over 3D space. Graphical representations of time are particularly useful to understand motion. 112

4-3 Luis Blackaller. Pulse, 2008, 6 variations of oscillation over a rigid matrix. The 6 images featured in the figure are generated by scripts that differ in nothing more than around 8 lines of code. The scripts are less than 30 lines long. Their apparent simplicity relies on the complex resources supported by the E15 architecture, including state of the art image processing and 3D graphics technologies. 117

1

Introduction



Eadweard Muybridge's phenakistoscope from 1893. In this early example of an interactive motion picture, the beholder was required to spin the picture in order to experience the illusion of a dancing couple in motion.

1.1 Motivation

There are four primary motivations for this work:

1. To examine the contemporary boundaries of authorship within the artistic practice when mediated by the digital medium.
2. To discuss about the relationship between artistic creative process and audience participation.
3. To create interactive digital graphics.
4. To find new ways of communicating digital knowledge.

I will show how this is closely related to major traditions of western artistic practice through an informal review of related events in recent history.

1.1.1 Pictures and Interaction

Like must of us, I was born in a world made of pictures. I often learned about things through their pictures rather than the things themselves [35]. Big billboards on top of every building, swarms of images, labels and logos in every newsstand and store alley, pictures in motion on television and in the movies, and pictures of pictures in books about art. Early on in life, when going to places like preschool, I would be provided with sets of watercolors, crayons, pens, scissors, glue, and scrap from dated magazines to make pictures of my own, learning the basics of depiction, representation, figuring and appropriation. I learned to spot relationships between a given medium and how accessible a picture is in the context of that medium. Pictures in the museum or in the movies inspired me a sense of importance and permanence that I couldn't feel for television, where the simple turn of a knob would give me the option to completely change the content of the picture. The same way, books and comic books gave me an illusion of control over the space where a collection of pictures were arranged. Instead of having to move around to find the next picture, I could turn the medium at will to change from one picture to the other.

Later on, I developed an ambition beyond just making pictures, and began to explore how to make, or invade, different mediums where pictures could be held. Like many other artists, I have embedded pictures in film, television, architecture and the internet. A few years ago, thinking of architectural space as an immersive canvas, I worked in a series of modular pieces constituted by sets of silkscreened aluminum plates. Having to show the work in progress to the representative committee of a founding institution, I made a cardboard scale model to illustrate the process of assembly. I was interrupted before I could finish my carefully designed layout, when all the members of the committee engaged in a playful discussion rearranging the modules in many different ways. It became impossible for me to make them understand the purpose of my final layout because they were more interested in finishing the piece themselves than listening to what I had to say.

Recording process. Luis Blackaller, digital self portrait working in the studio, ISCP New York, 2004.

Pictures and Interaction have been coupled together long before the digital era. If we understand interaction as a kind of action that occurs when two or more objects have an effect upon each other, and pictures as the collection of all 2-dimensional visual representations, we can then say that the simple acts of drawing and writing are both interactive pictures, acting upon their creators when they are decoded back to them [75], illustrating the feedback nature of artistic creation.

Flying pelican captured by E.J. Marey around 1882. He found a way to record several steps of motion in one photo.

However, the act of drawing and the finished drawing are two different things, and a picture in visual art has traditionally been considered to be the finished image (drawing, painting, print, photo, film, collage) rather than the process of making it. Because of this, the participation of audiences in the appreciation of art has been a history of contemplation, against an art mysteriously conceived by a mythical creature called the artist.

Marcel Duchamp. Rotary Demisphere, Hirschorn Museum, Washington, DC. Inspired by the emergence of cinema, artists like Marey, Duchamp and Muybridge initiated a series of works that changed the way we understand pictures and motion.

1.1.2 Art and Participation

In the early 20th century, the technical revolution devised by the likes of E.J. Marey, the formal revolution proposed by the likes of Picasso, the conceptual revolution by the likes of Duchamp [67], and the scientific models outlined by the likes of Norbert Weiner [79], shifted the role of art in contemporary culture, making it depart from the previous model, where the work of art was proposed as an untouchable monument of individual achievement. For Walter Benjamin [29], nothing is more revealing than the repercussions of the technology to reproduce works of art and the art of film: unlimited reproductions and moving pictures.

After establishing models of appreciation that were relative to the perception of the beholder, and works of art that were closer to social and psychological experiments than statements of beauty, contemporary artists have built an unresolved tension between what was traditionally understood as art, and what now constitutes art, a volatile term that stands for whatever a group of collectors and curators choose to promote. The seemingly libertarian force that has tried to bring a true experience of art to the masses has also become a substitute rating model, repeating the elitist tendency to shelter knowledge of art as a privilege of the few. Art is what is managed to be put in the gallery, and the valued skills of the artist have been displaced from craft and technique to politics, public relations, and management. This is the scenario where the digital arts were born.

1.1.3 The role of Digital Technology

Heavily abused by industrial media production, most digital resources have been crippled by the representation of tools that merely make non-digital process cheaper and faster. For example, the relationship between photographers and their craft has changed dramatically since the advent of digital photography, and so, therefore has the nature of photography. However, photographic images and the digital medium are not essential to each other, and the true strength of the digital medium lies elsewhere, separate from its use as an enhancement, or as a step forward in the evolution of all preceding media.

The interactive potential of the digital medium is infinitely richer than any other medium, making it possible to conceive pictures that can truly communicate with the beholder, or even with each other, providing an open canvas for the beholder to choreograph the picture as they understand it, transforming the process of appreciation from passive to active. Myron Krueger [49] used to say that interactivity should be raised to the level of an art form, as opposed to just making an art that happened to be interactive. Such

Interactive digital picture. Myron Krueger's Small Planet, SIGGRAPH 1993. In this work, participants pretend to fly over a computer generated, 3D planet. The childish gesture of holding one's arms out and leaning left or right and moving up or down controls the interface.

an idea is projected towards artists and their relationship with the beholder, challenging them to explore not only the space of the interactive picture, but also the space where this interaction might be delivered to the beholder.

At the same time, the emergence of distributed systems like the internet offer new kinds of remote communication between groups of humans and machines. A picture constructed by a digital medium can represent not only stored data or algorithmic processes, but live data retrieved from other human-machine interactions happening elsewhere. Even though this has been happening for a few decades already, the present scenario inherits from limitations imposed by the primitive architectures of pioneering digital systems, rarely exploring the potential of the medium fully.

1.2 The Studio as the space for process

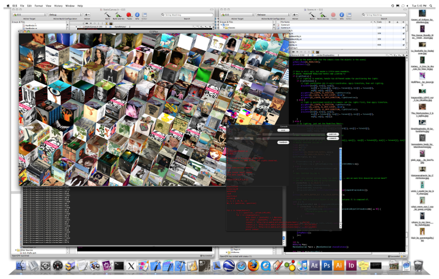
Luis Blackaller, spherical reflection of studio 8 in ISCP, New York, 2004.

Simply put, process is the space for change. A process can be understood as a sequence of transformations between two different states. In this sense, the artistic process turns experience into expression by delivering the work of art to the world. We can define the studio to be the space where the artistic process happens.

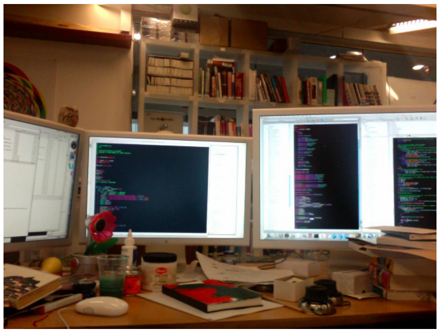
For the greek philosopher Heraclitus [43], we experience nothing more than process. His enigmatic river image: “We both step and do not step in the same rivers. We are and are not”, can be more simply understood as “The only thing that remains constant is change”. However, it is not my goal to claim that a work of art is an eternal process. After all, the work of art -it being itself a process or not- is what remains, or is born, after the process of artistic creation is complete. The work of art must be independent from the artist, and have a life of its own, regardless of whether it is conceived static or dynamic, or as an object, system or action.

1.2.1 Natural or Artificial

The digital artist experiences the space for process in two different domains, one physical, where the body gestures are captured as input to be transformed into data, and another one virtual, where the transformed input changes the state of the computer system. How much of this process is lost when only the virtual domain is observed? The separation of these two domains gives the artist a new space (at the boundary) to manufacture his own representations of process, the same way people create their own avatars or profile pictures in social networks based on how they choose to represent themselves instead than what they are. This is the reason why the digital medium is a perfect vehicle for the transmission of the spectacle. As it was simply put by Guy Debord [35]: "The spectacle is not a collection of images; it is a social relation between people that is mediated by images.", and: "Since the spectacle's task is to use various specialized mediations in order to show us a world that can no longer be directly grasped, it naturally elevates the sense of sight to the special preeminence once occupied by touch: the most abstract and easily deceived sense is the most readily adaptable to the generalized abstraction of present-day society". Process is transformation, and there is a difference between creating a performance of this transformation and performing the transformation itself. It is the difference between subject and representation, between the thing and its image, between natural and artificial.



Luis Blackaller, PLW Digital WorkSpace, 2008. Data from the computer's own internal process is transformed into an interactive audiovisual flatland, where the user plays the role of a god. Taking into consideration the user's input, the computer reorganizes this flatland at the speed of miliseconds, delivering an illusion of materiality. The representation appears as if it was the actual thing.



Luis Blackaller, PLW Physical WorkSpace, 2008.

However, information space is itself a reality, and the digital process is not an image of the physical process even if it is usually experienced as such, so this particular separation can occur the same way we normally separate two different subsets of a bigger thing, where code and pure data are perhaps the closest we can get to the reality of the digital medium.

1.2.2 Boundaries

The boundary is what determines a space. The boundary separates interior from exterior, and is always between the two. Simple notions of open and closed space can be understood in terms of boundaries, and these notions can be extended to loosely define public and private space:

1. A space is closed if it contains its boundary.
2. A space is open if it doesn't contain its boundary.
3. A space is public as much as it's open.
4. A space is private as much as it's closed.

It can be argued that a space only needs to not contain portions of its boundary for it to be open, like a room with a hole in a wall, but what's relevant here is to understand that the boundaries will define access and interaction politics on a given space, for instance, how much it is protected or shared.

Luis Blackaller, Lock for Black and White City, 2004. Eluding straightforward appreciations, definitions of boundary are possible at many different conceptual levels. A lock transforms a usually closed space into an open space for those who have a key.

1.3 Problems

In order to understand the relationship between the digital arts and contemporary culture, at least three fundamental problems need consideration:

1. The aesthetics, or appreciation of digital art.
2. The poetics, or process of creating digital art.
3. Accessibility and distribution, or how and where to experience digital art.

The first problem explores the following questions: What should digital art be? How should it be understood? In order to explore these questions the two other problems follow immediately. The second problem aims to answer the question of how digital art is made, and finally, the third one looks to answer the question of how and where to experience digital art, leading towards new problems between protected and public space that are raised when both spaces are artificial, which is the case in the digital medium. The design of artificial space is made by setting up boundaries to restrict or encourage access and interaction. Should the creative process of an artist be kept as a sacred secret that should happen in the intimacy of a closed space, or would it be of benefit to let an audience visit, interrupt and contribute to this process as it happens?

The third problem also has an economic aspect, that has to do with value and conservation of digital art. Conservation of digital art requires the conservation of the technologies that run it, making it far more difficult a task than it has been to conserve art in the past. It also becomes difficult to determine value when the art can be duplicated, or when value is placed in the process instead of the object. In terms of experience, the visual arts might move closer to the works of music and literature, where it is access to the content or participation in the performance what remains valuable, instead of the uniqueness of the art object.

As an effort to attack these broadly outlined problems from a reasonable perspective, I will frame the scope of this thesis to focus only on the expression and perception of art through interactive digital graphics, limiting the available materials to code and the resources provided by a personal computer connected to the internet.

1.4 Thesis Overview

Following a traditional ACG (Aesthetics and Computation Group) style, this thesis features the following sections:

1. Introduction: Motivation, definitions, problems.
2. Background: History, references, influences.
3. System: Foundation, methodology, design, implementation.
4. Analysis: Practical evaluations.
5. Conclusion: Results and future work.

In order to avoid breaking the continuity of the main text, most illustrations in this thesis have been incorporated as marginal notes.

1.5 Contribution

This work will help to explore digital media in its pure form, offering an alternative option to create and experience digital art, and contributing to validate code as a full featured mode of artistic expression that must find its rightful place in the contemporary art discourse, as one of the richest cultural constructs of the 20th century (code is the true new literacy that fuels new media). Finally, the development process will help me elaborate more precise ideas towards a personal poetics and aesthetics of digital art as interactive graphics.

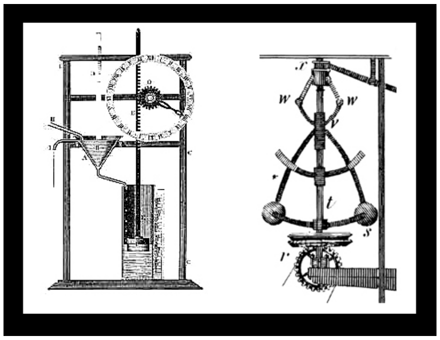
2

Background

2.1 Cybernetics

In 1948, MIT professor Norbert Weiner published a book called *Cybernetics, or control and communication in the animal and the machine* [79]. Inspired by decades of research collaboration with Mexican physician and physiologist Arturo Rosenblueth and other contributors, he articulated a set of principles to describe their then eccentric approach to science. My work in this thesis is an attempt to link the philosophy of Cybernetics with my perception of art and culture, using them to roughly localize how the ideas collected in Weiner's work migrated from science and engineering to other instances of culture, challenging the perception of authorship and art from hierarchy to system, from monument to experience, and from contemplation to interaction.

Weiner starts by describing systems that can be observed in live organisms, where state is regulated by feedback, using the resulting output as an input to produce further output. The first artificial automatic regulatory system, a water clock or clepsydra, was devised by the Greek inventor Ktesibios in Alexandria in the 3rd century BC. In his water clock, water flowed from a holding tank into a reservoir, and from the reservoir to the mechanism of the clock. The device used a cone shaped float to monitor the level of water in the reservoir, adjusting the flow accordingly to maintain the level of water constant, so that it could neither overflow nor could it run dry. It required no outside intervention between the feedback and the controls of the mechanism. Although he did not refer to this concept by the name of Cybernetics, Ktesibios is considered to be the first to study cybernetic principles.



Feedback. Ktesibios' clepsydra (left) and James Watt's centripetal Governor (right).

The next relevant improvement on a machine with corrective feedback dates from the late 1700s, when James Watt equipped his steam engine with a Governor, a centripetal feedback valve that controlled the speed of the engine. Alfred Russel Wallace identified this as the principle of evolution in his famous 1858 paper [78]. Ten years later, James Clerk Maxwell published a theoretical article on governors [53], one of the first to discuss and refine the principles of self-regulating devices.

But Weiner was not talking about specific mechanisms or generalizations of machinery. He envisioned principles that defined a philosophy of science, or an approach to scientific thinking that was substantially different than the preceding one. By focusing on behaviors rather than things, the question of "what is" was replaced by "what does it do". Weiner describes this different way of understanding systems and process as a revolution in science as important as Copernicus' model of the solar system, but he doesn't point out any scientific truth that would fail to be true when examined through the lens of Cybernetics. More than a revolution, Cybernetics seemed like an evolution. The displacement of causality from explanation to prediction (what will it do instead of what is) was a well known procedure in science since the late seventeenth century with the emergence of probability and statistical science [42], and Weiner's notes on Nonlinear Problems in Random Theory [81] show that he was very aware of this body of scientific knowledge, already mature when he graduated as a mathematician. However, he describes in detail a variety of problems that are still of concern to many popular branches of science. Artificial intelligence, neuroscience, decision theory, game theory, biology and psychology, are all key players in the events that have changed the relationship between art and technology through the 20th century and the first decade of the 21st.

Art and action. Pablo Picasso drawing with light.

In page seven of *Cybernetics*, Weiner talks about picking up a pencil. To do so, quoting from his words, "I have to move a few muscles, but I would hardly know what muscles I am moving, and even if I knew, I am not performing the action by consciously willing those muscles to move. What I do is just to pick the pencil up. I don't know why, but I can do the task". Sixty years later, when I turn my computer on and pick up my digital pencil to start brushing light on a picture with Photoshop (something almost everyone can do if the tool is given to them), it is impossible for me to know the constitutive elements of code that are running the task performed by the machine, or monitoring feedback through the input device and the event loop. I don't need to know. Digital computers, and user interfaces in particular, have been meant to follow the black box design metaphor described by Weiner in Cybernetics.

Performance. Jackson Pollock performing in his studio photographed by Hans Namuth, summer 1950.

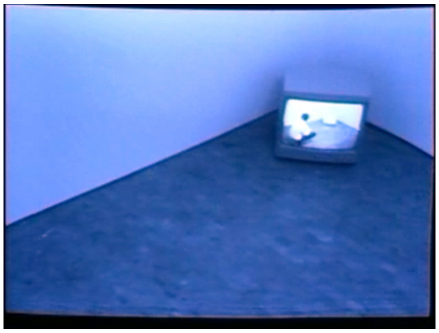
2.2 Performance, Process, Feedback

In 1949, the canvas was dripped and poured on by Jackson Pollock, slashed by Lucio Fontana, and punctured by Shozo Shimamoto. As curator Paul Schimmel explained [67], painterly action took precedence over the painted subject. This kind of gestural art got promoted to a point where other artists began to see the canvas as an unnecessary artifact, abandoning it to consider action itself as a form of art. Performance, understood as the activity of a unit intended to accomplish some desired result, became a powerful tool to communicate process, as well as to immerse the beholder into the process as an active participant. My interest, however, gears more towards the function of performance and process within the scope of the digital image, an image that remains in constant evolution. On the other hand, performance required a heavy use of interaction, and opened pathways to conceptual explorations that have later influenced creative movements in digital media, emphasizing the role of the beholder as an active participant. Inspired by the early 20th century experiments of Dada, artists that adhered to the Fluxus movement and the International Situationist began considering a kind of art receptive to change after responding to feedback from the beholder. Based on ideas originated by Marcel Duchamp and others, where the work of art is incomplete until the beholder gives it shape and meaning, the contemporary critic Timothy Druckrey declared [67]: "If images are to become increasingly experiential, then a theory of representation must be evolved to account for the transactions invoqued by participation". He is asking for an aesthetic of interaction, or how to appreciate interaction as an art form.

The fundamental limitation of Pollock's canvas, and any other, to accept feedback from the beholder and deliver interaction lies in its predetermined nature. Once the picture is finished, it will almost remain the same, changing over long periods of time that escape human perception. Only by putting Pollock himself in the hands of the beholder at the time of creation, interaction could inflict an effect over the art.

Cinema in art and entertainment, a sophisticated technocultural artifact that has succeeded in tricking the human eye to convey the illusion of motion, delivering a visual language to control the passage of time, is as predetermined to the beholder as a frozen painted canvas. The beholder can't change the course of a movie once it's set in motion. Cinema was meant to be contemplated passively, but radio, television and video were about to permanently change the relationship between art and the beholder.

Input. Security cameras were a favorite material of early video artists.



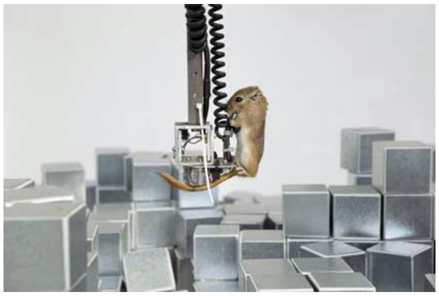
Feedback. Bruce Nauman, video surveillance piece. Public Room, Private Room, 1969 – 1970.

The video camera became an eye and memory to record the private activity of the artist in the studio, turning it into a picture of artistic personal process that could serve as a distorted mirror to look back at the beholder. Bruce Nauman and Vito Acconci avoided interaction with the beholder, and concentrated in the recording of the physical process of art making, suggesting to deliver the process as the art itself. Even though they were not willing to open the space of art to the participation of the beholder, they created a loop that offered an open ended form of art, where creation and appreciation began to merge. The beholder would not have any subject to lean onto, other than the process of making a subject that didn't seem to be there.

It didn't take long before art started pointing cameras at the beholder as well, using the beholder's actions as feedback to construct an art that could finally be completed by the participation of the beholder. Video technologies were crude and rudimentary, posing formal limitations that can't be overcome. When digital technology entered the scene, it offered tools for capturing and manipulating the beholder's actions in many other ways, making it possible to mold pictures' behaviors and interactions to the point where the picture can effectively engage itself in a conversation with the beholder.

2.3 Digital Art

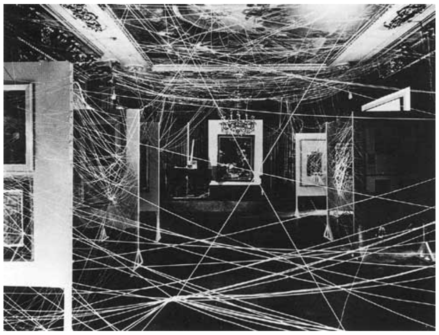
In the 1973 edition of his book *Art and the Future* [34], artist and art critic Douglas Davis declares the computer to be the ultimate creative tool for the hybrid artist, engineer, scientist, or designer, calling it the the final fusion. It was early in the history of the digital medium, and Davis did not pay much attention beyond its potential as a superpowerful tool. In his book, he explains how the computer is capable of creating images, sounds and simulations impossible to envision before, but he takes interaction and behaviors for granted, even when describing Nicholas Negroponte's and the Architecture Machine Group's Seek project from 1970. Seek, a feedback experiment composed of live Gerbils, metallic blocks and robotic arms, is not very close to the digital image I am discussing here, but embodies a spectacle with an interesting new component, obtained when embedding a community of living things within the feedback structure of the machine. If the plexiglass walls of Seek were painted black and people were made to interact with a disguised representation of some data measured from the internal behavior of Seek, a completely different statement could have been made about it.



Feedback and interaction. Nicholas Negroponte's Seek from 1970. Robotic arms reconfigure the arrangement of blocks after observing the gerbils' reactions to previous layouts.

Building aside of Davis' reasoning, the computer can be better understood as an abstract universal machine where the digital medium is controlled by the process outlined by code. In this model, the process itself becomes the content, immaterial, behavioral, interactive, and code is the new material. Because it happens over the passage of time through behavior and interaction, and because it can be reproduced with absolute precision every time the same code is run, digital content becomes the perfect example of what Walter Benjamin predicted 72 years ago in his paper about the work of art in the age of the Mechanical Reproduction [29].

The term “Digital art” covers a broad range of artistic practices that cannot be described with a unified set of aesthetics, and suggests the collapse of the boundaries between disciplines, merging all kinds of artistic media with aspects of science, technology and design [59]. Even though digital art was not fully accepted into the official institutions of art until the 1990s, and most of the revolutionary concepts embodied by digital art were already explored by theory and art visionaries in the first half of the 20th century, the interactive nature of digital art has been an important influence to reconsider the value of art as a system instead of an object, where the artist plays the role of a mediator or facilitator of the audiences’ interaction with the work. Furthermore, the virtual and network components of digital art are pushing towards a redefinition of the space occupied by a piece of art beyond the boundaries of any physical space, and even beyond the still dominant non relativistic conception of time.



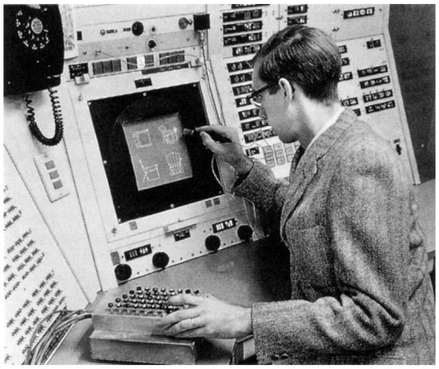
Marcel Duchamp, Mile of String, 1942, New York. The art contradicts the space that was traditionally supposed to frame it.

In his book *From Technological to Virtual art* [64], art historian and critic Frank Popper incorporates the term “virtual art” to identify an art where we can immerse ourselves completely into the picture and interact with it, experiencing not only reality but also a simulation of reality. The art he talks about is of a technological nature, but the digital component might not play more than a technical role, for example synchronizing the motion of several mechanisms based on the contents of a stream of data. Popper’s virtual art deals with much more than the scope of this thesis, which is constrained by code and the network enabled computer, but provides an interesting outline of the territories explored by early experiments using the computer and communication systems to produce art, and provides a conceptual framework that can help understand the studio of the artist as a space that can be experienced remotely.

Popper supports his theories about virtual art on the premise that it refines the relationship between artist and technology in a new way, where the aesthetic finalities of a virtual artist are linked with a number of other goals that appear to be of a more scientific or social order, but are in fact also concerned with basic aspects of the human condition. Popper suggests that these properties make virtual art (and digital art in consequence) a perfect candidate to humanize technology.

In the introduction of her book about Digital Art [59], Christiane Paul recognizes the label "new" as a fundamental element in the theoretical efforts to define the scope of digital art, once referred to as "computer art", and now considered a part of "new media art". The difficulties experienced by theorists, art critics and historians to reasonably characterize the different practices of art that are manifest through the digital medium are perhaps an evidence of the state of the medium itself. Flexible enough to permit the combined expression of any artistic form imaginable in an expanded interactive form, it makes sense that the first attempts to separate or identify this art have been through labeling it as "new" (it is actually new, although many other new things are not "it"), or describing it as determined by computation related concepts.

Computer Art. Vera Molnar, Comment faire sortir le carré de ses gonds? 1988. Vera Molnar thinks of the computer as an extension of her drawing process.



Ivan Sutherland's Sketchpad console, 1962. Sketchpad is operated with a light pen and a command button box. The four black knobs below the screen control position and scale of the picture.

2.3.1 Computer Art

When artists started approaching the computer to use it as a mode of expression in the 1960s, they tried to port their non-digital aesthetic ideas into the new medium. Interaction with the computer was crude, and did not permit for much physical communication with the machine until Ivan Sutherland came up with the invention of Sketchpad [74], the first interactive drawing program, that was followed by Douglas Engelbart's inventions of the bitmap and the mouse. Visual artists that did not have access to these revolutionary interfaces had to construct abstractions of their creative processes to translate them into rule based systems that the computer could understand. The transition was not difficult, because there were already non-digital experiments that approached the creation of abstract visual art as the result of a carefully designed formal analysis of composition and color. Vassily Kandinski, Paul Klee and Sol Lewitt are three examples that are worth mentioning. The early era of computer art was mostly dominated by a constructivist approach inherited from abstract and geometric schools of modern 20th century art.

Assistant executing Sol LeWitt's Wall Drawing No 65, first executed in 1971. National Gallery of Art, Washington, 2004. According to the principle of his work, LeWitt's wall drawings are conceived by the artist but usually executed by others.

For conceptual artist Sol Lewitt, when an artist uses a conceptual form of art, it means that all of the planning and decisions are made beforehand and the execution is a perfunctory affair. The idea becomes a machine that makes the art.

Telematics. Roy Ascott, Aspects of Gaia, 1989. Photo by Felix Nöbauer.

Computer art pioneer Vera Molnar, whose original artistic background comes exactly from these traditions of abstract, geometric and conceptual art, discovered the benefits of the computer in 1968. Her almost scientific attitude towards visual perception and cognition helped her embrace the computer with ease.

According to Molnar, the computer can serve four purposes. It first widens possibilities with an infinite array of forms and colors, and the construction of a virtual space. Second, it satisfies the desire for artistic innovation and lightens the burden of traditional forms. It can use randomness to rupture the systematic and the symmetrical. Third, it encourages the mind to re-conceptualize it's own aesthetic notions. Finally, Molnar's fourth purpose hints at the spirit of Cybernetics, suggesting that the computer can help the artist measure the physiological reactions of the beholder, tracking the motion of their eyes, for example, thereby bringing the creative process closer to it's effects [64], and bringing together certain characteristics of Dada, Surrealism, Fluxus, Happenings, and Pop Art with the science of cybernetics.

2.3.2 Communication Art

Frank Popper recognizes communication art as an important antecedent for several components of virtual art [64]. Today, media communications have been digitized, thus turning communication devices into nothing more than another color in the digital artist's multidisciplinary palette.

Telematics. Minitel, France, 1980s. Videotex was an early implementation of an "end-user information system". It was used to deliver information to users in a computer-like format, typically to be displayed on a television. The French Minitel system, unlike any other service, offered an entire custom designed terminal for free.

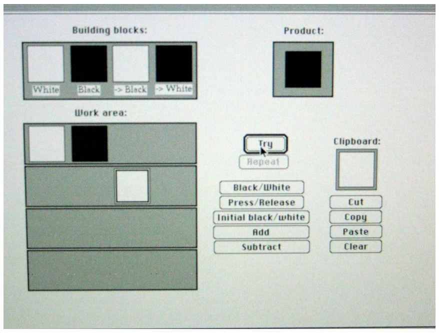
In Europe, artist Roy Ascott was one of the first practitioners of interactive computer art, and is internationally recognized as the pioneer of telematic art. Ten years before the personal computer came into existence, Ascott built a theoretical framework for approaching interactive artworks, dreaming of total participation from the beholder, and aiming to abolish the strict antinomy between action and contemplation. Although Ascott's concept of participation is primarily didactic in character, it is essentially cybernetic. Ascott's first intention was to initiate a creative behavior in the spectators by forcing a feedback relation with the responsive artwork, putting them in a position to handle ideas on their own, making them decide and react physically to the work. In 1987 Simon Nora and Alain Minc coined the term telematics [64] to describe the new electronic technology derived from the convergence of computers and communication systems. The process of "telematization" was first widely seen in the rapid growth in France since 1980 of the Minitel, a public videotex system that enabled widespread interaction between a network of users and database services.

At present, Ascott stands out as one of the most outstanding artists and theoreticians in the field of telematics. For him, the art of our time is one of system, process, participation and interaction. Based on multicultural and relativistic premises, and on the saturated, fast paced communication systems of our time, Ascott locates meaning and cultural value in the process of interaction between human beings. An artist could influence a significant audience only by leveraging telematic systems based on computer-mediated cable and satellite links. For Ascott, there are five features that define the art of our time, completely separating it from that of other eras:

- Connectivity between the parts and persons,
- Immersion into the whole, thus dissolving subject, frame and ground,
- Interaction as the very form of art (just as Myron Krueger used to say), transforming art from a behavior of forms to a form of behavior,
- Transformation (or process) as the perpetual flux of image, surface and identity,
- Emergence, a perpetual realization of ever-changing meaning.

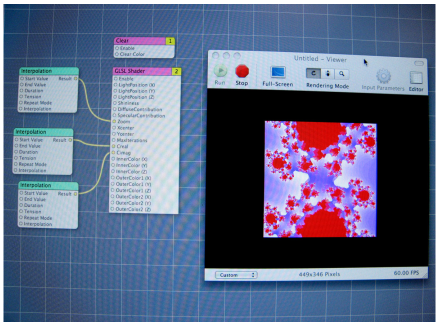
A careful examination of these features reveals that all of them, except connectivity, can easily be found in forms of art that don't need to be supported by technological artifacts, and are present in examples from Dada, Fluxus and "the Happening". It is important to understand that the key role of digital technology as a medium for the arts is connectivity, because it permits a virtual flow of ideas between people that otherwise would have remained unaware of each other, eliminating the natural separation imposed by space and matter, and challenging the traditional notions of public and private space. Whether this can be used as a tool to humanize technology as Popper optimistically suggests is still to be seen, as major tensions need to be resolved between these and other aspects of the digital medium that enable it to be an almost perfect instrument of control, like the ubiquity of surveillance and the media spectacle.

2.3.3 Code as a material

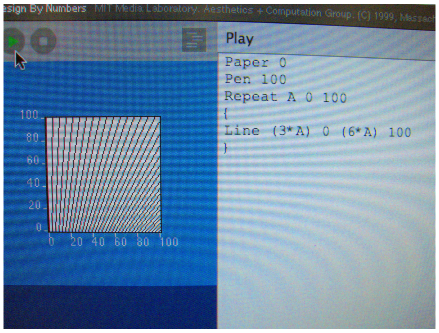


Dag Svanaes, Interactive Gestalt Editor, 1997. Svanaes devised a simple visual representation of Finite State Automata to study how Interaction Design skills could be developed visually.

The digital medium is not a tool for making “virtual” or “enhanced” models of architecture, painting and music. It embodies a universe for expression and communication that we are still far from understanding, where the rules of space, time, perception and memory are in our hands to mold. Programs as a medium are expressed through the difficult art of programming, which traditionally requires to master the craft of writing code. however, other ways have been explored. Dag Svanaes [75] is concerned with the problem of making computer programming accessible to artists and designers that are not trained as programmers, or people that lack the kind of rational thinking acquired after an education in science and engineering. He argues that visual and spatial thinkers might offer intuitions that could help choreograph interactions in better ways. Enough researchers and developers have shared Svanaes’ perspective through the last decade, influencing the creation of a family of programming environments where users are sheltered as much as possible from the intimidating streams of cryptic code. VVVV [24], Max MSP [10] and Quartz Composer [18] are some of the successful ones to note, that have been used recently to create interactive audiovisual installations by large communities of artists, designers, musicians and amateurs. The majority of these systems rely on hierarchical diagram representations of algorithms, data structures and feedback process. Experienced programmers can usually create their own nodes or customize others’ when specific behaviors are required. The elements of programming are there, but a deeper understanding of computer science is not required.


Clear
GLSL Shader
Interpolation
Interpolation
Interpolation
Untitled - Viewer
Run
Stop
Full-Screen
Rendering Mode
Input Parameters
Editor
Custom
449x346 Pixels
60.00 FPS

Apple's Quartz Composer in 2008. The Patches and connections in the left side are interactive visual representations of the data structures and instructions that draw the Julia Set GLSL shader on a Quad in the right side. Users can program sophisticated graphics without writing any code.


esign By Numbers MIT Media Laboratory, Aesthetics + Computation Group, (C) 1999, Massach
Play
100
80
60
40
20
0
0
20
40
60
80
100

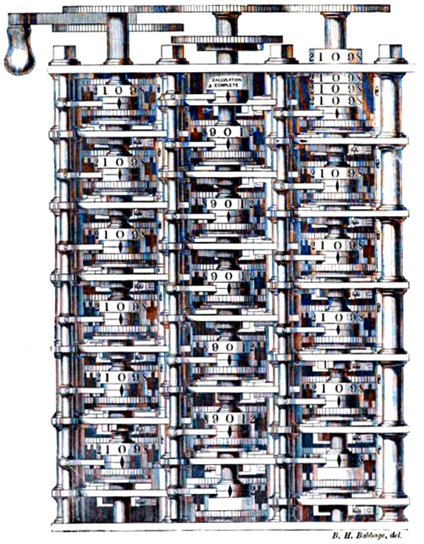
```
Paper 0
Pen 100
Repeat A 0 100
{
Line (3*A) 0 (6*A) 100
}
```

John Maeda, Design By Numbers, 1999. Design by Numbers is an extremely simplified programming environment for computer graphics. One of Maeda's intentions is to help understand the fundamental concepts of computer programming through the interaction with a code editor and its visual output.

However, as digital artist and MIT Media Lab alum Casey Reas pointed out [66], the differences in design and scope of computer languages can make them feel like different materials for the experienced programmer. This is the reason why educators interested in promoting the literacy of computer programming as an art form have kept developing code-based programming environments that simplify, or solve and hide, the harder aspects of coding, making it an easier skill to learn. From Seymour Papert's and Wally Feurzeig's Logo Turtle [56] [57] [58] in the seventies, to John Maeda's *Design by Numbers* [2] [52] in the late nineties, and more recently Reas+Fry's Java based *Processing* [66] [16] to mention a few, the scope of the ambition and the resources provided by these environments have increased to a point where the latter one, Processing, has been used to produce commercial level applications and content.

During a conversation I had with G.J. Sussman about the relevance of a literacy in computation and code, he pointed out to me that code is a material with a recipe. Code communicates a set of interactions as it carries out the process it was written to run, but it also communicates detailed instructions on how to make a similar system if it can be read back as code. The same thing can't be done with any other medium. Network artist and theorist Alexander Galloway agrees, for him "code is a language, but a very special kind of language. Code is the only language that is executable" [39].

But there is more to be said about the specifics of code as a material. When Reas compares differences between programming languages and materials extracted from nature -like wood or steel [66]-, his point is to illustrate the expressive limitations or advantages coming from choosing a particular language. He doesn't touch on the important fact that a programming language is already a cultural construction, representing a set of human idiosyncrasies that make it essentially different from the purity of a natural material. More generally, when making the choice to hide the code underneath a visual interface like Svanaes has suggested, the danger of not communicating the implicit cultural meaning of computer languages to the people using them, might prevent other cultural voices to contribute in shaping this new materials. Not addressing this issue might consolidate the digital medium as a colonizing agent, forcing artistic expression to fit forms that are predetermined by the requirements of a potentially alienating technology.



Difference Engine, Charles Babbage, plate from 1853. A difference engine is a special-purpose mechanical digital calculator, designed to tabulate polynomial functions. Since logarithmic and trigonometric functions can be approximated by polynomials, such a machine is more general than it appears at first. J.H. Müller, an engineer in the Hessian army, conceived the idea of a Difference Engine in a book published in 1786, but failed to find funding to progress this further. In 1822, Babbage proposed the use of such a machine in a paper to the Royal Astronomical Society entitled "Note on the application of machinery to the computation of very big mathematical tables". In 1837, Babbage began the design of the Analytical Engine, a more general version of the Difference Engine. Ada Lovelace Byron, one of the few people who fully understood Babbage's ideas, created a program for the Analytical Engine. Had the Analytical Engine ever actually been built, her program would have been able to calculate a sequence of Bernoulli numbers. Based on this work, Lovelace is now widely credited as the first computer programmer.

In a conversation with John Maeda, he suggested that I think deeper about Reas's comparison between programming languages and construction materials. After all, even though different computer languages can feel like different worlds, they are all high-level abstractions of variations originated from the universal computation models outlined by Alonzo Church, Alan Turing and John von Newmann half a century ago, and all they do is compute output after processing input. Regardless of them being compiled, interpreted, or designed to follow a functional, procedural, object oriented or any other abstract model, programming languages are just different wrappers around an underlying data-crunching cybernetic deterministic system, where symbols and numbers alike are represented by strings, which are nothing but arbitrary sequences of bytes used to represent locations in memory and ways to remember them (by naming them or pointing at them). Alexander Galloway carefully refers to "all code" in general when talking about programming, calling it "a language", without making a distinction between the levels of abstraction represented by different programming languages. In this case, would it make sense to think of code as the same unique material available, where programming languages would be different tools to mold or cut the process in easier or different ways, depending on the task at hand?

Another issue to consider when approaching code as a material has to do with vocabulary rather than with rules. Complex and sophisticated code can be written with a very simple language like Scheme, consisting of a small number of primitive types and operators, but complexity emerges after having to define and name many new composite datatypes, structures and operators that must become part of the vocabulary of anyone willing to use that code. When an artist has mastered a conventional material, like oil paint, the emergence of new colors or surfaces will not render their craft obsolete. Considering code as a material raises difficult problems that are inherent to the nature of writing instructions of control for rule-based systems, or understanding the principles that determine the behavior of a computer. When higher levels of abstraction are build on top and around each other, knowledge about how to call and connect the ready-made structures, objects, methods or other abstractions becomes a vocabulary issue, where there is no other way to preserve fluency than by absorbing hundreds of new words every month. In this contradictory scenario, where conceptual knowledge is still valid (principles remain roughly the same because the underlying model hasn't changed), but descriptions of the world are constantly replaced, it is the artist, not the material, who becomes obsolete.

2.3.4 Commenting Code

The main reason for the escalating evolution of higher-level programming languages is to find more natural ways for humans to express their needs that can still be translated to instructions a computer can follow. In spite of this effort, even highest-level written code can be difficult to read by anyone that didn't write it. At the same time, it is evident that the recipe described in the code will only be of any good if it can be well communicated to humans. Perhaps the syntactic obscurity of early programming languages like FORTRAN [28], ALGOL [55] [63] and LISP [54] inspired their creators to implement marker symbols to tell the compiler or the interpreter to ignore certain lines or blocks of code, so that programmers could use these lines to explain details about their coding strategies that the code might not make clear enough. Strings denoted by this markers are called “comments”, and are usually in the source code of a program. The syntax of the programming language specifies the form that comments can take. I haven't been able to trace back a history of the commenting practice in code, as it seems all references take comments for granted and give them little importance beyond questions of coding style. However, comments seem to be present since the early days of computer programming [82] [77].

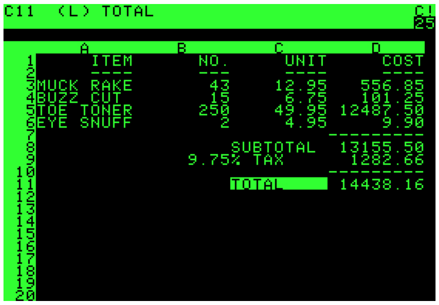
Word processors often support an option that enables writers to write notes to themselves about the material they are writing. These notes are not visible to a reader of the document unless explicitly asked for. This use of comments is also present when writing code, and programmers can use temporary comments to remember things to do or possible ideas, as well as to keep track where a problem might not have been solved in the best way.

Programming literature classics by Kerninghan, Ritchie, Plauger, Pike and Knuth [45] [46] [47] [48] all refer to comments as explanatory aids for comprehension. It is Kerninghan and Pike, in *The Practice of Programming* [46], who dedicate more than a couple of lines to describe in detail a sort of technical guide for commenting code. Their guide includes around five pages of example cases that illustrate what they say in the first few lines of the comment section in their book: “Comments are meant to help the reader of a program. They do not help by saying things the code already plainly says, or by contradicting the code, or by distracting the reader with elaborate typographical displays”. Other authors, like Abelson and Sussman [73] believe that descriptive names should be enough to make a program explain itself; In their view, well written code should self-comment itself. Both approaches to comments are mostly pragmatic, and discard the more free -and potentially playful- annotation style where a programmer could be writing about their creative process in a more literary manner, totally “distracting” the reader. The structure of code is very strict, and the only space it has for natural human expression is between comment markers. Open source applications, for example, often carry contact details, licensing and marketing information about the vendor inside of comments, making them useful to carry meaning that is not part of the code’s intent, even if related to it.

2.3.5 Programs as tools

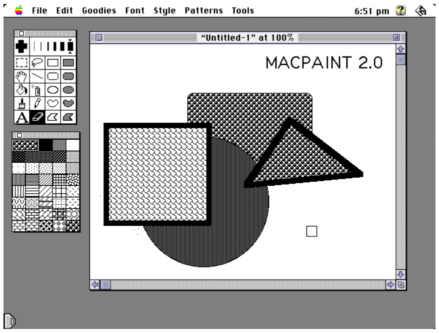
Not very long after Sutherland’s and Engelbart’s pioneering research on Graphical User Interfaces [74] [37] became an important reference in Application design, the emergence of the personal computer during the 1980s promoted an explosion of commercial products that changed how computers were understood by public perception, turning them from high end computational tools for scientists and engineers to readymade solutions for everyday problems. In 1978, Daniel Bricklin and Bob Frankston created VisiCalc, the first spreadsheet visual calculator, for the Apple II personal computer, that moved from being considered a hobbyist’s toy to a much-desired, useful financial tool for business. Other efforts quickly followed, and soon enough all personal computers were entering the home equipped with enhanced virtual versions of anything you can think of that could combine typewriters, calculators and sketchpads in the same place, merging the creation and manipulation of text, statistics and pictures into the same multimedia experience. Programs like MacPaint, poorly simulating the crafts of drawing and image manipulation, were set as a starting point to the evolution of industry standards for the manipulation of visual media like Adobe Photoshop and Adobe Illustrator, that now expand -instead of just simulate- the skills and tools required for creative imaging. However, mostly because of practical reasons , users were kept completely unaware of the sophisticated computational processes running their interactions with the imaging machines, and the possibility to directly manipulate computation through custom algorithmic process was kept a privilege of the few technically savvy, usually with a strong scientific or technical background.

In the early 1990s, researchers like Michael Eisenberg were aware of this divergence, and based in a historical tradition represented by the likes of Leonardo da Vinci and M. C. Escher, whose imaging has been heavily influenced by mathematical thinking, Eisenberg considered exploring the possibility of an interface that could -if not merge- at least pair together procedural graphics and visual digital tools. SchemePaint [36] is a program that combines a Scheme interpreter [73] with a drawing canvas where images can be loaded, doodles can be made, and every element in the canvas can be manipulated through simple Logo-style Scheme turtle scripts. Even though Eisenberg didn't find much use for SchemePaint outside of the academic sphere, many commercial applications started incorporating embedded interpreters to generate data or control mechanisms through the evaluation of scripts.Today, an interpreter is considered to be a required component in most leading commercial imaging packages, especially if they deal with the manipulation of graphics in motion, and some of them have incorporated support



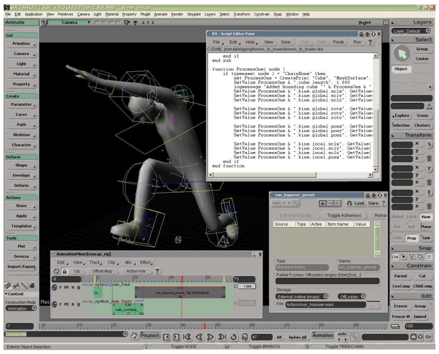
```
C11  (L) TOTAL                                C!
                                              25
           A            B          C          D
 1        ITEM         NO.       UNIT       COST
 2        ----         ---       ----       ----
 3MUCK RAKE             43      12.95     556.85
 4BUZZ CUT              15       6.75     101.25
 5TOE TONER            250      49.95   12487.50
 6EYE SNUFF              2       4.95       9.90
 7                                    ----------
 8                        SUBTOTAL      13155.50
 9                   9.75% TAX           1282.66
10                                    ----------
11                        TOTAL         14438.16
12
13
14
15
16
17
18
19
20
```

VisiCalc, 1978. VisiCalc was the first spreadsheet program available for personal computers. It may well be the application that turned the microcomputer from a hobby for computer enthusiasts into a serious business tool.



MacPaint 1.0, 1983. MacPaint 1.0 was written by Bill Atkinson, a member of Apple's Macintosh development team. MacPaint was one of the first commercial GUI-based bitmap editing applications, and became a required reference by which similar efforts were measured.

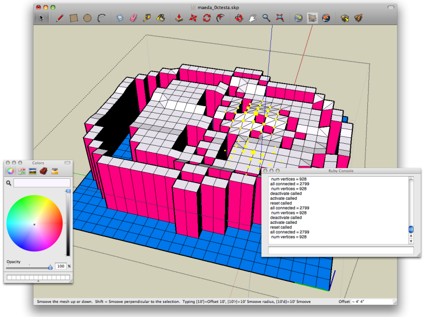
for several languages, adding interpreters for popular Open Source languages like Perl and Python as an extra option to their own proprietary languages.



C:\Softi[...]nimate\rigging\bones_to_boxes\bones_to_boxes.vbs

```
    end if
end sub

function ProcessOne( node )
    if typename( node ) = "ChainBone" then
        set ProcessOne = CreatePrim( "Cube", "MeshSurface",
        SetValue ProcessOne & ".cube.length", 1.000
        logmessage "Added bounding cube '" & ProcessOne & "
        SetValue ProcessOne & ".kine.global.sclx", GetValue
        SetValue ProcessOne & ".kine.global.scly", GetValue
        SetValue ProcessOne & ".kine.global.sclz", GetValue

        SetValue ProcessOne & ".kine.global.rotx", GetValue
        SetValue ProcessOne & ".kine.global.roty", GetValue
        SetValue ProcessOne & ".kine.global.rotz", GetValue

        SetValue ProcessOne & ".kine.global.posx", GetValue
        SetValue ProcessOne & ".kine.global.posy", GetValue
        SetValue ProcessOne & ".kine.global.posz", GetValue

        SetValue ProcessOne & ".kine.local.sclx", GetValue(
        SetValue ProcessOne & ".kine.local.scly", GetValue(
        SetValue ProcessOne & ".kine.local.sclz", GetValue(
        SetValue ProcessOne & ".kine.local.posx", GetValue(
    end if
end function
```

Embedded Python Interpreter in Softimage XSI. AliasWavefront Maya and Softimage XSI, industry leading 3D animation programs for visual effects, combine hands-on manipulation of graphical components with procedural graphics using several embedded interpreters, supporting their own proprietary languages along with popular Open Source scripting languages like Perl and Python.



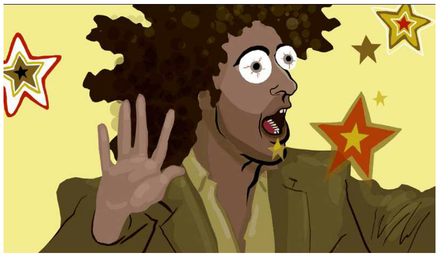
```
 num vertices = 928
all connected = 2799
 num vertices = 928
deactivate called
activate called
reset called
all connected = 2799
 num vertices = 928
deactivate called
activate called
reset called
all connected = 2799
 num vertices = 928
```

Embedded Ruby Interpreter in Google Sketchup. The Sketchup Ruby API provides ways to expand the application's functionality.

Prepackaged software tools have the disadvantage that they evolve slowly and are designed to follow industry trends that often discard experimenting with possibilities that are not immediately profitable. Media Lab alum and filmmaker Bob Sabiston [68] [6] started experimenting with alternative tools to create computer animations. His master thesis deals with a system that tries to take advantage of the traditional animator's know-how and skills to help him manipulate virtual puppets in 3D space through an interface of drawing gestures. It didn't get him anywhere. Instead, he found success with a much simpler application where the computer is left to perform a straightforward task, and the heavy work is left for the human to deal with. Being himself an animator, Sabiston enjoyed creating programs he could use to produce content, and he believed he could use digital technology to simplify the painstaking task of producing commercial level animations. Based in a traditional animation technique named rotoscoping, that consists on tracing over frames of real-life footage to reproduce complicated motion, and aware of the speeding progress digital video was experiencing through the 1990s, he created a simple program to let the animator trace over an initial frame of digital video, and then deform his drawing to fit the subsequent stages of motion. He called this technique interpolated rotoscoping, as opposed to traditional rotoscoping, and he called his program *Rotoshop*. Sabiston's approach differs from traditional animation techniques by deforming a single first drawing instead of placing many drawings in sequence, like traditional rotoscoping does.

Sabiston's visionary artistic practice did not stop there. He was aware that, as much as he could simplify the process of making animated movies by tracing over key frames of digital video, the task of producing a complex full-featured movie would still require a small battalion of artists crunching drawings through the footage. He needed to figure out how to build a community around his tool and he managed to do it.

The one thing he didn't do, however, was to look back at the source of his tool as a material for other programmers to mess with, perhaps protecting his well deserved niche in the market from being duplicated by a hundred other studios everywhere in the world



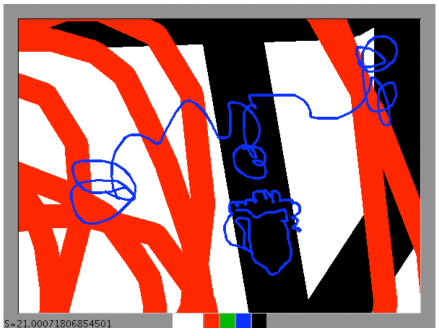
Frame from Waking Life, 2001. Directed by Richard Linklater. Art and Animation direction by Bob Sabiston. Waking life was made using Sabiston's Rotoshop, and became an unusually popular full-featured experimental animated film, where the creator of the tool was also directly involved in the creation of the content.

Bob Sabiston, InchWorm, an unreleased version of Rotoshop for the NintendoDS game system, 2007.

Andrew Deck, an artist who is well known for pioneering research on the creative possibilities of the Internet as a medium, was aware of the problems and limitations posed by the imposition of proprietary software. Deck has declared he makes "public art for the Internet" [64]. He believed that no program could be "public" unless its source code was available to the "public". With his project "Openstudio", presented in the Open Source Lounge exhibition in Athens in 2000, he explored three fundamental possibilities offered by the Internet as a medium:

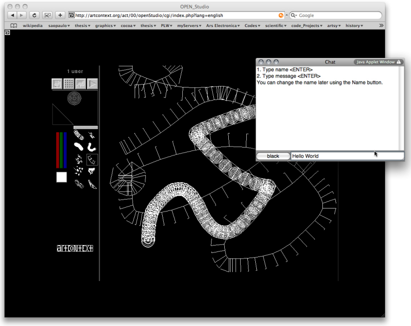
- Public access to the source code for modification.
- Multiple access to the tool for collaboration.
- Embedded communication within the tool.

With Deck's Openstudio it was possible to add new features to the program, draw together in the same place with someone else anywhere in the world, and coordinate the drawing activity through an embedded chat interface. However, the conceptual framework to understand such a program is still "the tool"; Deck's main concern seems to open a discussion about what digital tools could become, given the technologies we have. He was right, the emergence of commercial online studio suites is today a reality, but looking at programs as tools instead of interactive hubs to enable the creation of art as participation is not prevalent in his work.


S=21.00071806854501

Ken Perlin's Whiteboard ZoomPad. The gray frame is used as a navigation interface that allows users to infinitely zoom into their drawing, hinting on what a digital canvas can become, source code available [13].

Oneline, by John Maeda, a possible influence to Deck's Openstudio that fits better in the subsection about communication and participation, but is worth discussing in contrast to Deck's tool based work, was created in 1999 as an experiment to explore the collaborative potential of the Internet space. Maeda carefully deconstructed any tool reference his program could have by proposing a ultimately useless form that could not have any commercial application. He made a space where visitors could continue drawing over the end of a line, suggesting the collective authorship and the dynamic nature of artistic practice within the networked digital space.


OPEN_Studio
http://artcontext.org/act/00/openStudio/cgi/index.php?lang=english
Google
wikipedia
saopaulo
thesis
graphics
cocoa
thesis
PLW
myServers
Ars Electronica
Codes
scientific
code_Projects
artsy
history
1 user
Chat
Java Applet Window
black

1. Type name <ENTER>
2. Type message <ENTER>
You can change the name later using the Name button.

Hello World

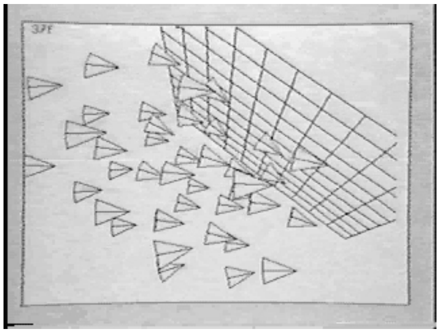
Andrew Deck's Openstudio, 2000. Open Source, networked collaboration and embedded communication channels challenged how we understood digital tools.

2.3.6 Programs as art

A program can be used as a tool to make art, or it can become art itself. In such a case, a program exemplifies with clarity the recent transformation of art from object into system, because the program determines how the art will be experienced, even in cases where it is not interactive. In 1986, computer-graphics animator Craig Reynolds got interested in simulating the flocking behavior of birds. Even though ornithology couldn't help him with a clear enough explanation of how the social complexities of the flocking behavior can be modeled, Reynolds believed he could write a computer program that could simulate this behavior correctly, even if he didn't understand the full dynamics involved in the natural phenomenon. Not surprisingly, he chose an approach that fits the Cybernetics principles outlined by Weiner [79]. Even though the behavior of a flock seems to be coordinated by a centralized control, Reynolds speculated that it must emerge from the independent actions of each bird. All evidence suggested him that flock motion had to be merely the aggregate result of the actions of each bird, acting solely on the basis of its local perception of the world. Reynolds defined a simple set of three rules in terms of the opposing forces of collision avoidance and the urge to join the flock:

- Flock centering (try to stay close to nearby flock mates),
- Collision avoidance (avoid collision with nearby flock mates),
- Velocity matching (try to match the velocity of nearby flock mates).

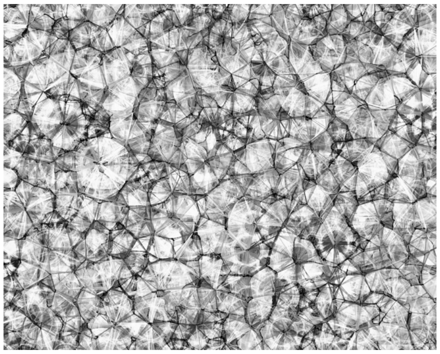
The technique developed by Reynolds succeeded, not only helping promote the computer graphics revolution that was happening at the time, but inspiring ornithologists to approach their scientific problems from a different angle.



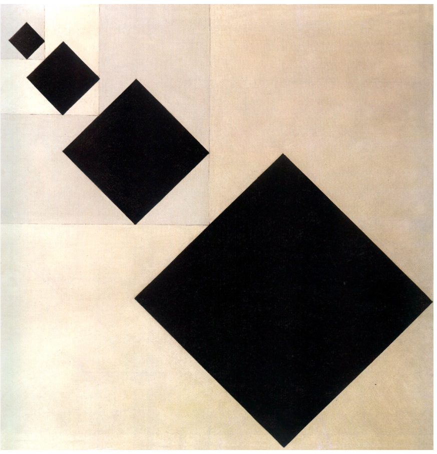
Craig Reynolds, Boids algorithm, 1986. Based on three very simple proximity rules, Reynolds figured out how to model the behavior of a flock by concentrating on the local perception of each individual.

The relationship between biological sciences and the digital arts has constituted a feedback loop due to the heavy influence of information theories and computer science over biology, making it natural for computer scientists to think in terms of biological concepts and for biologists to understand their science in terms of information systems. The very idea of life, intriguing to most aspects of human knowledge, became the subject of representation models that conceive us as information systems, and there has been a growing awareness that programs evolve through process, tracing information footprints that can lead to new forms of understanding the world.

The idea of rule based systems -or programs- as art has already been explored outside the digital domain by some expressions of Dada, and more explicitly by the conceptual wall drawing series started by Sol LeWitt in the late 1960s, where he defined systems of instructions in natural language that anyone else could later



Casey Reas, Process 7, 2005. As Reas describes it, Process 7 is a text that defines a process and a software interpretation of the text. The text is a set of simple rules that will set the system in motion.



Theo van Doesburg, Arithmetic Composition, 1930. Van Doesburg saw in these paintings his ideal in painting: a complete abstraction of reality. In his *Manifesto of Concrete Art*, 1930, van Doesburg suggests that this form of abstractionism must be free of any symbolic association with reality, arguing that lines and colors are concrete by themselves.

perform without his intervention. The instructions in the program entirely defined the outcome of the art each time it was performed, even if each wall drawing would slightly change depending on the site and the people that performed it. The interpretive influence of the performer on the outcome of the art would act as a translator to overcome the ambiguities of natural language when preforming the process outlined by the program. Digital Artist Casey Reas, interested in the differences between process performed by the machine instead of the human, points out how the instructions have to be programmed in a very precise manner in order for them to be executed by a computer, as different from the more ambiguous character of the instructions that a human can execute. This comparison, he says, is made between the programmer and the entity of execution, human or machine [65]. A contradictory point of tension rises from this comparison, making it hard to reconcile LeWitt's conceptual mysticism with artistic expression through digital programming. The first three statements in LeWitt's 1969 text *Sentences on Conceptual Art* [31] read:

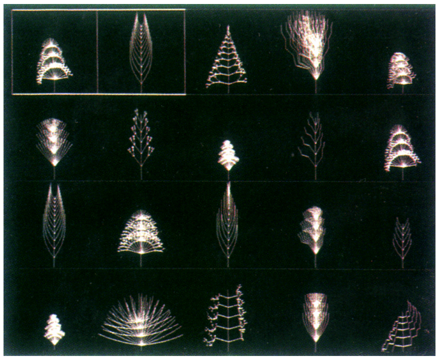
- Conceptual Artists are mystics rather than rationalists. They leap to conclusions that logic cannot reach.
- Rational judgements repeat rational judgements.
- Irrational judgements lead to new experience.

Whether or not this collection of sentences defines a methodology of art that should be taken seriously, or why would he think a reaction against reason should be the best way to outline the first principles of his aesthetic practice, LeWitt emphasizes these sentiments in his writings by consistently using the terms “illogical” and “intuitive”. Reas acknowledges that these terms easily categorize human thoughts, but they cannot be as easily used to define software. As a computer programmer, Reas understands that the undefined mental space where the conceptual artist usually works will not materialize in a computer program until it is made to fit within the constraints imposed by digital technologies. This raises a fundamental question about the artistic process and digital programming, separating it from other artistic practice where random impulse will always lead to new, unexpected results, because when programs are not written exactly right, they will just fail to compile.

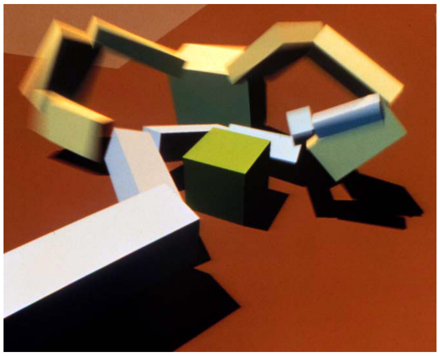
It might be that Reas' conceptual approach takes a departure from the conventional use of digital media to simulate, or design artificial representations of natural things. His point of view might be related to the spirit of Concrete Art from the 1930s, where lines and colors are considered concrete by themselves. Reas looks at code as an extension of nature, performing computations that are themselves -as computations- the subject of his art.

Karl Sims, a media lab alum with a MIT biology degree, looked deeper into the space where Craig Reynolds got his inspiration when he created the Boids algorithms. Trained as a computer graphics animator with a Hollywood experience, Sims decided to use biological evolution as a source of inspiration for his work.

"Simulation" and "Interaction" are key elements in the language of the digital medium (simulation mesmerizes, interaction activates), and Sims made an interesting use of both of them in the early 1990s, suggesting new ways of interaction between digital art and the beholder. Because of his evolutionary perspective, Sims designed simulations that could compete and reproduce, looking to see how the initial conditions in his systems could spawn unexpected artificial life forms [70] [71] [72]. However, his programs depart dramatically from the simplicity illustrated by the systems created by Reynolds and Reas, where simple rules of interaction between components led to complex system behaviors. The systems developed by Sims simulated of 3 dimensional physical environments and complex definitions of the creatures' bodies, that had to take in mind sophisticated components like physical locomotive systems, sensing systems and brains. When looking at his animations, where simple creatures made of cubic shapes swim around in an almost empty world, it's impossible to recognize what happens at the program level. The program that simulates the creatures, makes them compete, and chooses which ones should be kept around.



Karl Sims, Artificial Life, interactive plant evolution, 1990. Sims created a program to mate digital simulations of plans, so that the combined genetic information would be mixed with a mutation factor to create a new generation of plants. Natural selection was replaced by the aesthetic appreciation of a human user, who would pick out which plants would "survive" to reproduce again.



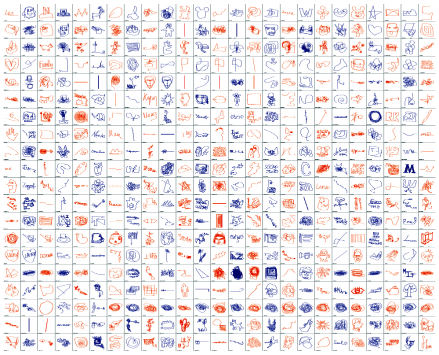
Karl Sims, Evolved Virtual Creatures, simulation 1994. A population of several hundred creatures is created with a supercomputer, and each creature is tested for their ability to perform a given task, such as the ability to swim in a simulated water environment or fight over a cube. Those that are most successful survive.

This idea of an art that evolves led Sims to experiment with a system where the evolution was controlled by a human user, substituting the process of natural selection with one of aesthetic appreciation. In this system, Sims would run a plant generation algorithm, and choose two plants to reproduce. Once another group of plants were born and grown, Sims would choose the ones he liked or found visually interesting to survive, discarding all the rest. The idea of an art that is receptive to the perception of the beholder and can evolve towards a form that is more aesthetically pleasing raises a few questions about the role of art, and whether it remains to be art, if its only goal will be to please the beholder. However, the ability of art to change in response to the beholder's gaze can be channeled in many ways, making this ability a powerful tool to communicate art.

Ten years after Sims worked on his interactive evolution experiments, we experience a World Wide Web full of social rating systems that push digital content up or down the media spectacle based on the preference of the virtual masses, in a system where creative humans perhaps play the role of the competitive creatures in Sims' simulations, fighting for attention instead of survival, and willing to change their appearance in order to gain acceptance.

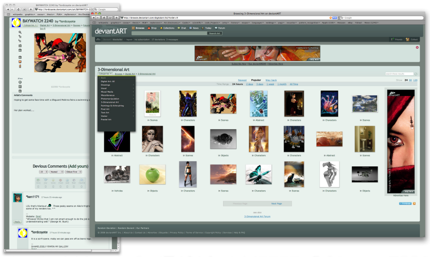
2.3.7 Participation and Communication

I have already mentioned John Maeda's OneLine [51] as one of the first online participation systems that use networked systems to mediate the creation and distribution of art and entertainment by large online communities. Historically, networked environments have been primarily used as tools for administration and management, and it was not evident that the networked environments themselves could be looked at as creative social systems, where the content becomes not only what a community of artists creates, but also the interactions between them, and with their audiences, all whom usually belong to the same community. This gave rise to a new kind of production model where the separation between producer and consumer blurred, leading to an influx of optimistic literature that saw this as a path towards a new utopia. Even if every single human in the world had the resources to spend the rest of their productive lives online I find that very hard to believe, and I feel more comfortable joining a critical discourse that examines this new space for trade and communication as another way to discuss the tension built around expression and control.

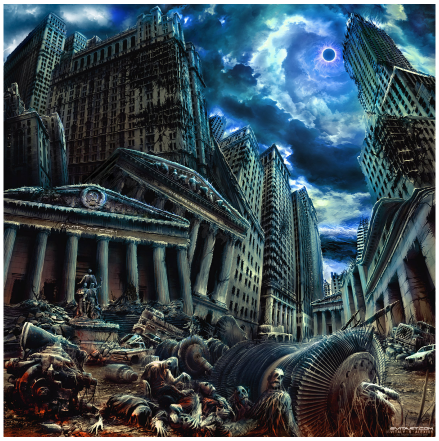


John Maeda OneLine Project, 1999. The One Line Project would take lines drawn by people using a networked drawing tool and connect all of those lines end-to-end. Maeda was wondering if it were possible to realize a line that was as long as the perimeter of the Earth. In the computer, however, any line can scale to any size.

At first sight, opening new channels for expression and communication seems like a good thing, but several problems arise from these conditions. First of all, when everyone is encouraged to participate in the creation of content and given equal opportunity to distribute it, content loses its sacred character and becomes trivial. Apart from it being good or bad, this type of participation could work both ways, helping people live more creative lives and communicate their feelings and ideas better, or it could lead to a collective ignorance that would not let an important message get across, because the only measure for importance would be audience ratings, which are already biased by statistical homogeneity.



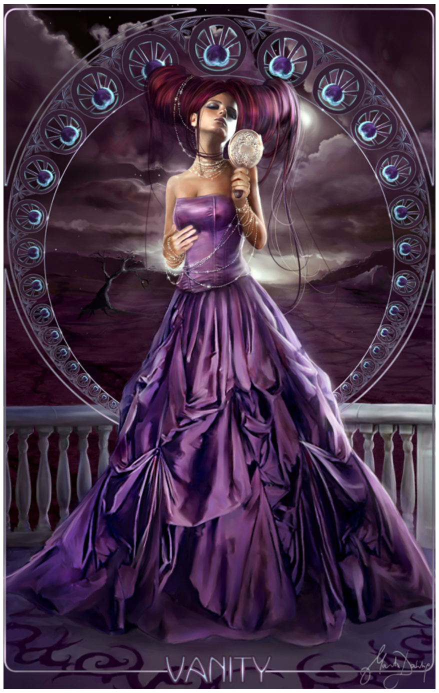
DeviantArt is a social network built around the creation and appreciation of pictures. Users give each other feedback about their pictures and rate them, and a service is included to order prints.



Deviant art, most popular picture in the digital art category on April 13th 2008, Aeons of Eclipse by Vitaly S. Alexius. It is evident that this picture, other than being manipulated in a straightforward manner with digital tools, has nothing inherently digital about it.

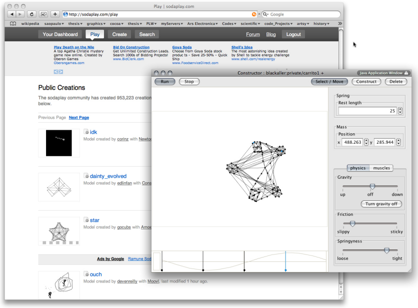
Over the last ten years, a large number of online communities have been harvested around the creation and appreciation of all kinds of content, making digital media -embodied by the personal computer and the internet- the perfect medium for almost anything, and turning “collaboration” and “participation” into the new gold of the first decade of the 21st century. It is not relevant for the scope of this thesis to elaborate in detail about what has happened, or to deliver precise descriptions and comparisons of the multitudes of social media services that exist today, so I will concentrate on briefly analyzing the ones I find relevant to the construction -or distortion- of the idea of digital art, and the creation and appreciation of it.

Evidently, a fast and easy way to build a community that shares and trades on culture would be to channel old media; everybody is familiar with it and a lot of people practice it as amateurs. This is the case of illustration and most importantly photography, and it is relevant to mention that the digital medium -in its role as a tool- has completely blurred the distinction between the two. Sites like deviantart [3] and flickr [7] are a couple of early examples that have experienced extraordinary growth. Other services like youtube [26] followed, as the technology became more robust and allowed for the distribution of more complex content like digital video. Still, all that these websites had in common was that the creative process to produce their content happened elsewhere, and they would be nothing more than publishing channels, where everything posted in them was at a more or less finished stage. Rhizome [19] and the Processing website [16], have followed this gallery model regarding art especially produced with digital technologies or code. By taking advantage of the Processing programming environment that can compile programs as java applets, the Processing website succeeds in delivering interactive digital content together with the source code, that can be downloaded by the users to change and recompile in their computers. As I already mentioned, these online spaces have to be understood as just galleries, where the work already made is displayed, and feedback about it collected. A lot of the examples I have mentioned have been experiencing dramatic changes recently, incorporating features to compete with the other more experimental online spaces for social creativity, where the content is created in them. Flickr for example now features a mini-image editor in the spirit of Adobe Photoshop, that lets users manipulate their uploaded pictures within Flickr itself. DeviantArt is an interesting case in terms of the nature of the content it displays. DeviantArt represents a social idea of art that is closer to popular culture and a naive perception of the traditional arts than the contemporary notions of art discussed in the majority of this thesis' background. The divergence between high culture and popular culture is nothing new, but an important issue to consider. DeviantArt divides "digital art" between eleven different categories: drawing, vexel, mixed media, miscellaneous, photomanipulation, 3D art, painting and airbrush, pixel art, text art, vector and fractal art. Aside from these categories being quite arbitrary, none of them has much to do with the digital art discussed by academics, critics and curators. Rather than figuring out how the digital medium is modifying how we represent reality, digital artists in DeviantArt are more concerned about how to color a comic book page or how to make a realistic depiction of a fairy.



Deviant art, digital art category most popular picture of all time until May 1st 2008, The Seven Deadly Sins: VANITY by Marta Dahlig. This picture, just as the previous one, is not inherently digital, but it can be argued that the digital medium has empowered people, that otherwise would have remained passive, to produce and self promote their own artwork. "Fan Art" became this way an important new genre in popular culture.

Networked services like version control and messaging have been embedded in commercial applications for a while. Softimage XSI features a technology called Delta referencing, a lightweight referencing system that allows production teams to store 3D assets, and the changes made to those assets, in external files that can be assembled dynamically to produce characters, props and environments. Artists can modify assets and then update them non-destructively in a collaborative workflow. Adobe Version Cue is a server-based file-management system included with the Adobe Creative Suite 3 software. Version Cue CS3 can centrally manage shared project files, coordinate parallel group work using an intuitive version control system, track file status with comments, and host Adobe PDF reviews. However, these communication services should not be considered as part of the social media, because they are constrained by industry standards and pipelines, aiming to foster efficiency instead of creativity, where collaboration is a management strategy to get the creative work reviewed and approved faster.



Ed Burton, Soda Constructor, 2000. SodaPlay is an online collaborative learning environment for the Soda Constructor. Participants build and share mechanical models in a simplified physics simulation.

In the year 2000, Ed Burton launched a social networking website for his Soda Constructor Application called SodaPlay [21]. The Soda Constructor consists of a simple 2 dimensional physics engine and a simple set of tools to create and edit spring-mass structures.

Control over gravity, friction and spring tension delivered an experimental mechanical simulator, where an appropriately built structure would start walking or rolling away. SodaPlay, the social networking website, facilitated functionality to let users share their knowledge and their models, creating a community of Soda Constructor enthusiasts that didn't take long to populate the web with a zoo of 2-dimensional creatures competing against each other.

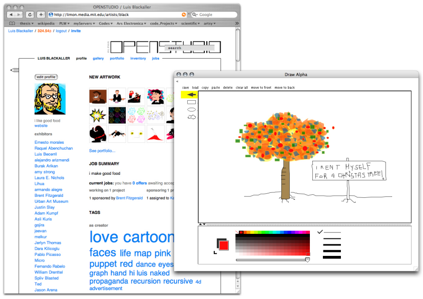
In 2004, Blizzard entertainment launched World of WarCraft, the most popular massively multiplayer online role-playing(MMORPG) game of all time. These kind of games raised a lot of interest because their internal social rules produced virtual economic activities that started merging with the real world, blurring the distinction between virtual and real value.

In a game like World of WarCraft, players personify roles within a certain narrative, and the performance of these roles gives them access to virtual goods that they can trade later or lose against others. An OpenStudio [15] that is not directly related with Andrew Deck's project from 2000 was launched in 2005 by the Physical Language Workshop in the MIT Media Lab as a collaborative online art-based economy. The premises were fairly simple. When creating an account, the new artist would receive a fixed amount of virtual money, access to buy from the previously created artwork, an embedded tool to edit the purchased artwork or create his own from scratch, and a gallery space to put his art for sale.

Dunmore
Damador
<Night Crew>

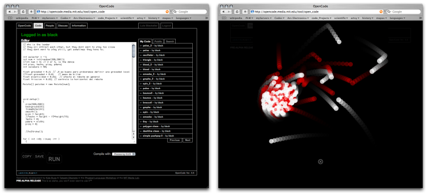
World of WarCraft, Blizzard entertainment, 2004. World of WarCraft is the world's most popular massively multiplayer online role-playing game, where a large number of players interact with one another in a virtual world. Many MMORPGs feature living economies, as virtual items and currency have to be gained through play and have definite value for players.

Virtual space in OpenStudio was far from public, and was the subject of very strict mechanisms of control that emulated the market dynamics in today's capitalist creative industries. Artwork was a secondary excuse not relevant as itself, and existed only because it was the only way for members of the community to communicate with each other. In less than a few weeks artwork was used to submit policy requests to the developers of the website, or advertise artwork sales.

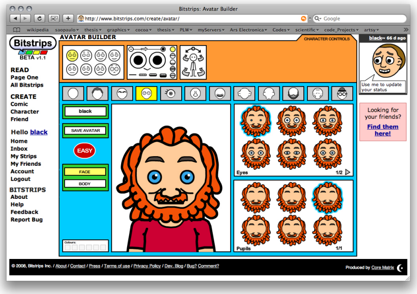


PLW Openstudio, MIT Media Lab, 2005. Openstudio combines a simple drawing tool with an online virtual economy of artists and collectors.

OpenCode [11] was created by Takashi Okamoto and Kyle Matthew Buza in the Physical Language Workshop right after OpenStudio. They had two goals in mind. One was to port the graphics programming experience to the realm of the web browser the same way OpenStudio did with drawings, where the code could be written and run after a single click without having to download any software, or save and compile any files. Based on the Processing programming environment, OpenCode defined the social interaction between the programmers that joined with a one simple rule: In order to show the resulting interactive graphics application, the programmer would have to disclose his code, making it accessible to everyone else.



Kyle Matthew Buza and Takashi Okamoto, OpenCode, 2006. OpenCode ports the entire graphics programming experience to the web browser. When the programmer clicks on “run”, the code is submitted to a server that compiles it and spits back a running applet or a compilation error.


Bitstrips: Avatar Builder
http://www.bitstrips.com/create/avatar/
Bitstrips
BETA v1.1
AVATAR BUILDER
CHARACTER CONTROLS
black~ 66 d ago
READ
Page One
All Bitstrips
CREATE
Comic
Character
Friend
Hello black
Home
Inbox
My Strips
My Friends
Account
Logout
BITSTRIPS
About
Help
Feedback
Report Bug
black
SAVE AVATAR
EASY
FACE
BODY
Colours:
Eyes
1/2
Pupils
1/1
Use me to update your status
Looking for your friends?
Find them here!
© 2008, Bitstrips Inc. / About / Contact / Press / Terms of use / Privacy Policy / Dev. Blog / Bug? Comment?
Produced by Core Matrix

BitStrips, 2008. Comic Strip social performance using your own self created caricature.

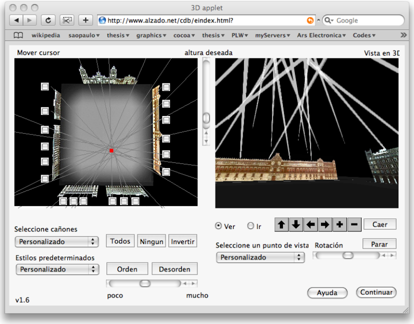
BitStrips [1] is a social online system that features a set of ready-made editable body parts to build cartoon characters and a tool to edit their posing and facial expressions in order to put them in a comic strip. When creating an account, users are guided through the process of creating their own avatar, and are then offered the opportunity to use themselves -represented by their avatars- as characters in their own strips, where they can mix themselves with characters borrowed from other users. This model of participation, just like OpenCode that is based on code writing, requires specialized users taking their participation seriously, and already having some skills to express themselves in the given medium, which is more complex to manipulate than a simple doodle or uploading a picture or video.

2.3.8 Performance and Participation

Rafael Lozano-Hemmer [64] is best known for creating theatrical interactive installations in public spaces. Using robotics, real-time computer graphics, film projections, positional sound, internet links, cell phone interfaces, video and ultrasonic sensors, LED screens and other devices, his installations seek to interrupt the increasingly homogenized urban condition by providing critical platforms for participation.

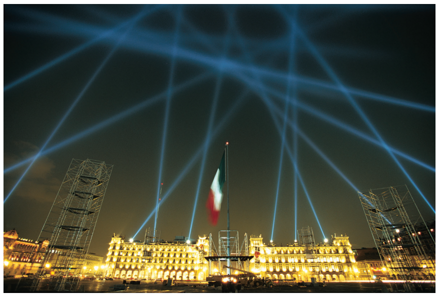
He created what may be the world’s largest interactive installation, Vectorial Elevation, where hundreds of thousands of online participants directed searchlights to create “light sculptures” over a city.

Vectorial Elevation was first installed in Mexico City in 1999, following installations in Vitoria-Gasteiz in 2002, in Lyon in 2003 and in Dublin in April-May 2004. In this kind of work the art is performed remotely by the participants, making them truly experience the creative process, as well as a feeling of participation in collective "anonymous" authorship. However, this model of interaction works one way only, following a "broadcasting model" of communication. The participant has access to the creative process, but only the result of this process will be experienced as the outcome, while process will happen in isolation.


3D applet
http://www.alzado.net/cdb/eindex.html?
Google
wikipedia
saopaulo
thesis
graphics
cocoa
thesis
PLW
myServers
Ars Electronica
Codes
Mover cursor
altura deseada
Vista en 3D
Seleccione cañones
Personalizado
Todos
Ningun
Invertir
Estilos predeterminados
Personalizado
Orden
Desorden
poco
mucho
v1.6
Ver
Ir
Caer
Seleccione un punto de vista
Personalizado
Rotación
Parar
Ayuda
Continuar

Rafael Lozano-Hemmer, Relational Architecture 4: Vectorial Elevation, Mexico City 1999-2000. Web Application interface.

Dialtones, by Golan Levin and collaborators [59], is a large-scale concert performance whose sounds are wholly produced through the carefully choreographed ringing of the audience's own mobile phones. Before the concert, participants register their mobile phone numbers at a series of web terminals; in exchange, new ringtone melodies are automatically transmitted to their phones, and their seating assignment tickets are generated. During the concert, the audience's phones are dialed up by live performers, using custom software which permits as many as 60 phones to ring simultaneously. Because the exact location and tone of each participant's mobile phone is known in advance, the Dialtones concert is able to present a diverse range of musical structures, such as waves of polyphony which cascade across the audience. In the case of Dialtones, the creative process was successfully activated in a participation loop that merged the artists with their audience.



Rafael Lozano-Hemmer, Relational Architecture 4: Vectorial Elevation, Mexico City 1999-2000. Vectorial Elevation is an interactive artwork designed to transform the Zócalo square in Mexico City. Using a three dimensional interface, a web site allowed you to design a light sculpture with 18 robotic searchlights located around the Plaza. A web page was made for each participant with photos from 3 webcams. The piece was unplugged on the 7th of January, 2000, after receiving around 800,000 visits from 89 countries and all regions of Mexico.

Reface, a participation example that takes unusual advantage of natural body gestures -like blinking- as triggers for interaction, is a video mash-up by Golan Levin and Zachary Lieberman that composes combinations of the visitors' faces. Based on the Victorian "Exquisite Corpse" parlor game, the Reface installation records and dynamically remixes brief video slices of its viewers' mouths, eyes and brows.

Reface uses face-tracking techniques for automatic alignment and segmentation of its participants' faces. As a result, visitors can move around freely in front of the display without worrying about lining up their face with the system's camera. The recorded video clips are "edited" by the participants' own eye blinks. Blinking also triggers the display to advance to the next set of face combinations, making communication and control between the human and the machine merge seamlessly with their environment in a true cybernetic manner.

Golan Levin, Gregory Shakar, Scott Gibbons, Yasmin Sohrawardy, Joris Gruber, Erich Semlak, Gunther Schmidl, and Joerg Lehner, Dialtones [A Telesymphony], 2001-2002. In this work, artists and audience participate together in the performance of a feedback system that becomes a "cellphone dialtone orchestra".

Golan Levin and Zachary Lieberman, Reface [Portrait Sequencer], 2007. Similar to the feedback observation loops that are created when gazing into one's own reflection in a mirror, the art is the performed process that results from the construction of a recombined self image.

3

System

3.1 Foundations

I mentioned in the introduction of this thesis a personal experience where I presented my work in progress to an evaluation committee, only to find out the evaluators were more interested in finishing the piece themselves than understanding what I was trying to do. There are two reasons why I think this happened. First, it was incredibly easy to transform the piece from its unfinished state to a seemingly finished one, because all that was left to do was to arrange a small number of rectangular modules in a configuration that admitted a very large number of different combinations: easy to do and with many choices. Second, it makes sense to think that most people would try to complete anything that appears unfinished. Any of these attempts could be used as feedback to evaluate or compare with other completion strategies, delivering an opportunity to reconsider, and either reinforce a decision already made or help find a more suitable one.

The idea of performing artistic process and making it receptive to feedback opens a scenario to study how the perception of external reactions can change the process, setting up a participation system where art can happen at two different levels, first performing process as an action that activates and includes the beholder's participation, and second, producing an outcome that can be read as a depiction of what happened.

It is not necessary to rely on the digital medium to explore the space of what I have just described -I often use non digital examples as anchors and inspiration-, but the digital medium is a perfect candidate to expand the limits of participation because it is interactive and telematic.

3.1.1 Natural Interactions

Interactions between people happen naturally. Communication, the process of meaningful interaction among living beings, often fails to happen between humans even if they share a common language. Art, an instrument of social communication, has traditionally been used to capture collective meaning and express it through the individual perspective of the artist, celebrating, questioning or discussing the specifics of this meaning. However, the twentieth century witnessed a tendency of artists to reconsider the nature of art as a constructor of meaning, suggesting that the beholder should actively participate in the construction of this meaning with the artist, turning the value of artistic communication from the contemplation of a finished piece to participation in an open process, turning the artist from craftsman into a sort of social worker, and appreciation of art from the perception of beauty to the construction of meaning.

Halfway through the process of writing this thesis, I visited the city of São Paulo in Brazil to participate in an international seminar about public art that used the public space in São Paulo as a case study and source of inspiration. Public art is closely related to what I am dealing with here. It takes the work of art outside of the protected space of the gallery and the museum to perform it in public, opening it to participation with no more mediation between the art and the participant than the artist himself (and in the case of this thesis, digital technologies).

The author painting a wall in a favela in São Paulo, Brazil. Photo by Mariliana Arvelo, 2008. Graffiti in the São Paulo favelas can play a community role where the artistic process is shared in the public space as a performance open for participation. Members of the community help the artist decorate the space, becoming artists themselves and changing the outcome.

The very notion of "public" is not well defined in the digital space, mainly because the digital is an artificial space where things are often designed in terms of predetermined needs. There is a layman understanding that Internet (and the World Wide Web) is an unregulated "public" space where content is distributed freely, and every voice can find a channel for expression. However, communication within digital networks is mediated by protocols of control [39], and a relevant discussion about what can constitute the public space, anonymity, and a kind of public digital art in the Internet and the Web has not yet matured. Inspired by Graffiti, I have been dealing with notions of "open" and "public" online for the last year and a half, building and deploying several networked systems with the goal in mind to facilitate an [almost] truly public space where visitors could express themselves and communicate with each other using drawings or posting pictures without requesting anything from them, and recording no more interactions from them than the ones they would willingly deliver. I will describe these experiments in detail later, for now it is sufficient to mention that my experiences with them inspired me to look back at the public space in the city, and Graffiti itself, from a completely renovated perspective. Cities and the Web are alike in many ways. They both are very complex dynamic systems that we experience and navigate partially, and we usually represent them as static totalities. Web pages can be thought of as the walls in the city, with elements like signage and doors and windows not unlike the different units of content that constitute a web page. Roads and streets are like paths through sequences of links, while people are simply like user profiles. However, at least two things stand out as fundamental differences: First, in the Web it is impossible to perceive a representation of a person detached from a collection of names and labels. The name might be fake, but everybody has a name, even if it is “no name”. The second thing to notice is the impossibility to graffiti the Web. Because of the artificiality of digital space in the web, a space to paint has to be designed and facilitated first, and a space where painting has not been implemented will never lend itself to be painted, because paint would just not exist there.

In the city of São Paulo, where graffiti plays roles from aggression to ornament and everything in between depending on what kind of walls are affected by it, there are circumstances where graffiti will be performed to the community, as a festive activity open to participation that is meant to embellish the improvised walls built with construction leftovers in the labyrinths of the favelas (brazilian slums). A group of artists will find out if a member of the community wants their shack painted, interior, exterior or both, and after a quick agreement the artists come back to paint the place during the day. Anybody else is welcome to participate and the mural evolves in an unplanned manner as it goes, following a single rule that anything anybody does has to come from the heart. As romantic as this action may sound, there are several questions that point out contrast with a possible similar situation when porting the performance of artistic process to the digital realm and the particular case of code:

- Accessibility: It is easy for anyone to access and experiment with something tangible like spray paint. Artistic expression is a physical gesture that is not mediated by an intellectual process like the understanding and writing of code. With paint, the outcome of an error can easily become a new visual feature, with code, an error will most likely prevent the program from running, interrupting visual feedback.

- Awareness: Nature and the digital space invert awareness. The spray painter can be aware of what the people around him are doing in the near vicinity, but will perhaps not be able to notice what someone is doing across the corner, and will be completely unaware of what's going on two blocks away. On the other hand, when writing code, the artist will pay little attention to people around him, and even if he tried, would extract very little information about what others are doing by looking at them typing code into their computers. He could however, gain rich interactions with people across the planet through the resources provided by the computer's user interface and remote communication resources.

- Individuality: The physical gesture of spray painting is a signature of the individual. With not much skill the painter can produce shapes that will distinguish his work from the rest. Code can just be passed around and produce exactly the same outcome, blurring individual authorship after a while.

3.1.2 Artificial Interactions

Herbert Simon makes a clear distinction between artificial and synthetic. He explains that an artificial thing is an imitation of a natural thing, and a synthetic thing is a duplicate of a natural thing where there can be no distinction between the natural and the synthetic [69]. As an example, a synthetic banana would be a banana bred in a laboratory that pops out of a petrie dish after bombarding a special configuration of molecules with a banana generation ray, where an artificial banana can be a rubber banana that can be found in regular sex shop. This distinction can be used to label the differences between natural human communication and human communication mediated by the digital medium. Simon puts aside the discussion around a precise delimitation of the boundaries between the terms “artificial” and “synthetic”, and concentrates on finding ways to identify the artificial from the natural. For him, there are four indicia that separate the artificial from the natural:

- Artificial things are synthesized by humans.
- Artificial things may imitate the appearance of natural things, while lacking, in one or many respects, the reality of the latter.
- Artificial things can be characterized in terms of function, goals and adaptation.
- Artificial things are often discussed, particularly when they are designed, in terms of imperatives as well as descriptives.

Even though Frank Popper’s [64] optimistic vision of the digital medium and its extension to the virtual suggests that a new vision of reality is being constructed by these new interactive representations, they should still be regarded as artificial, and this will not change until digitally mediated communication stops being conceived as a combination of fragments collected from the natural world, characterized in terms of function and goals, or discussed in terms of imperatives and descriptives. The real challenge is for the human intellect to conceive something that is not based in the experience of nature, perhaps if an iterative process of experience and representation can lead to something human that is not related to the imitation of a natural phenomenon, or the achievement of a particular set of goals.

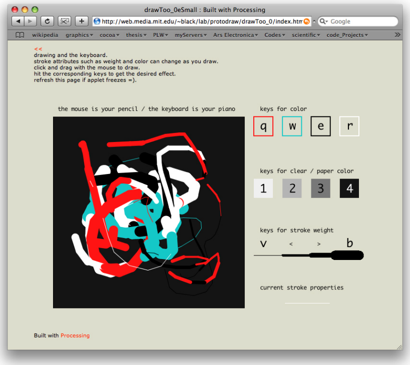
When I first got interested in working with the digital medium to produce art, I approached the computer influenced by ten years of experience using simulation tools to draw, manipulate pictures or create animations in digital representations of 2 and 3 dimensions. My personal history had my traditional skills of drawing and rendering greatly enhanced by the use of the computer. Simple interactions that belong exclusively to the digital realm -like the popular undo command or the iterative reviews of progress with a client over the net- changed my technical practice and how I communicate my craft for good. This determined the two anchors of the frame I was developing to approach and understand how to manipulate the digital medium:

- Technique.
- Communication.

I developed a constructive methodology based on iterative experimentation, where tools and the space to make them were in the same place (as location in space), merging the process of creating digital content and what would become its medium. How to make programs and how to use programs were the main questions asked. Using a program is easy once a working copy of the program is provided, but thinking about how to use a program that does not exist yet raises a number of problems. From an industry point of view, the design of a program will be centered on the task at hand, but an academic, more experimental point of view should be looking at the space of all possible ways to approach the task at hand. The performance of an interactive program is artificial, and as such it might imitate the appearance of natural interactions, and the aspects of reality that it will discard will be subject to the design process. As an example, If I want to write programs that can be used as tools to create artistic content, a degree of reality will immediately affect my judgement and lead me towards making programs to draw or play music, even though none of those art forms need the digital medium. However, the process of developing digital tools to make drawings or music suggests a space for digital experimentation that transcends the original intent, where the artificial is made not to imitate but to combine related aspects of different realities in a lateral manner, bringing an opportunity to create new concepts and create new worlds, where drawing and music, for example, can be the same thing.

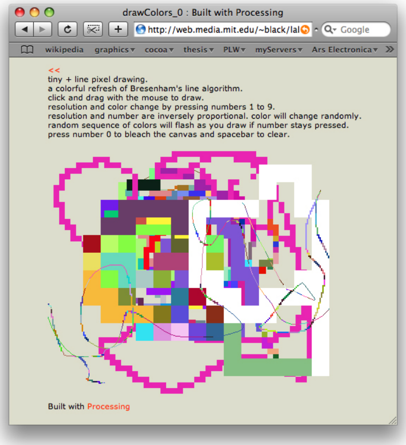
3.1.3 Tools: Drawing and Animation

Heavily influenced by tool centric social networks like the PLW's Openstudio, and by Golan Levin's early interactive drawing experiments (Floccus, Meshy and YellowTail) [50], over the summer and early fall of 2006 I deveoped a series of drawing experiments called DrawToo. Inspired by the potential of enhancing social networks with features that could let developers explore the design and modifications of those tools in the same space where artists were using them, I explored drawing interactions in ways not featured in commercial applications like the Adobe or Corel products. At the same time, Kyle Buza and Takashi Okamoto were developing OpenCode [11], an online space that allows programmers to develop interactive graphics with the Processing [66] [16] environment entirely within the web browser. OpenCode suggested that I could develop an abstracted library of drawing classes and methods to represent different ways of approaching time, drawing tools and surfaces in a flexible enough manner to facilitate easy ways for others to customize their own tools and interactions. Even though I focused on the drawing gesture as the main interaction to explore, the development of these simple prototypes revealed problems that I ignored before, like format compatibility and accessibility. As I have already mentioned, the Processing environment lets the developer compile programs as Java Applets, making it easy for people to access and play with them within the Web browser. However, file format management, database connectivity and networking imple-


drawToo_0eSmall : Built with Processing
http://web.media.mit.edu/~black/lab/protodraw/drawToo_0/index.htm
Google
wikipedia
graphics
cocoa
thesis
PLW
myServers
Ars Electronica
Codes
scientific
code_Projects
<<
the mouse is your pencil / the keyboard is your piano
keys for color
q
w
e
r
keys for clear / paper color
1
2
3
4
keys for stroke weight
v
<
>
b
current stroke properties
Built with Processing

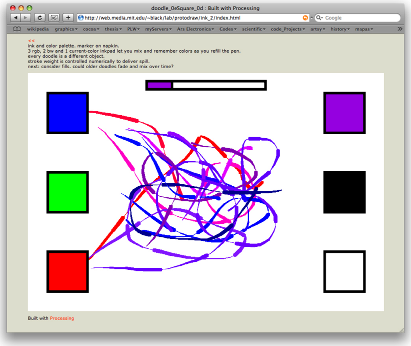
drawing and the keyboard.
stroke attributes such as weight and color can change as you draw.
click and drag with the mouse to draw.
hit the corresponding keys to get the desired effect.
refresh this page if applet freezes =).

Luis Blackaller, DrawToo 1, 2006. This collection of drawing application prototypes is available for testing in the following URL: "http://web.media.mit.edu/ black/lab/protodraw/drawtoo.html".


drawColors_0 : Built with Processing
http://web.media.mit.edu/~black/lal
Google
wikipedia
graphics
cocoa
thesis
PLW
myServers
Ars Electronica
<<
Built with Processing

tiny + line pixel drawing.
a colorful refresh of Bresenham's line algorithm.
click and drag with the mouse to draw.
resolution and color change by pressing numbers 1 to 9.
resolution and number are inversely proportional. color will change randomly.
random sequence of colors will flash as you draw if number stays pressed.
press number 0 to bleach the canvas and spacebar to clear.

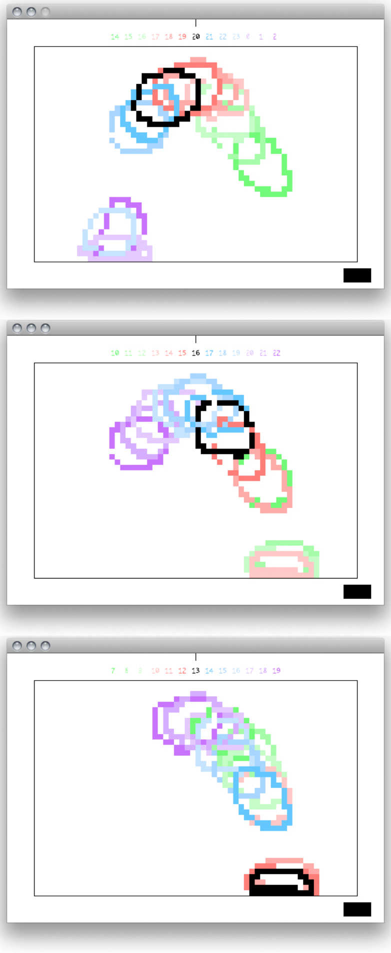
Luis Blackaller, DrawToo 2, 2006.


doodle_0eSquare_0d : Built with Processing
http://web.media.mit.edu/~black/lab/protodraw/ink_2/index.html
Google
wikipedia
graphics
cocoa
thesis
PLW
myServers
Ars Electronica
Codes
scientific
code_Projects
artsy
history
mapas
<<
Built with Processing

ink and color palette. marker on napkin.
3 rgb, 2 bw and 1 current-color inkpad let you mix and remember colors as you refill the pen.
every doodle is a different object.
stroke weight is controlled numerically to deliver spill.
next: consider fills. could older doodles fade and mix over time?

Luis Blackaller, DrawToo 3, 2006.

mentations inherited the quirks and limitations imposed by Java, making it very difficult to archive and manipulate content and user data over the Web. A question of relevance was deciding which interactions were more important: interactions of the person with the machine to produce content, or interactions between two persons, mediated by the machine, to communicate content. Later came a third option that was not clear at the moment, where communication could be mediated by the production of content via collaborative or participatory models.



Luis Blackaller, FlipClip, 2006. A simple animation tool with a minimal user interface.

Even though a finished drawing is a static object, the dynamic nature of the drawing process immediately suggests exploring animation. DrawToo 3, for example, features a dynamic array of color wells and a simulated pen that change their physical appearance in reaction to the drawing, and the strokes themselves keep mutating after they have been drawn, in a way similar to what happens when drawing on toilet paper with a sharpie marker. The digital medium has the tendency to reflect motion, and drawing can easily become animation. The last experiment I developed in this phase is an animation application called FlipClip. Inspired by an Icon drawing tool developed almost at the same time for an experiment I will mention later in collaboration with Brent Fitzgerald called the Tiny Icon Factory [22], FlipClip is a simple tool for animating 48 frames (or 2 seconds) in a low resolution canvas of 60x40 pixels. The timeline is navigated with the keyboard, following an interaction model that I already explored in DrawToo 1 and 2, where the simplicity of the tool, added to the deep knowledge acquired about it after building it almost from scratch, permits the definition of most actions as keystroke sequences, and allows the design of a clean Visual Interface where there is nothing but what's essential to pure traditional animation practice: drawing and timing.

Ideally, these tools should have been put in a social context where a community could have used them, but my main goal with them was to test my own understanding of the relationship between code and the simulation or invention of drawing gestures, and the best way for me to explore these relationships was by choosing an introspective approach, where I would develop, test and iterate the tools based on my own feedback, emulating perhaps the creative act of drawing, which is the one I am more familiar with after decades of experience.

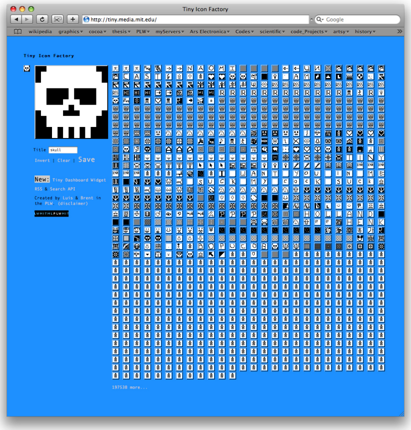
3.1.4 Social Systems: Anonymous Participation and Public Space

The experiments explored in DrawToo and FlipClip were simple to use, but mysterious and obscure to the newcomer. Most users will not know what to do with an application that only presents them with a blank canvas and a square on a corner, and even though today there is a strong enough interaction literacy regarding personal computers that most people would probably start "scratching" the digital canvas with their mouse or pressing well known key sequences, it will be hard for this experiences to transcend a playful engagement with the more immediate interaction level. The tool will be strongly perceived as a curiosity, and interaction explored in a manner similar to how one traditionally experiences a picture, as a passive beholder. Even though interaction activates a big deal of activity, the transition towards participation is not achieved. There are three main reasons why participation is not real in a closed environment that has no extensions beyond the tool and its user, and real participation can only happen when this three conditions are activated:

- Persistency. The work created with the tool has to be preserved.

- Multiple Accessibility. The work created with the tool has to be accessible from multiple locations.
- Collaborative Communication. The work created with the tool has to be accessible by others, and means should be provided for them to modify, expand or crop this work in any way they want.

Authorship, although important, has been proved not to be essential to activate participation.


Tiny Icon Factory
http://tiny.media.mit.edu/
Google
wikipedia
graphics
cocoa
thesis
PLW
myServers
Ars Electronica
Codes
scientific
code_Projects
artsy
history
Tiny Icon Factory
Invert | Clear | Save
New: Tiny Dashboard Widget
RSS & Search API
197538 more...

Title skull

Created by Luis & Brent in the PLW. (disclaimer)

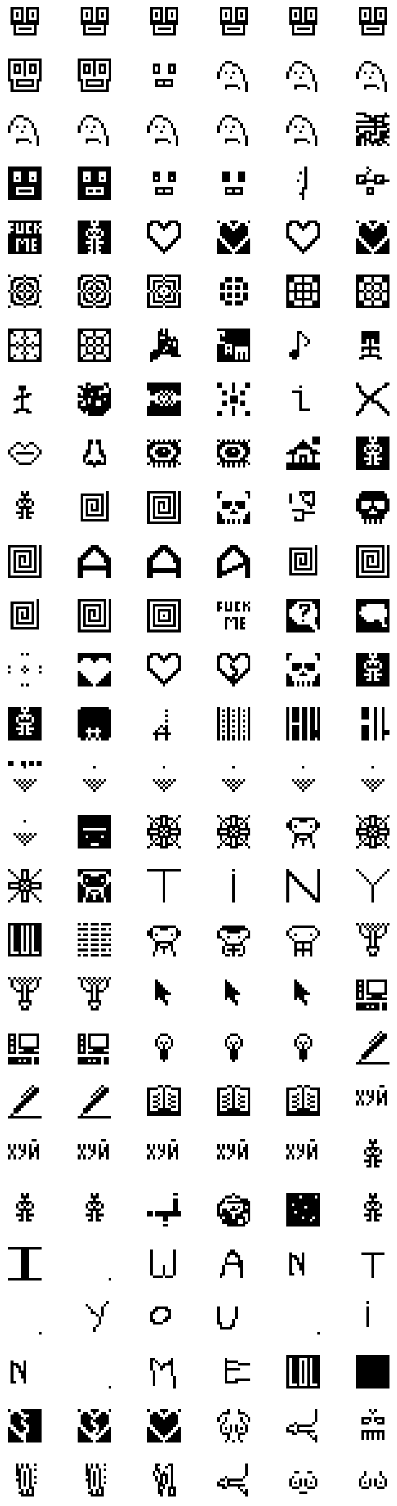
Brent Fitzgerald and Luis Blackaller, Tiny Icon Factory, 2006. An anonymous online space for the creation and communication of Tiny Icons.

Having these principles in mind, Brent Fitzgerald put together a simple communication model to add an equally simple drawing tool for making 13x13 pixel icons. It is the Tiny Icon Factory that I mentioned before [22]. Due to the trouble caused by trying to integrate the Ruby on Rails web development framework [20] [76] with the Java Applet I developed, we used Fitzgerald's implementation of the drawing tool directly using Javascript over the elements of the Document Object Model (DOM) in the page. The icons were modeled as 169 character long binary strings, and Fitzgerald's communication model consisted of an AJAX (Asynchronous Javascript and XML) [33] enabled web-form to post icon data from the drawing tool with corresponding metadata (icon's creation time-stamp, an optional title, and the IP address of the submitter), and a box where the last 1000 icons would be displayed. To make drawing easier, we blew up the canvas to a size at least 13 times bigger than the original icon. Even though it's significantly harder and much more limited to program drawing interactions with Javascript over the DOM than it is to program them using Java's graphics and user interface support, more limited interactions make a tool easier and more straightforward to use, and the complexity of modeling user interface components with javascript pays back with it's flexibility to talk to the server either through conventional HTTP requests or using AJAX.

Online social systems have been around for less than a decade, and the user-password model of identification has already become the ubiquitous standard to define the grounds for social activities in the Web. For me, it doesn't matter if the established protocols are flexible enough to let Web navigators create imaginary alter-egos and use them to disguise themselves, they still have to assume an identity, and they still have to surround the data generated after their interactions with an aura of protective privacy. Web applications that provide a service without requiring any virtual credentials open a discussion about how to access and create truly public digital content, and possible modes of moderating this content.

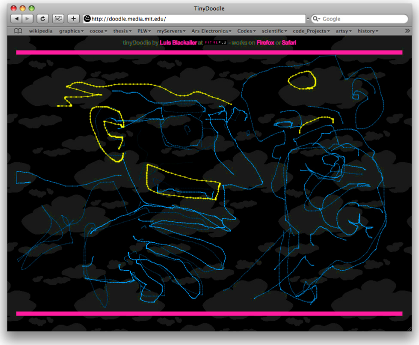
A visitor finds it easier to create content in a site where he doesn't have to bother creating an account or signing-in all the time, but it is possible that he will feel less attached to content that was too easy to create, or impossible to claim authorship over. This content, then, becomes valuable as a form of interaction, narrowing it's communication value to a more immediate space. To make this more clear, the difference between visual digital content that is anonymously and effortlessly produced is closer to the sentences in a conversation or the lines we write in a chat client, than it is to the work of art, closer itself to the written book or the pre-composed speech. The goal then becomes to simulate casual conversation with visual content, and in this case, communication between humans is regarded as the most important form of interaction.

Tool based social online spaces help understand the current rules of communication in the Web, and hint towards outlining new models. Interested in developing further ideas around public space and the usage of visual content for communication, I created another


Brent Fitzgerald and Luis Blackaller, Tiny Icon Factory, 2006. Sequence of 168 icons, participants unknown.

two anonymous web services: A public blackboard to chat with drawings and a public picture aggregator with tag classification and an embedded tracing tool.

The public blackboard is called TinyDoodle [23]. It's connection with Maeda's OneLine is almost direct. At the time when I developed this space I knew about at least another ten other websites where people from different places could draw at the same time and see what the others were doing. It is almost trivial to port the technologies that are used for chat and audio communications to make them deal with graphical components instead, but my interest was not to focus on innovation and looked more to use a tool like this one as a controlled medium for performance and communication. Because I was making the tool, when I invited people that knew me, they assumed a particular relationship with the tool and the communication channel, resulting in meaningful visual conversations. A common-ground imagery that we all recognized filled the space with content, inspiring strangers that visited the site by chance. TinyDoodle also gave me an opportunity to experiment with the performance of process when making a drawing, because its chat nature made it possible for anyone else to look at how I would make a drawing.


TinyDoodle
http://doodle.media.mit.edu/
Google
wikipedia
graphics
cocoa
thesis
PLW
myServers
Ars Electronica
Codes
scientific
code_Projects
artsy
history
tinyDoodle by Luis Blackaller at MITML PLW - works on Firefox or Safari

Luis Blackaller, TinyDoodle, 2007. An anonymous online blackboard to chat with doodles.

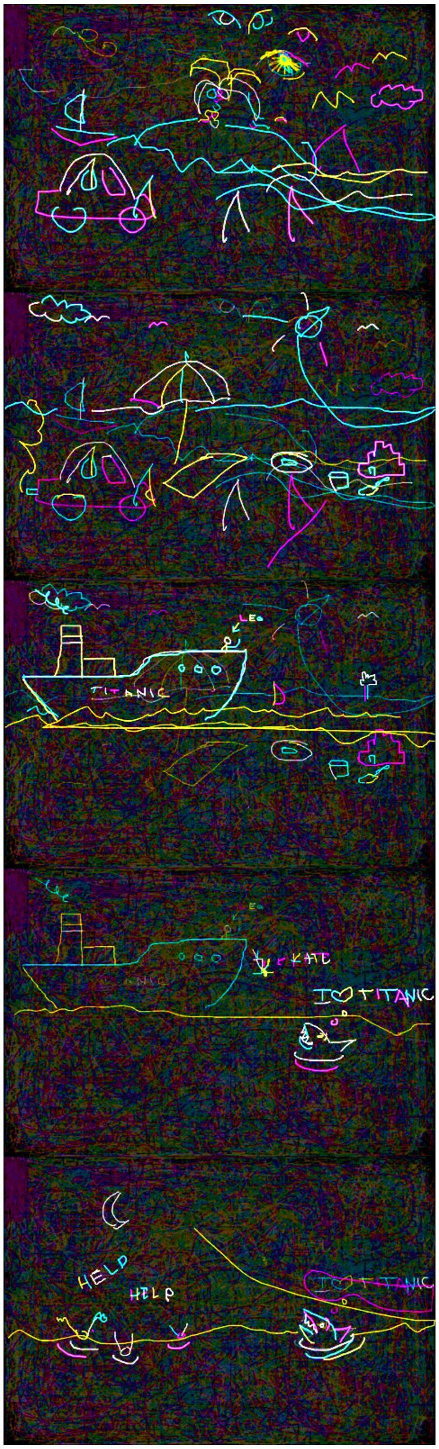
The chat communication model opens three main modes of interaction between a performer and the participants. These modes will be present in each aspect of communication when I outline a model for the digital representation of the artist studio:

- Performance.
- Participation.
- Interruption.

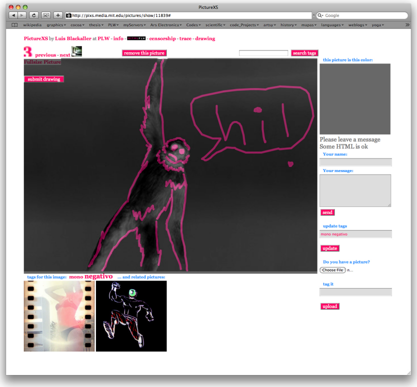
In the realm of pure form, I used TinyDoodle to experiment with slices of drawing action through the machine-relative passage of time. The computer captures data coming from the drawing gesture, and there are infinitely many ways to organize this data so it can be translated back into a drawing later. Even though there is nothing more than a discrete mapping of points from the mouse or a similar device, the way these points are organized in terms of the recorded passage of time offers a lot of freedom, and every choice made has a meaning that can eventually determine how this data is going to be interpreted back into graphics. For example, if my interest is only on the finished drawing, there is no need to store more than the coordinates of the points and their edge relationships to reproduce the drawing, but, if my interest is to playback the drawing action, then I need to record a time-stamp for each point, so I can ask the computer to redraw the sequence in the correct order and speed.

A drawing is made of many lines, every time the hand goes down to produce a stroke a new line is created. Each one of these lines can be long and complex. It can even be thought of as an independent drawing. Should every single line have the properties of a full drawing or not? In TinyDoodle, I tried to pair as much of the drawing actions with the machine-relative time, as an effort to later develop tools to browse the resulting visual conversations in many different ways.

The last public anonymous web service I will discuss here is the public picture aggregator I mentioned before. PictureXS [14] was meant to be a simple modular scalable application, to be used as a scrap-board for me to learn Ruby on Rails. It was first conceived as a tool to collect and organize pictures in an open, anonymous, online space. I would be the primary user, but anyone else that



Luis Blackaller, TinyDoodle, 2007. Five frames in sequence describing a visual conversation, participants unknown.


PictureXS
http://pixs.media.mit.edu/pictures/show/11839#
Google
wikipedia
graphics
cocoa
thesis
PLW
myServers
Ars Electronica
Codes
scientific
code_Projects
artsy
history
mapas
languages
weblogs
yoga
PictureXS by Luis Blackaller at PLW - info - - censorship - trace - drawing
3 previous - next
remove this picture
search tags
Fullsize Picture
submit drawing
hi!
this picture is this color:
Your name:
Your message:
send
update tags
update
Do you have a picture?
Choose File
tag it
upload
tags for this image: mono negativo ... and related pictures:

Please leave a message
Some HTML is ok

mono negativo

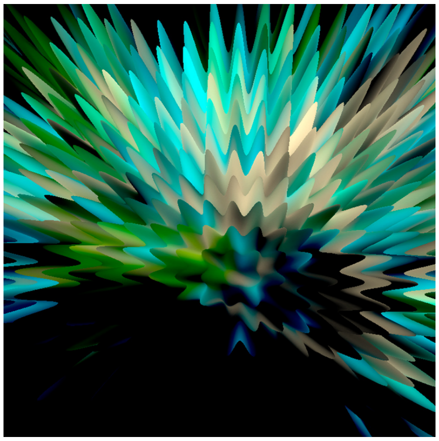
Luis Blackaller, PictureXS, 2007. An anonymous online picture aggregator with tags, comments, censorship and and an embedded tracing tool.

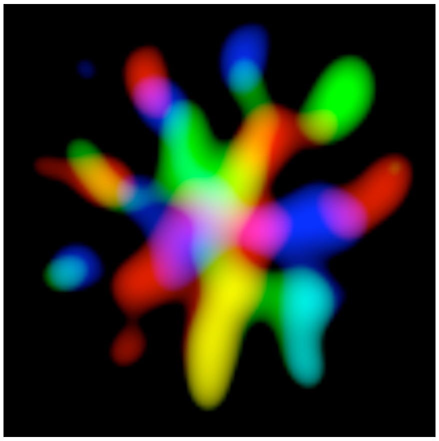
stumbled upon it could browse through the pictures, tag them, or post their own pictures if they wanted. Over time, I have built a number of extensions to the picture model that include labels for censorship, tracing over the pictures, and a message model to comment on the pictures, giving additional classification information to the original tag model that was implemented in the beginning. Until now, I have avoided incorporating a user model to the application. Forced anonymity has been reflected in the statistics that compare posting versus browsing actions. According to Google Analytics [9], an average of 43 visits a day for the last year and an average of 6 page-views per visit result on an average of approximately 258 page-views per day and a total of 92,804 page-views the last year, compared with 12,970 pictures posted over the same year, result on a scale factor of 7.155 between the more passive action of browsing through the pictures and the participatory action of posting new ones. Looking at these numbers, it would be safe to assume that the provided degree of authorship and the degree of active participation must be directly proportional, but the model explored with the Tiny Icon Factory suggests exactly the opposite conclusion: in a space where no authorship was acknowledged, the active creation of icons has been larger than the number of visits to the page.

From a personal point of view, I think participation can be encouraged through a combination of interest, social interaction and creative action. All the experiments reviewed in the last couple of subsections deal with interaction and participation. In one way or the other, even a tool that has been isolated from the connections provided between humans by a networked environment asks for a participation level that goes far beyond the mere passive contemplation required to experience a piece of art in the form of a static visual artifact. The question to ask is which of the observations made about highly tangible interactions can be of any meaning when applied to a mostly intellectual interactive process, such as the experience of writing, editing and/or reading computer code. After running a script, the resulting process can be of an audiovisual and/or tangible interactive nature, but the connection between code and its result is not easy to note, and sometimes as hard to understand as the relations between the invisible laws of particle physics and their definite influence in everyday human experience.

3.1.5 Code as a material: E15, a new world

E15 [4] is a networked interactive 3D graphics programming environment developed by the Physical Language Workshop (PLW) at the MIT Media Lab between the summers of 2007 and 2008. E15 is platform to experiment with new visual interfaces to access the World Wide Web, and a state-of-the-art programming environment for motion graphics called oGFx. I will provide more detailed descriptions of these technologies in the design section of this chapter when I talk about resources. What concerns me here is the early days of development of the oGFx platform. Working in collaboration with Kyle Buza to build the foundations of an interactive graphics programming engine, I was inspired to approach computer graphics in a different manner, closer to their core, as an ongoing effort to find a middle ground between raw data processing and visual representation. Possible digital representations of space, motion, form and light were compared with actual resources and techniques provided by computer graphics technologies in an effort to build the simplest possible and most flexible bridge between an interpreted programming language (Python) and the graphics resources provided by OpenGL and Apple's Cocoa Framework for MacOS X 10.5. oGFx is a networked interactive graphics programming environment for motion graphics. The following two paragraphs explain the foundation of our vision behind the oGFx



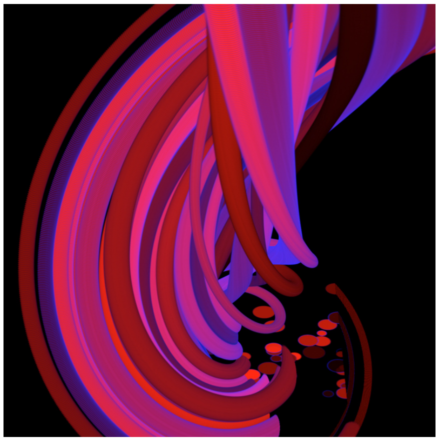


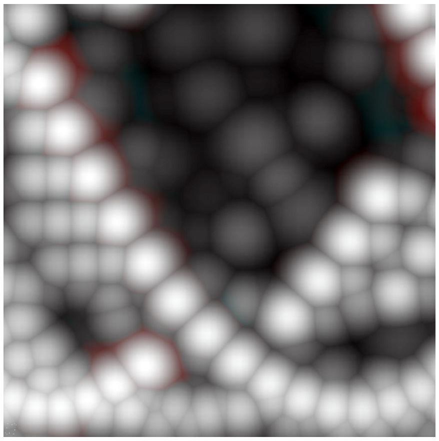
Previous three figures: Luis Blackaller, oGFx: Urchin, Splatter and Hex, 2007. Interactive digital explorations in “depth”, “color” and “form”.

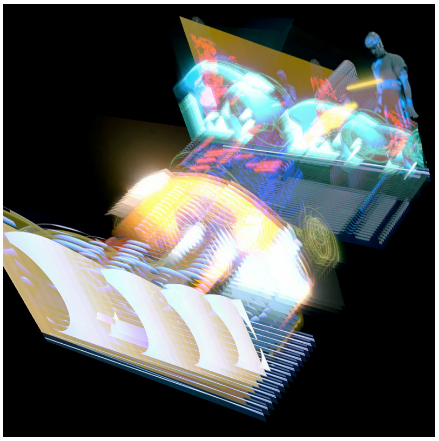
graphics engine:

The exponential growth in computer power over the last two decades has delivered a visually compelling and interaction immersive experience that encourages researchers to look for a redefinition of the way information is displayed on screen. Graphics programming is both a tool and a subject in the quest to shape digital information in meaningful new ways, providing resources to experiment with interactive environment prototypes. In that spirit, recent efforts have been made to design programming environments that favor visual interfaces, avoiding the difficult task of writing code. However, these efforts shelter users from dealing with core concepts of the underlying graphics pipeline, and prevent access to full featured resources that are only available by writing code. Experimenting with code provides users with a more powerful understanding of graphics programming, giving them a perspective that can break away from conceptual limitations like the separation between 2D and 3D graphics environments. Our goal is to provide an experimental platform that can help change public perception of interactive motion graphics.

Design Principles: The design of an interactive graphics programming environment should encourage modularity and connectivity. Resources should be packaged to be developed and tested separately, with easy ways to connect them at runtime to test their interactions. Based on the idea that it is easier to envision two dimensional mathematical structures than their three dimensional counterparts, we have placed the two dimensional graphics principles in the top layer of our system's hierarchy. Intuitive natural alterations of a flat surface, like deformations or fragmentations, are used as a starting point to project the dynamic 2D textures onto a rich three dimensional environment. By having full control







Previous three figures: Luis Blackaller, oGFx: coil, cell and tron. Interactive digital explorations in "depth" and "time".

over the object model that runs the 2D context and enough connections to reach down into the lower levels of the graphics pipeline, we can inject parameters from the programming environment across dimensions, allowing synchronization of dynamic effects at all levels, from texture to vertices, geometry, and fragments.

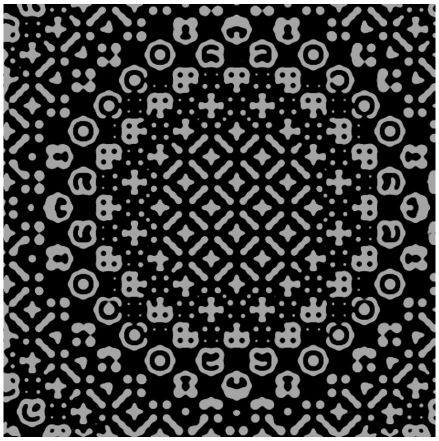
The methodology I chose while experimenting with oGFx was formalist, constructivist, futuristic and rationalist, following conceptual ideas I later found articulated by Christopher Alexander [32], perhaps because I was trying to reconcile my theory of digital visual representation with the analytic nature of code. The following are the essential components of digital visual representations I proposed as the building blocks, atoms, or alphabet, to construct a personal language of digital visual expression:

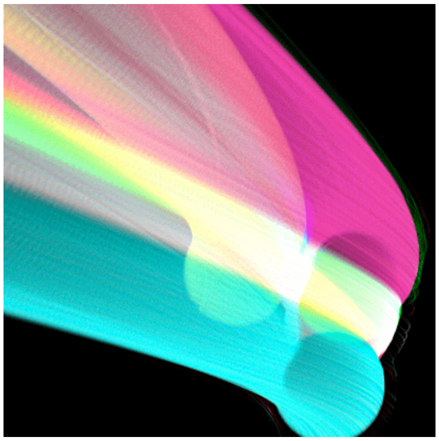
- Space: depth.
- Light: contrast, color.
- Form: symmetry, structure, rhythm.
- Motion: time, history, rhythm.

Whether I will succeed or not to construct this language based on the proposed atomic components doesn't matter. What's important is the implicit effort to reconcile the experienced aspects of reality outlined by these principles with digital representations built by putting together a system of digital processes that are controlled by the rule based patterns defined in the written code.

3.2 Scope

An implementation of a system like this one deals with many questions and problems that are beyond the scope of this thesis. It is of essence to find a balance between conserving focus on the key





Previous three figures: Luis Blackaller, oGFx: sanrio, pacman, orbit. Interactive digital explorations in “color”, “structure”, “contrast” and “rhythm”.

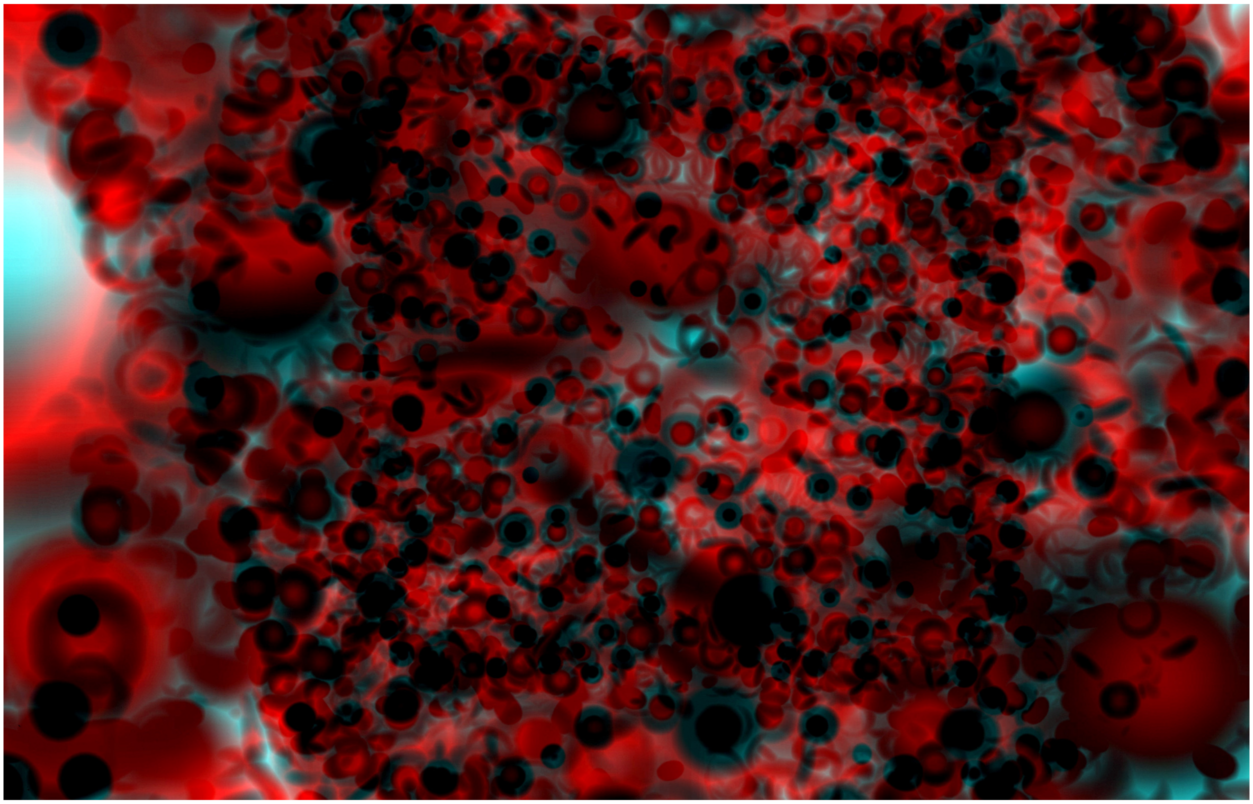


Fig. 3-1: Luis Blackaller, oGFx: blood, 2007. The coding session that produced this capture was the result of several hours manipulating sensitive parameters, such as OpenGL blending functions, CoreImage image processing filters, and the variables necessary to move and draw the original elements in the scene. Like with many other experiments of its kind, keeping track of the state that produced a specific result becomes hard, suggesting the need to consider an image based code versioning system, to help the programmer roll back to particular versions of the code by browsing through an image log.

points of this thesis, while staying aware of issues that are worth consideration in a speculative level, even when they won't affect the implementation process.

3.2.1 Trust

Security problems and issues of trust lead to the design of mechanisms of control to regulate identity and prevent abuse of the system. Opening one's own code to view and copy by an unidentified participant assumes an implicit agreement of respect between both parts, placing the artist in a position of vulnerability. Authorship could be lost, and chunks of code could be reused with a malicious intent. Furthermore, in many cases a script can contain private or personal information like passwords and API keys provided by popular services like Flickr [7], Facebook [5] or Google [8]. Opening this information to the public is equivalent to distributing copies of important personal identification documents like passports or credit cards in a plaza or a park. These problems must be dealt with when a system like this one is designed. However, these important issues are not essential to the questions about interaction, creativity and communication. From an experimental point of view, taking them into consideration does not affect the exploration of interaction space, but it will delay them until the mechanisms of control are set in place. For this reason, I will carefully outline the limits of scope of this work, so the problems that I face when designing or testing the system can be discarded if they don't affect the course of communication between the artist and the participant.

3.2.2 Limits

These points are key to the scope of this thesis. Any problem of technical, social or conceptual nature that does not determine a dramatic change in the dynamics of these interactions can be avoided, ignored, or discussed in the realm of theory:

- The notions of the studio, process and authorship when mediated by digital technology.
- The mediation of digital technologies when communicating artistic creative process.
- Human personal communication when mediated with digital visual content and code.
- The participant's perception of the digital art form as interactive graphics and the code that runs it.

- The potential relevance of interrupting artistic process with feedback from the participants.

3.3 Approach

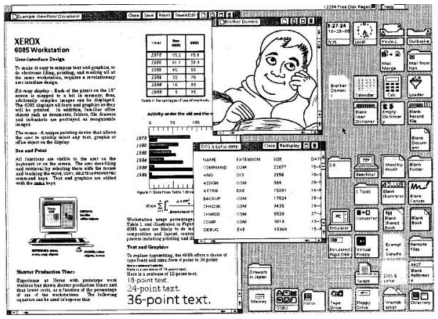
The previous sections of this thesis explain how the participatory features of digital media make digital art the ideal medium to discuss key aspects of the contemporary state of art in general: Art shifts from the traditional conception of it as an object or a commodity, to the more radical conception of it as a system of rules, interactions, and outcomes, thus becoming immaterial and organic. Objects produced by an art system become nothing more than representations of nodes that can help read the complex interactions occurring while the art unfolds itself over time. Authorship is blurred by interactivity and participation, making the collective more meaningful than the individual.

In particular, the system proposed should offer a straightforward mode of interaction to expose computer code and the interactive visual abstractions it generates, facilitating direct communication between code writer and audience at the time code is being written, and opening space for feedback, to change the final outcome of the written code by responding to intermediate appreciations of the generated digital graphics.

3.3.1 Goals and Expectations

I expect to develop a better understanding of digital art, after observing my response to feedback over a work in progress, and I will elaborate a discussion about the meaning of opening the intimate studio space to external observation and perturbations. All conclusions reached in this thesis are expected to be of a subjective nature, and their main importance lies in the discussions inspired around the themes of authorship, artificial communication, virtual public space, and the role of computer code and interaction design as the new literacies of our time.

When looking at a plum in our hand we can see nothing more than just a plum, and maybe an extra couple of levels of detail like the texture of the skin. There is no way to see the electrons, atoms, or even chains of molecules that hold the plum together, giving it glow, color and taste. It has been an incredible feat of human imagination that people like Democritus have had the insight to conceive things like atoms, with no way to measure more granularity than one tenth of a millimeter. We can't experience the deeper structures of nature physically, but the rise of digital technology has given us a space where an artificial reality can hold objects like plums with their DNA structure open to our limited senses. Until now, the design of digital tools has been following principles that try to shelter untrained users from the difficult experience of dealing with code. It is a natural thing to do, because things that are easy to use will always sell better, and the Graphical User Interface has evolved since the times of Douglas Engelbart [37] to make it possible for people to make complex programs and digital systems without ever having to look at a single line of code. However, between the hidden code and the graphical components of a user interface, many decisions have been made beforehand, limiting the potential to explore alternative representations. I expect this work will make users aware of the code signature underneath the visual representations they interact with, hopefully helping inspire some curiosity about it.


Example ViewPoint Document
Close
Save
Reset
Save&Edit
XEROX
6085 Workstation
User-Interface Design
Brother Dominic
DOS & Lotus data
Close
Redisplay
NAME
EXTENSION
SIZE
Text and Graphics
18-point text.
24-point text.
36-point text.
Calendar
Calc
Loader
Blank Folder
Blank Book
Floppy Drive
Tape Drive
Directory
Emulator
Converter
Virtual Floppy
Drawers in Japan

Screenshot from a Xerox Star, 1981. Based on Douglas Engelbart's research, the Star interface pioneered the desktop metaphor that would become ubiquitous in later Graphical User Interfaces.

3.3.2 Design Alternatives

The artist and the beholder are meant to communicate remotely through the use of telematic technologies. When thinking about this, a few different interactions are available. Because the space of interaction is the computer, and there is already a broad collection of commercial systems whose main functionality is communication, I will examine some of them one by one, thinking about how could they help define open channels to communicate the space where code is written and tested, with the space where the program is experienced by the beholder. There are two main variables that affect the way these interactions are outlined:

- Immediacy, or how close to the author is the beholder, and how close to the author's coding are the beholder's observations.
- Persistency, or how much of the communication will be memorized by the machine for future analysis.

Both of them are key to the conceptual definition of the space. A transparent mode of observation, which represents total immediacy but no persistency, can be exemplified by Apple Computer's recent screen sharing technology (available in Mac OS X version 10.5), that lets users invite remote participants to directly interact with their own user interface through Apple's iChat protocols. In this scenario, I could be working on someone else's computer anywhere in the world through their own user interface. I could take over their cursor and start moving their windows or typing on their open text documents. The disadvantage of using such an approach to open the creative space of a digital artist is that it offers too much freedom, opening more than what's necessary; everything in my computer will be accessible to the beholder as I work, and they could read my emails, browse my collection of images, open Adobe Photoshop or simply close the application I am working with, instead of interacting with my creative process. To define the space for interaction between my audience and my studio I first need to deal with isolation. The artist studio has traditionally been a closed space, and a partial intention of this thesis is to examine what happens when this space is opened. But there are many different levels of opening. Filtering out events that are not directly related to the creative process is a fundamental step. Unrelated items should be discarded, unless they could be found to have a meaningful role in communication. The design principle of reduction suggests that interaction should be leveraged using as little digital technologies as possible. However, a superficial approach to this principle can lead to contradictory loops; If one application can do what is done by another two, the reduction of the number of applications necessary to perform the task increases the complexity of the resulting new application, bloated with the new features.

Apple Computer's recent screen sharing technology is equivalent to having the beholder in the studio and letting him freely do whatever he wants, even to the extreme of taking the tools away from the artist and use them himself. It might be a good space for collaboration, where a group of artists work together in the same artistic process, but it opens an opportunity to negatively affect the process, disturbing its fragile balance and separating it from meaning. Artists need a degree of control over the creation process, even if it is in the form of directing or balancing external forces that are independent from them.

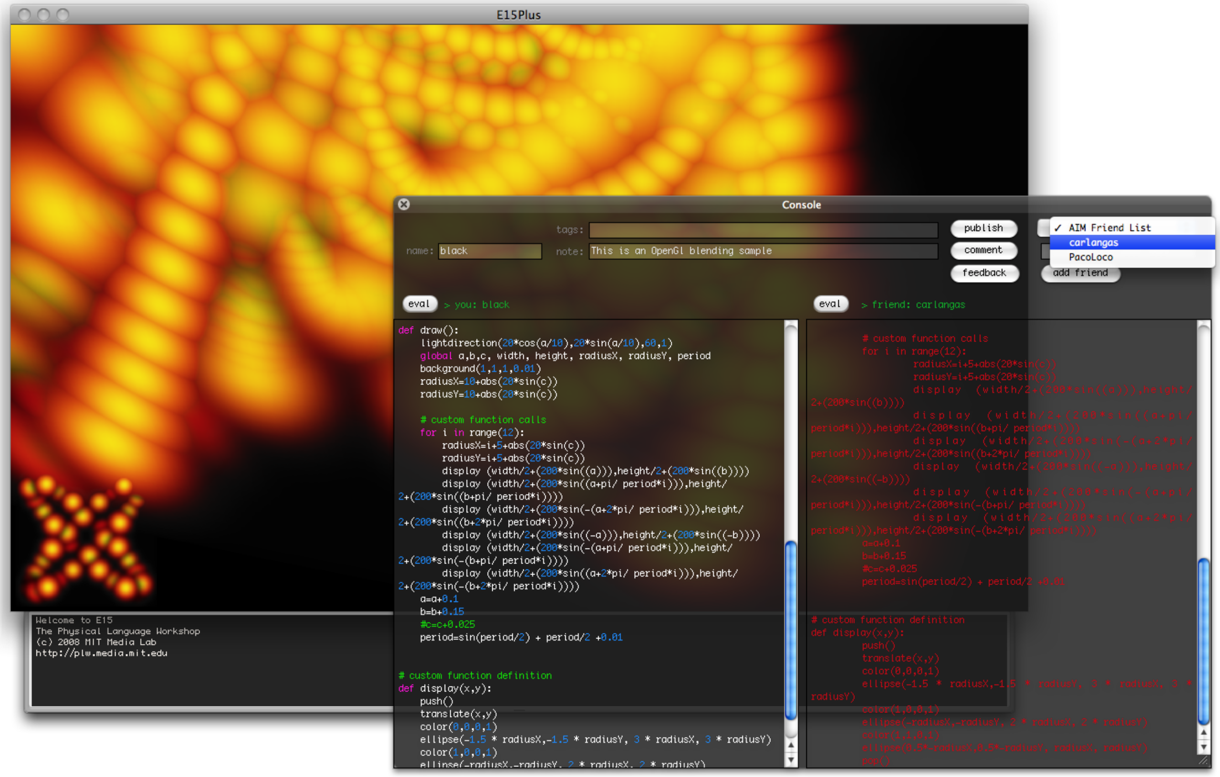
A crude system could be built without using digital technology at all, a video camera with a radio transmitter on one side, and televisions tuned to the right channel on the other side, and telephones to talk both ways. However, the digital medium surpasses by far this analog experience, because it can transmit the actual contents of the art and let the beholder directly interact with them, where in the television example all the beholder gets is an image of the artistic process, and no means to experiment with it. Conventional chat services, live video feeds, web forums, blogs and even email are a source of inspiration to model remote interactions around the studio space. The most important aspect of the digital medium is that communication is fully supported both ways, allowing access to all of what each side is doing. Even though the means to enable this communication are not trivial, and some design decisions will have to be made based on limitations imposed by what the designer can't do, the main issues to consider are not of a technical nature, and should be determined by a careful examination of the space where interaction (between people mediated by the machine) will happen. In this case, I will focus on the theoretical examination of three different ways to understand remote communication between the nodes of a network that represents an artist and a group of beholders, each one posing a set of advantages and disadvantages. The time constraints of a masters thesis don't permit me to explore the space of each option experimentally, so I will take a theoretical approach to discard two of them and experiment with only one of the communication schemes that follow:

- Conversation.
- Broadcasting.
- Publishing.

3.3.3 Synchrony or Asynchrony

The difference between the three models listed at the end of the previous subsection is a matter of transmission synchronization. In communication theory and electronics, synchronous transmissions are synchronized by an external clock, while asynchronous transmissions are synchronized by special signals along the transmission medium. In a trivialized way, it can be said that synchronous communication between a human and a machine can be exemplified with driving a car, or between humans when Salsa dancing, because the communication nodes must be synchronized at all times in order for the task to be performed. Piloting a modern computerized airplane or using today's telephone illustrate the asynchronous model, because the nodes perform their tasks separately and submit or check messages to the other nodes at any given time, without responding immediately. These two methods are essential to define abstractions of the three communication models that can be used to develop the studio system proposed in this thesis.

The conversation model can be understood to be totally synchronous, because communication has to happen under the same clock. If I say something it has to be delivered immediately to the other participants, and I should be able to read any reaction they've had almost at the exact time they've had it. If two messages are emitted by different ends at the same time, both messages should be received at the same time as well, even if that can lead to confusion. Special considerations shall be taken to reduce the noise that can cause this confusion. For example, a different channel can be opened for each node of the communication model, so that incoming and outgoing messages wont overlap in space, even when they can overlap in time. The broadcasting model should be partially synchronous, in the way that there should be a reception channel for the artist where his actions should be transmitted live, and kept under the same clock, but the participants could be following their own clocks as they communicate with the artist through channels that are not the one occupied by the artistic process. The broadcasting communication model resembles the structure of a television call-in show, where phone calls are taken from callers listening at home, in their cars, etc, but might not immediately passed back to the performers at the show.



tags:

name: black

note: This is an OpenGl blending sample

> you: black

```
def draw():
    lightdirection(20*cos(a/10),20*sin(a/10),60,1)
    global a,b,c, width, height, radiusX, radiusY, period
    background(1,1,1,0.01)
    radiusX=10+abs(20*sin(c))
    radiusY=10+abs(20*sin(c))

    # custom function calls
    for i in range(12):
        radiusX=i+5+abs(20*sin(c))
        radiusY=i+5+abs(20*sin(c))
        display (width/2+(200*sin((a))),height/2+(200*sin((b))))
        display (width/2+(200*sin((a+pi/ period*i))),height/
2+(200*sin((b+pi/ period*i))))
        display (width/2+(200*sin(-(a+2*pi/ period*i))),height/
2+(200*sin((b+2*pi/ period*i))))
        display (width/2+(200*sin((-a))),height/2+(200*sin((-b))))
        display (width/2+(200*sin(-(a+pi/ period*i))),height/
2+(200*sin(-(b+pi/ period*i))))
        display (width/2+(200*sin((a+2*pi/ period*i))),height/
2+(200*sin(-(b+2*pi/ period*i))))
    a=a+0.1
    b=b+0.15
    #c=c+0.025
    period=sin(period/2) + period/2 +0.01


# custom function definition
def display(x,y):
    push()
    translate(x,y)
    color(0,0,0,1)
    ellipse(-1.5 * radiusX,-1.5 * radiusY, 3 * radiusX, 3 * radiusY)
    color(1,0,0,1)
    ellipse(-radiusX,-radiusY, 2 * radiusX, 2 * radiusY)
```

> friend: carlangas

```
        # custom function calls
        for i in range(12):
                radiusX=i+5+abs(20*sin(c))
                radiusY=i+5+abs(20*sin(c))
                display  (width/2+(200*sin((a))),height/
2+(200*sin((b))))
                display  (width/2+(200*sin((a+pi/
period*i))),height/2+(200*sin((b+pi/ period*i))))
                display  (width/2+(200*sin(-(a+2*pi/
period*i))),height/2+(200*sin((b+2*pi/ period*i))))
                display  (width/2+(200*sin((-a))),height/
2+(200*sin((-b))))
                display  (width/2+(200*sin(-(a+pi/
period*i))),height/2+(200*sin(-(b+pi/ period*i))))
                display  (width/2+(200*sin((a+2*pi/
period*i))),height/2+(200*sin(-(b+2*pi/ period*i))))
        a=a+0.1
        b=b+0.15
        #c=c+0.025
        period=sin(period/2) + period/2 +0.01

# custom function definition
def display(x,y):
        push()
        translate(x,y)
        color(0,0,0,1)
        ellipse(-1.5 * radiusX,-1.5 * radiusY, 3 * radiusX, 3 *
radiusY)
        color(1,0,0,1)
        ellipse(-radiusX,-radiusY, 2 * radiusX, 2 * radiusY)
        color(1,1,0,1)
        ellipse(0.5*-radiusX,0.5*-radiusY, radiusX, radiusY)
        pop()
```

```
Welcome to E15
The Physical Language Workshop
(c) 2008 MIT Media Lab
http://plw.media.mit.edu
```

Fig. 3-2: E15:chat mockup scenario. In this model, programmers can have a coding conversation, looking synchronously at each other's code as it is edited, and choosing to evaluate between their own code and their partner's. The option to edit each other's code is discarded because it leads to confusion; Only your own script should be edited by you. Personal communication can happen in the form of embedded comments within the code. A web based logging application can still be used for optional persistency of important states of the code. This is a perfect "future work" model to explore.

0in-2.5in

The publishing model is asynchronous the same way a conversation model has to be synchronous, so they can be considered opposites in these terms, with broadcasting sitting as an intermediate step. In the broadcasting model, artistic process happens independently from the participant's activity, and independent channels are opened to let each of them check on each other's state at any given time. In this model, timing is not critical, and messages can be sent at irregular intervals on both sides. The obstruction of direct communication opens a space for moderation and control.

The conversation model will most likely lead to confusing interactions when more than 2 participants are involved (artist included). Feedback between several participants could overlap and the artist would have little control over incoming messages, because the communication channels will always be open and transmitting. The best use of the conversation model is when there is only one participant, and the artistic process can be envisioned as a continuous dialog between both ends. Ideally they could be running and testing their own code at the same time, and they could be able to look at each others typing activity, writing messages to each other as comments in their own code, and copying whatever they could find attractive from each other's writing. The hierarchical differences between the artist and the beholder will be almost negligible, established only at the initial steps, because the artist will happen to be the one that starts the session. This model of participation, however, will be of not much use if the participant has little experience with code, because the artistic process will be interrupted by the didactic process of helping the participant understand.

3.3.4 Comments and Version Control

This project takes a radical approach in the realms of commenting and version control. The choice of publishing code with pictures as the mode to communicate process is in a way a form of version control. Different instances in the evolution of the code are logged together with a visual annotation that works both as a result and as a reference to what that particular chunk of code achieved. Future revisions can let the programmer roll back to a state in the past of the code, but it is different from traditional version control systems like CVS or SVN. It does not activate a control mechanism to moderate a team of programmers working over the same code.

It merely serves as a log for process that represents its evolution over time.

A few places have been considered to annotate in the process log. There is, of course, the space for conventional comments within the code itself, that can be used to explain aspects of the code itself as usual, or to embed messages from the artist to his audience. Along with these, the relational database structure embedded in the web application component opens an opportunity to attach other kinds of data to the logging. Tags are facilitated to help describe or classify the generated pictures, and a system of external comments has been added to the code-picture pairs. This is a space open for detailed conversations between the artist and his audience, where the participants can actually submit chunks of code back to the artist for testing. The asynchronous nature of the publishing model lets these conversations unfold after the performance time is completed. An interesting implementation question regarding the design of the studio space is how to pipe the conversation data through the performance space without distracting the artist from the process.

Commenting becomes a wrapper around the process, opening a discussion about how code is written from within the process itself, rather than simply having the function of explaining what the code does.

3.3.5 Sessions

Ideally, sessions should be the fundamental top node in the system's hierarchy. It is important to determine if a sequence of code submissions belongs to a session by checking for the following conditions:

- If the submissions belong to the same uninterrupted process.

- When this process is terminated.
- When a new process is initiated.

However, the implementation of sessions is of organization relevance. It is not immediate to the perception of process, and it does not affect participation. Sessions and an authorship model to let several artists perform separately within the same studio space are important features to consider in the future, but they can be discarded from the present scenario.

3.4 Design

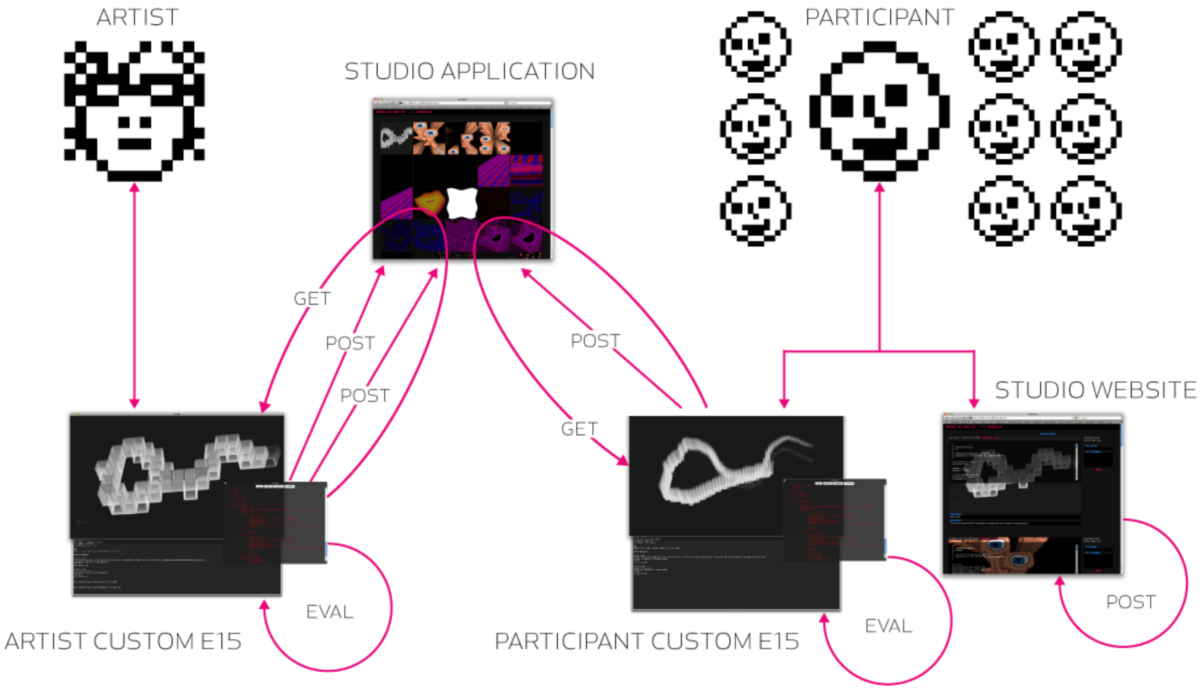

ARTIST
STUDIO APPLICATION
PARTICIPANT
GET
POST
POST
POST
GET
STUDIO WEBSITE
EVAL
EVAL
POST
ARTIST CUSTOM E15
PARTICIPANT CUSTOM E15

Fig. 3-3: Studio Interaction diagram. At the same time the artist is performing in E15, participants have the option to review the performance from other instances of E15, or from a process-log web application. The artist and the participants can access and publish feedback about the performance during or after process time.

In the introduction of his book *Notes on the synthesis of form* [32] from 1964, rationalist architect and urban planner Christopher Alexander says his book is about the process of design, which he defines as the process of inventing physical things that display new physical order and form in response to function. When facing a complex design problem, he says, designers rarely confess their inability to solve it. Instead, he adds, they fall back to some arbitrarily chosen formal order. Alexander explains how simple design problems can be outlined as conceptual tension diagrams that are easily solved by mere intuition, but argues that the dynamic complexity of contemporary problems dealing with social interactions are impossible to grasp intuitively. Alexander thinks contemporary times have pushed designers far beyond their cognitive capacity. He suggests that modern mathematics, more concerned about relations than about magnitudes, are an excellent aid to explore the conceptual pattern which a problem presents to a designer.

An important consequence of the mathematical representation of a design problem is that -because it is based on assumptions that are brought into the open- it is easier to criticize than the vague representation formulated after an intuitive approach. Even when intuition can be given a great importance in the design process, the possibility of asking reasonable questions about it should not be excluded.

Almost at the same time, in his book *Designing Programmes* [41] from 1968, graphic designer and typographer Karl Gerstner explores similar ideas. For him, Designing Programmes means inventing rules or arrangement. Even though he starts his work from a visual perspective, it doesn't take him long to transport his ideas about form to the broader fields of architecture and urban planning.

In another book, *A Pattern Language* [27], Alexander opens an illustrated discussion of a pattern language derived from traditional architecture, following a set of rules that are invoked by circumstances, in a form that a theoretical mathematician or computer scientist could call a generative grammar. The work originated from an observation that many medieval cities are attractive and harmonious, and is designed to empower any human being to design and build at any scale, pointing towards a sort of rationalist participatory model for architecture and urban planning.

Alexander's discussion of how to approach design problems is useful for thinking about the separation between art and design. How much are they different? Do they intersect at all? When I think about making a space for the communication of the creative process of making digital art, am I thinking about that space itself as digital art? Performing process can become art itself, but the studio space will be created in the digital medium, becoming a design subject. The design of the studio space will pose clear communication and participation problems. An reasonable outline to follow is an interactive system with a definite function: to remotely connect artist and audience through the performance of process in an artificial telematic studio.

Even when used as a determinant of the design process, Logic is not a determinant of form. The patterns uncovered by a rational approach are meant to uncover conceptual relations that might help justify specific decisions the designer can take when choosing how to deal with a particular problem. In the case of this thesis, an intuitive set of functional principles will define what the studio system should be by describing what it should do:

- The studio system should help the artist control his craft (writing interactive graphics code) better (by the creation of logs and opening process to to feedback), and

- the studio system should help the audience understand digital art and the artist better (by experiencing process and having access to feedback).

These functional principles will be balanced against a set of methodological rules that will be described in the following subsection. From now on, I will refer to the digital studio system as **MyStudio**: An artificial studio to perform process.

3.4.1 Methodology

When writing about his theory of messages in *The Human use of Human Beings* [80], Norbert Weiner explains that messages are themselves a form of pattern and organization. He claims that it is possible to treat a set of messages as having an entropy like sets of states of the external world. Just as entropy is a measure of disorganization, the information carried by a set of messages is a measure of organization, making it possible to interpret the information carried by a message as essentially the negative of its entropy, and the negative logarithm of its probability: "the more probable the message, the less information it gives" [80]. Making an entirely aesthetic judgement, Weiner claims this is the reason why clichés are far less illuminating than poems. But are they?

It is not hard to disagree with Weiner's comparison between clichés and poetry. He jumps with no warning from the quantitative realm of information to the qualitative realm of art appreciation, and doesn't seem to take account of the beholder as a creative agent that adds meaning to the piece of art. To Weiner, it seems the whole meaning was already there, and there are reasons why he must have thought that way; for Weiner, communication and control are defined in terms of each other, and messages are viewed as imperatives that must not lead much space for ambiguity or interpretation. In scientific fields like Biology, control makes important sense because a control failure in a biological system often leads to the eventual termination of its life cycle. Should the same value be attributed to control when applied to a human social system? If so, where should control come from? Are humans capable of designing good control systems to direct other humans?

In **MyStudio**, implicit control takes place in the form of the networking technologies that make communication possible (Internet and the WWW), and explicit forms of control will be avoided as possible, because the higher level communication model -happening between the artist and his audience- should mimic as possible the power dynamics of a natural human conversation, where imperative messages are usually not the norm. It might be argued that the Publishing model I chose already disables conversation due to its asynchronous nature, but I am talking here about power dynamics, and not about the communication model itself. Even if communication happens in a form different from a synchronized conversation model, power dynamics can still be presumed to be flat, giving equal publishing access to all parts, and making sure imperatives of control are conducting the interaction, except when they are needed to manipulate the interface components in the machine.

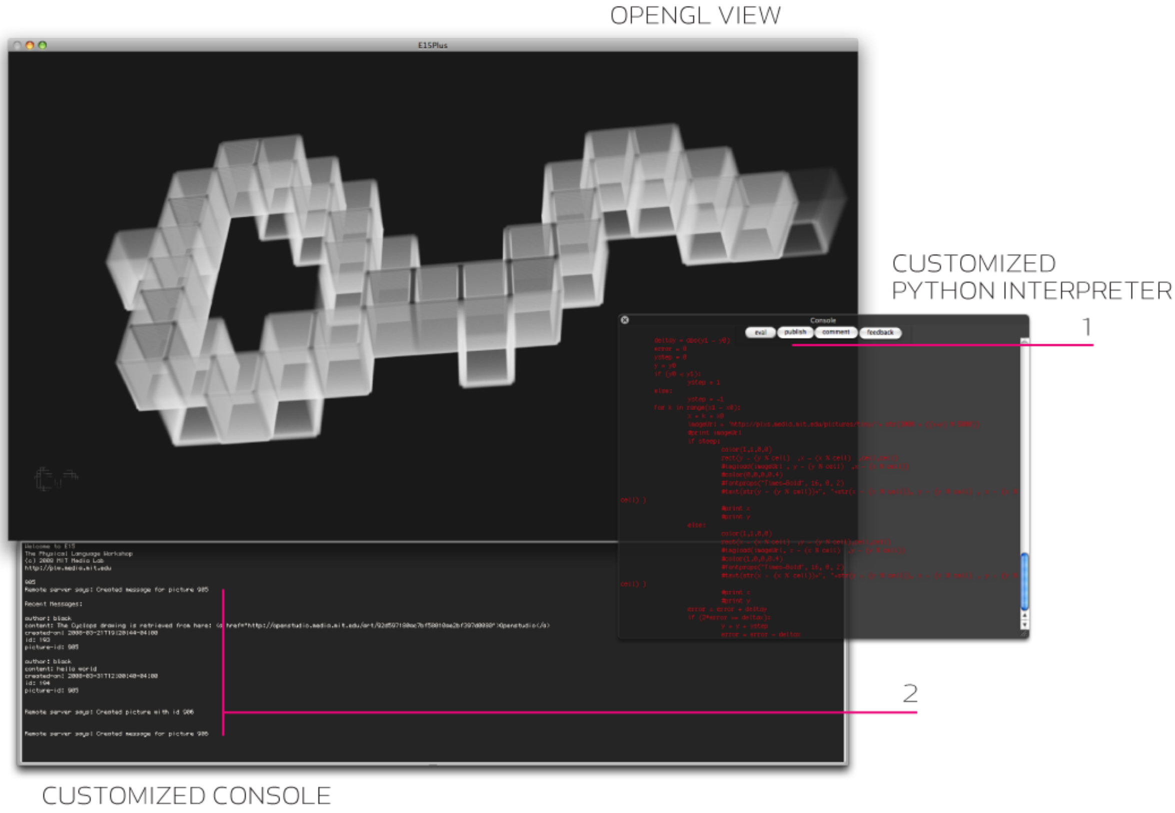
To find a form that can respond to the required functions that **MyStudio** must perform, I chose two methodological principles that work in opposition to each other:

- The antiNorm.
- Repetitive Achievement.

The antiNorm principle states the following: Avoid or ignore whatever standard you don't need to follow, as long as it's not worth being the subject of the Repetitive Achievement principle. This is perfectly suited for top-level concepts, that don't usually determine technological functionality and are merely a matter of representation. This is the principle I would apply when refusing to include a user model in the interaction design of my websites. The webpage standard, however, is harder to avoid or ignore within the contest of the web browser, because it is all there is, but that's where E15 comes in handy =).

The Repetitive Achievement principle states that anything that has already been made is a candidate to be remade from scratch, as long as it doesn't violate the antiNorm principle. In this case, implementing Perlin Noise [61] [60] [62] or building a Ruby on Rails web based picture aggregator both apply the Repetitive Achievement principle, in the sense that both have already been built hundreds of times, but building them again sheds light on unexplored applications for both.

Regarding hierarchies within an art communication system, I think the fundamental norms to observe are those of authorship and appreciation. As it has already been discussed, the evolution of these norms through the 20th century is a point of tension that has manifested as a dislocation of value. Older norms that defined the artist and his art as an imperative form have been displaced by norms that foster participation and blur the line where authorship is drawn. However, the art system still defines itself as a closed, self-regulated elite that exercises an opaque mediation between artists and their audiences. The intention to open a direct link between those who make art and their subjects of appreciation applies the antiNorm principle to the art system, and the Repetitive Achievement principle to the process of making art. Once the social space for art is redefined, how will the need for art manifest when the space for appreciation becomes one with the space for process?



```
    deltay = abs(y1 - y0)
    error = 0
    ystep = 0
    y = y0
    if (y0 < y1):
        ystep = 1
    else:
        ystep = -1
    for k in range(x1 - x0):
        x = k + x0
        imageUrl = 'http://pixs.media.mit.edu/pictures/tiny/'+ str(3000 + ((x+y) % 5000))
        #print imageUrl
        if steep:
            color(1,1,0,0)
            rect(y - (y % cell)  ,x - (x % cell)  ,cell,cell)
            #imgload(imageUrl , y - (y % cell)  ,x - (x % cell))
            #color(0,0,0,0.4)
            #fontprops("Times-Bold", 16, 0, 2)
            #text(str(y - (y % cell))+", "+str(x - (x % cell)), y - (y % cell) , x - (x %
cell) )
            #print x
            #print y
        else:
            color(1,1,0,0)
            rect(x - (x % cell)  ,y - (y % cell),cell,cell)
            #imgload(imageUrl, x - (x % cell)  ,y - (y % cell))
            #color(1,0,0,0.4)
            #fontprops("Times-Bold", 16, 0, 2)
            #text(str(x - (x % cell))+", "+str(y - (y % cell)), x - (x % cell) , y - (y %
cell) )
            #print x
            #print y
        error = error + deltay
        if (2*error >= deltax):
            y = y + ystep
            error = error - deltax
```

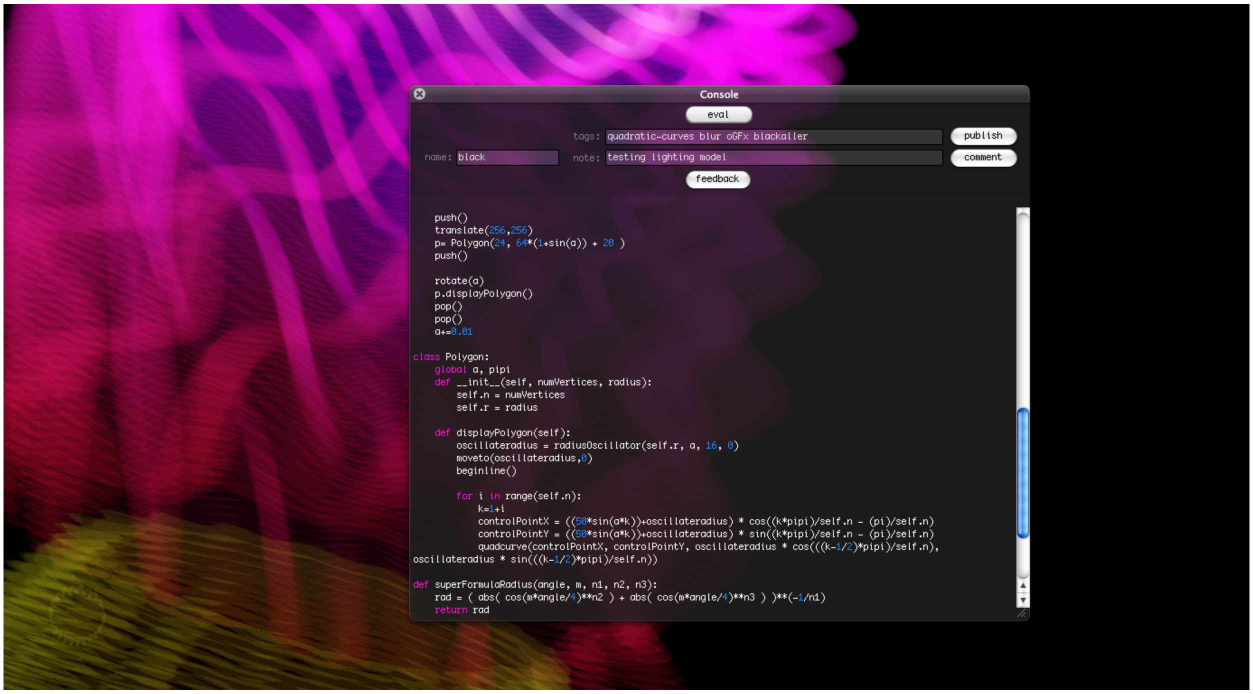
```
Welcome to E15
The Physical Language Workshop
(c) 2008 MIT Media Lab
http://plw.media.mit.edu

985
Remote server says: Created message for picture 985

Recent Messages:

author: black
content: The Cyclops drawing is retrieved from here: <a href="http://openstudio.media.mit.edu/art/92d597180ac7bf58010ae2bf397d0080">Openstudio</a>
created-on: 2008-03-21T19:20:44-04:00
id: 193
picture-id: 985

author: black
content: hello world
created-on: 2008-03-31T12:00:40-04:00
id: 194
picture-id: 985


Remote server says: Created picture with id 986


Remote server says: Created message for picture 986
```

Fig. 3-4: E15 custom components. 1: Code, snapshots and comments are sent to the studio log using buttons or custom methods embedded in the script. The same resources are used to check for feedback on specific items. 2: Feedback is printed in the console. Additional commands allow the artist to fetch feedback about specific items and post in the console.

3.4.2 Resources

The resources available to **MyStudio** will be provided by a customized version of the E15 [4] development environment and the Ruby on Rails [20] web development framework.

In short, E15 can be described as a Python interpreter [17] that controls an OpenGL 3D graphics engine [83] [12] connected with a Cocoa [44] WebKit engine [25]. Python is dynamic object-oriented programming language, OpenGL is a computer graphics develop-



tags: quadratic-curves blur oGFx blackaller

name: black

note: testing lighting model

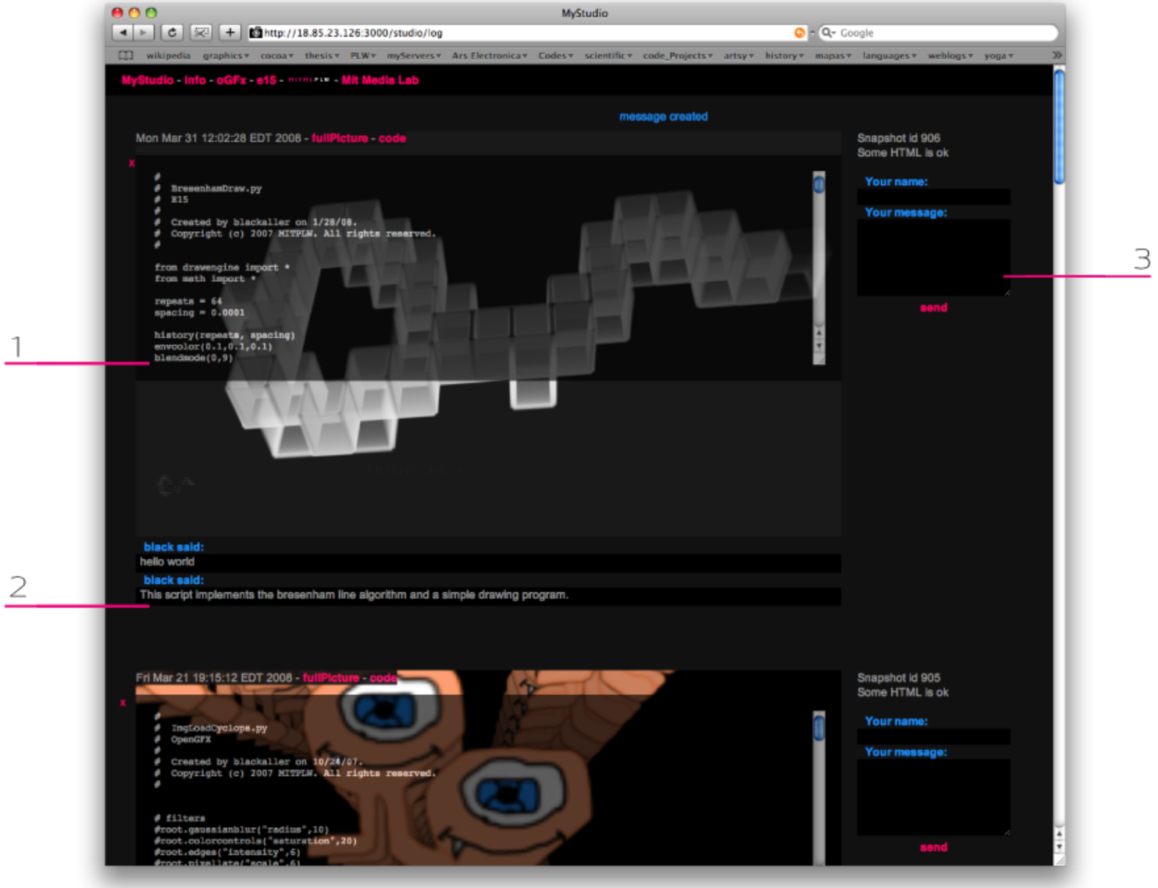
```
    push()
    translate(256,256)
    p= Polygon(24, 64*(1+sin(a)) + 20 )
    push()

    rotate(a)
    p.displayPolygon()
    pop()
    pop()
    a+=0.01

class Polygon:
    global a, pipi
    def __init__(self, numVertices, radius):
        self.n = numVertices
        self.r = radius

    def displayPolygon(self):
        oscillateradius = radiusOscillator(self.r, a, 16, 0)
        moveto(oscillateradius,0)
        beginline()

        for i in range(self.n):
            k=1+i
            controlPointX = ((50*sin(a*k))+oscillateradius) * cos((k*pipi)/self.n - (pi)/self.n)
            controlPointY = ((50*sin(a*k))+oscillateradius) * sin((k*pipi)/self.n - (pi)/self.n)
            quadcurve(controlPointX, controlPointY, oscillateradius * cos(((k-1/2)*pipi)/self.n),
oscillateradius * sin(((k-1/2)*pipi)/self.n))

def superFormulaRadius(angle, m, n1, n2, n3):
    rad = ( abs( cos(m*angle/4)**n2 ) + abs( cos(m*angle/4)**n3 ) )**(-1/n1)
    return rad
```

Fig. 3-5: E15 custom interpreter window, detail. The background window is part of the OpenGL view where the graphics specified by the script are generated.

ment environment, and WebKit is an open source web browser engine. The main purpose of E15 is to provide a platform for experimenting with new interaction experiences in the context of the WWW, but it can be used to model interaction with all kinds of data or other communication platforms. E15 is not a solution to propose a new kind of web browser; it is more like a futurist's tool to explore ideas about what the Web might become.

In this context, **MyStudio** is a simple example to imagine how creative coding can take place in the space of a new Web.

In Web terms, E15 is a client just like any browser, but it doesn't force any conceptual or formal commitments on the user. In other words, the user can choose, or even invent, how to represent the data. Since the Web itself is outside of E15, server side functional-



MyStudio
http://18.85.23.126:3000/studio/log
wikipedia graphics cocoa thesis PLW myServers Ars Electronica Codes scientific code_Projects artsy history mapas languages weblogs yoga
MyStudio - Info - oGFx - e15 - Mit Media Lab
message created
Snapshot id 906
Some HTML is ok
Your name:
Your message:
send
Snapshot id 905
Some HTML is ok
Your name:
Your message:
send
1
2
3

Mon Mar 31 12:02:28 EDT 2008 - fullPicture - code

```
#
#  BresenhamDraw.py
#  E15
#
#  Created by blackaller on 1/28/08.
#  Copyright (c) 2007 MITPLW. All rights reserved.
#

from drawengine import *
from math import *

repeats = 64
spacing = 0.0001

history(repeats, spacing)
envcolor(0.1,0.1,0.1)
blendmode(0,9)
```

black said:
hello world

black said:
This script implements the bresenham line algorithm and a simple drawing program.

Fri Mar 21 19:15:12 EDT 2008 - fullPicture - code

```
#
#  ImgLoadCyclope.py
#  OpenGFX
#
#  Created by blackaller on 10/24/07.
#  Copyright (c) 2007 MITPLW. All rights reserved.
#


# filters
#root.gaussianblur("radius",10)
#root.colorcontrols("saturation",20)
#root.edges("intensity",6)
```

Fig. 3-6: Web components. 1: Scripts and pictures are logged together, making it easy to keep track of the exact code that produced a specific effect. 2: Comments are attached next to the published content. Comments can be posted from the web browser or directly from e15. 3: Form for comments. Authorship is optional.

ity needs to be provided to monitor and control communications, and for this reason it is necessary to develop Web applications that can serve and store data for a network of clients. Ruby on Rails provides very good resources to do that, offering database connectivity and RESTful (Representational State Transfer) services [38] that can serve page requests in a number of formats (JSON, XML, YAML, TXT, HTML) just by asking for a different file extension.

3.4.3 Implementation

The system implementation follows the Model-View-Controller [40] pattern outlined by the Ruby on Rails framework as much as the E15 architecture permits. Using REST, it is possible to serialize and transmit XML representations of the database model that are easily converted into Cocoa objects by the studio system. The problem is to design a good database schema, that can both respond to the Web and the E15 components' needs, and a User Interface design that can represent these needs in the best possible manner.

In the case of the Web, the functional aspect of the problem is already solved. There are many standards available for interactive visual representations of logs and aggregators. Ruby on Rails borrows from all of them, as they all borrow from the structure of relational databases like MySQL. Blogs, dynamic picture galleries, and social networks are all shaped in terms of a structure of tables and relations between their fields. At the same time, as much as relational databases determine the form of user-generated-content dynamic web applications, their time scope is determined by the internet asynchronous communication model. The implementation inside of E15 will mirror this pattern, where queries to the database model will be converted into Cocoa object models after the completion of an HTTP request. Abstract representations of sessions, submissions or messages will be packaged together with their relationships, ready to be summoned by the artist or the participants at play.

On the Cocoa side, a session class will hold a dictionary of snapshots and related messages (both are their own class). The snapshot class encapsulates the picture-code pair, plus extra data like time-stamps and tags. Over time, the studio controller will check for new pictures or messages to add to the dictionary, flagging pictures with unread messages. The artist can see this way where communication has been sent. The user interface implementation can be enhanced with a picture browser, to look for comments on different snapshots, or to choose which particular snapshot should be attached to the next outgoing message. During session time, communication between artist and participants will happen at most in a pseudo-conversation mode. The following are it's main characteristics:

- Asynchronous.
- Non-linear.

Because of time constraints and technically challenging issues, some features from the previously outlined implementation are still not fully functional, in particular the user interface picture browser in the Cocoa application. However, this is an interaction component that does not impair the communication model. The main components of the communication model have all been implemented, and listed here:

- Ruby on Rails Web application for logging and Web based communication.
- Implementation of Cocoa HTTP connection delegates.
- Implementation of Cocoa session, snapshot and message models.
- Implementation of Cocoa controller mechanisms to send snapshots and check for messages.
- Customization of E15 console and E15 Python interpreter to interface with the previously mentioned controller mechanisms

There is an important observation to make between the experience of programming with larger frameworks like Ruby on Rails and Cocoa, or programming in a sandbox environment like Processing, Design by Numbers, Quartz Composer or any other. Programming a real world system seems to be more about abstraction, representation, relational structures and communication, as opposed to sketch or prototype programming in a sandbox environment, which is a matter of computation and rules. Reality perhaps lies somewhere between both worlds, and the development of environments like E15 -where the sandbox is blurred away, and both worlds coexist- might be a fundamental first step towards a deeper evaluation of what the digital medium might become.

4

Analysis

Analysis is the process of breaking down a complex topic into smaller parts to achieve a better understanding of it. In the case of this thesis, the subject of analysis is an interactive human feedback system, that consists of an artist and a group of participants sharing an artificial space where the artistic process is meant to happen. The way to break down such a system will determine the strategies to evaluate what happens in it. Because this is a hybrid system that consists of natural and artificial layers (humans and the machine), it is important to clarify a first level of separation, where it makes sense to analyze the whole system at a natural level, considering the artificial components as if they were seamlessly transformed into the natural things they are supposed to represent. For example, at some level it makes sense to forget that a participant inputs a message into a computer that delivers this message to another computer, where it is made accessible to the artist, and just think instead that a participant sent a message to the artist. It all happens in the artificial studio space. However, when analyzing the design decisions made to facilitate these interactions, it will be of relevance to reconsider the "true nature" of the artificial studio space, and think back about it as the networked computerized communication system it is.

4.1 Conceptual Analysis

The studio space will be divided in functional layers in terms of explicit and implicit actions. The explicit actions are the top level representations that the artist and the participants will use to communicate between each other. Norbert Weiner's observation of the implicit physiological action that coordinates brain, eyes and muscles to make someone draw, points out the fact that this action is unknown to the person drawing. The same way, artist and participants should not be aware of what the system is doing to keep them communicated. The functions of these implicit machinery, which are relevant in terms of design and structures of control, should be kept in a separate layer, where humans are mere input to the machine's process. In this terms, the layer division of the studio space is made as follows:

- Performance or explicit layer: human to human interactions, studio space, computational art.
- Implementation or implicit layer: interaction design, control structures.

A conceptual analysis will be used as a starting point to frame the constitutive elements of the performance layer, where it is easy to determine the different components as a set of participants (artist and friends), the space where they communicate (an artificial studio), the content of their communication (code, pictures, natural language), and the interactive art itself.

4.1.1 Artist and Participants

The relationship between artist and participants is what holds the system together. The other components are consequences or derivatives of these two atomic components. The art is made by the artist because art is what the artist does, and the process of making art needs a space to happen that we call the studio. These observations make it immediate that the artist is an atomic concept from which the art and the studio are immediate ontological reductions. The participants, however, as an instance of the more general beholder, are independent from the artist and his art. Even though it is true that there is an implicit need for the beholder's gaze in the creative impulse of the artist, it is the same artist who plays the role of the beholder when he falls back to a contemplative mode in the transition between steps within the creative process. Art can be created and appreciated only by the same person and still be art, and for this reason, the participants play the role of an independent atomic concept that is not determined by any other component in the system.

Communication between the artist and the participant is composed of two different kinds of messages. The first kind only goes one way, from the artist to the participants, and it's made of two different projections of the interactive graphics the artist is working with: the code generating the graphics, and a snapshot of the graphics at a given moment in time. Together they represent a perception of what the artist is doing, but they are not it. At the same time, there is a distinction between 2 kinds of participants, the ones that have access to the E15 development environment and can copy the code to run their own interactive graphics, and the ones that can only experience the code-picture projections in the web browser. In both cases, all the participant experiences is a window to the studio. The studio concept can't include the artist and the participants in the same space, because they can't directly experience each other's actions. Their communication is mediated by an asynchronous channel, and as such, it gives each one of them their own timeline. However, the fact that code can be seamlessly transmitted and run once received, helps construct the illusion of a shared experience that succeeds in communicating the art in progress with a total degree of fidelity. On the other hand, the actual performance of the artist (his own physical gestures) will inevitably get lost.

An important issue to discuss is the justification for separating the artist from the rest of the participants as a different kind. Is this division the core of authorship? As initiator and performer, the artist plays two roles that are not played by the rest of the participants. The performance role doesn't need to be exclusively claimed as essential to the artist, because it is easy to imagine scenarios where other participants will perform their participation as well, but the role of initiator can't be as transitive. It could be so that the means are provided for other participants to start their own process, in a space that could function as a kind of sponge of studios, where different artists borrow from each other and initiate their own performance process to different sets of participants.

The role of the artist plays a social activation role, and separates itself from the traditional category of authorship. This makes sense in terms of the current state of affairs, where process takes precedence over object as the result delivered after artistic practice. Because of this, time based analysis should take precedence over spatial analysis, choosing to look at relations between events over time rather than between objects laid out in space.

4.1.2 Time over space

Local physical space and time are only one instance of the intricate web of locations and time-zones that delineate the artificial studio space. In spite of the asynchronous model of communication that lets every participant run their own clock independently from ev-

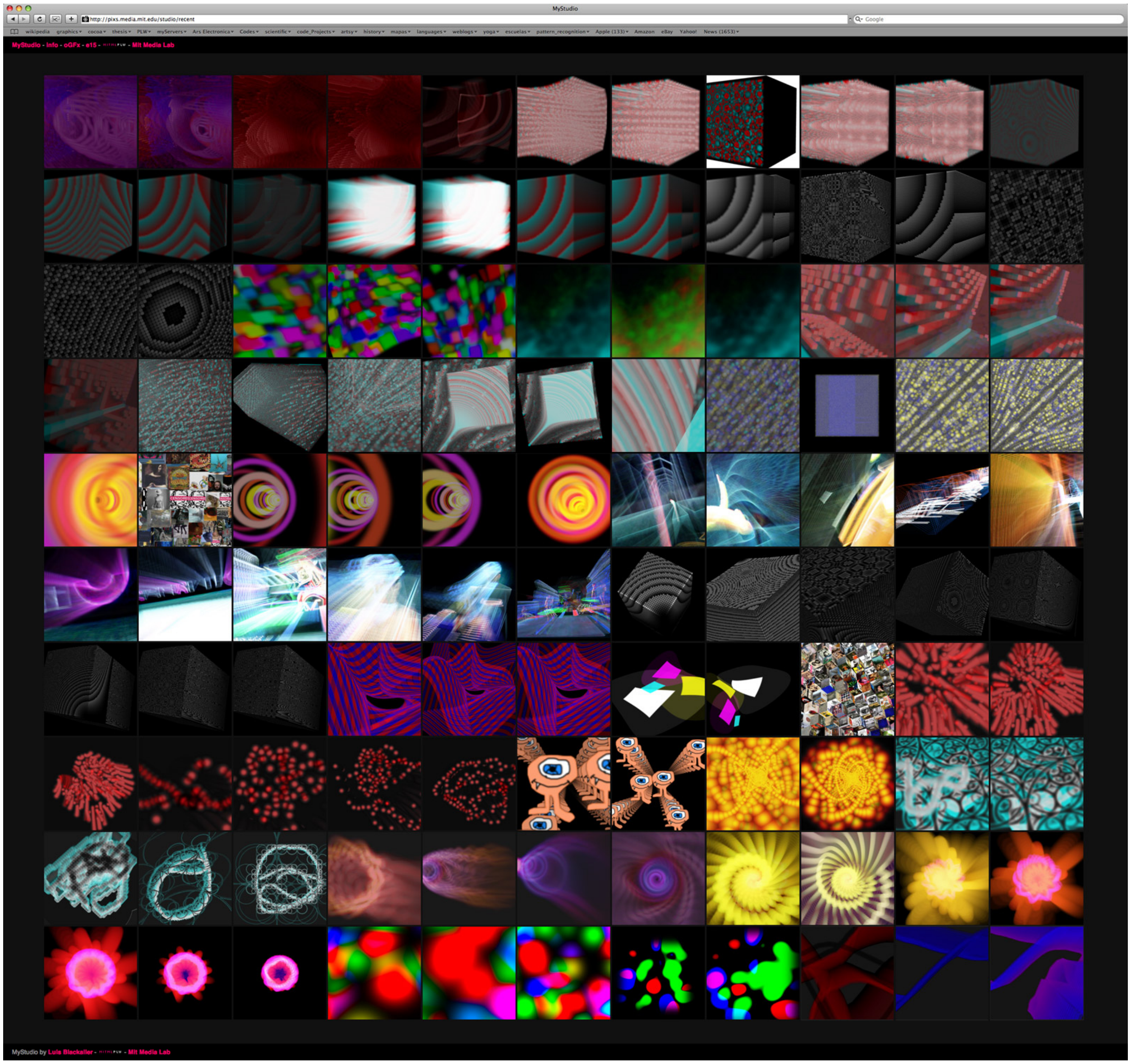

MyStudio
http://pixs.media.mit.edu/studio/recent
wikipedia graphics cocoa thesis PLW myServers Ars Electronica Codes scientific code_Projects artsy history mapas languages weblogs yoga escuelas pattern_recognition Apple (133) Amazon eBay Yahoo! News (1653)
Google
MyStudio - Info - oGFx - e15 - - Mit Media Lab
MyStudio by Luis Blackaller - - Mit Media Lab

Fig. 4-1: Time over space: First 110 submissions in the online studio snapshot log. The picture component becomes a sort of visual timeline. Different sessions can be identified by looking at the rate of change between consecutive pictures. Like in a movie, cuts from one scene (or session) to the other differ from smoother transitions, where analogies of shape or color maintain a visual coherence. However, visual jumps are present even within the same session.

erybody else, space is the one that still experiences most fractures, not only because of the remote location of every participant, but also because it is infinitely more notable how space is compressed into the digital representation than time. Modern computers are fast enough to deliver an illusion of the continuum of time. Even though digital time is discrete, it has enough granularity for humans not to notice the empty steps between one calculation and the other. Space, however, changes from a rich three dimensional and tangible experience in the physical world to an immaterial world where 3 dimensions or less are simulated, only observed through the 2-dimensional window of the computer screen. This and other factors like the precision of computer memory, the ability to reproduce digital content anywhere at any time with no loss of information, and the fact that process in general is a time-based concept, make time an ideal axis to map against other components of analysis. Different timeframes are easily located within the lifespan of the system, keeping the centralized HTTP web server component active even when the artist is not performing, and leaving channels open for communication about events that were performed in the remote past, or to review this kind of events actively by running the recorded code. When a given scenario is set for analysis, it will be important to look at these two non-overlapping kinds of intervals in time: Performance time and its complement, both determining different reactions and modes of participation.

4.1.3 Computational art

The only way to get feedback is by asking for it, and in this case, the process of writing code for computational art will be the initial input to set the studio system running. The oGFx component of the E15 development environment is an excellent tool for the exploration of new forms of interactive computational art through the following levels of interaction:

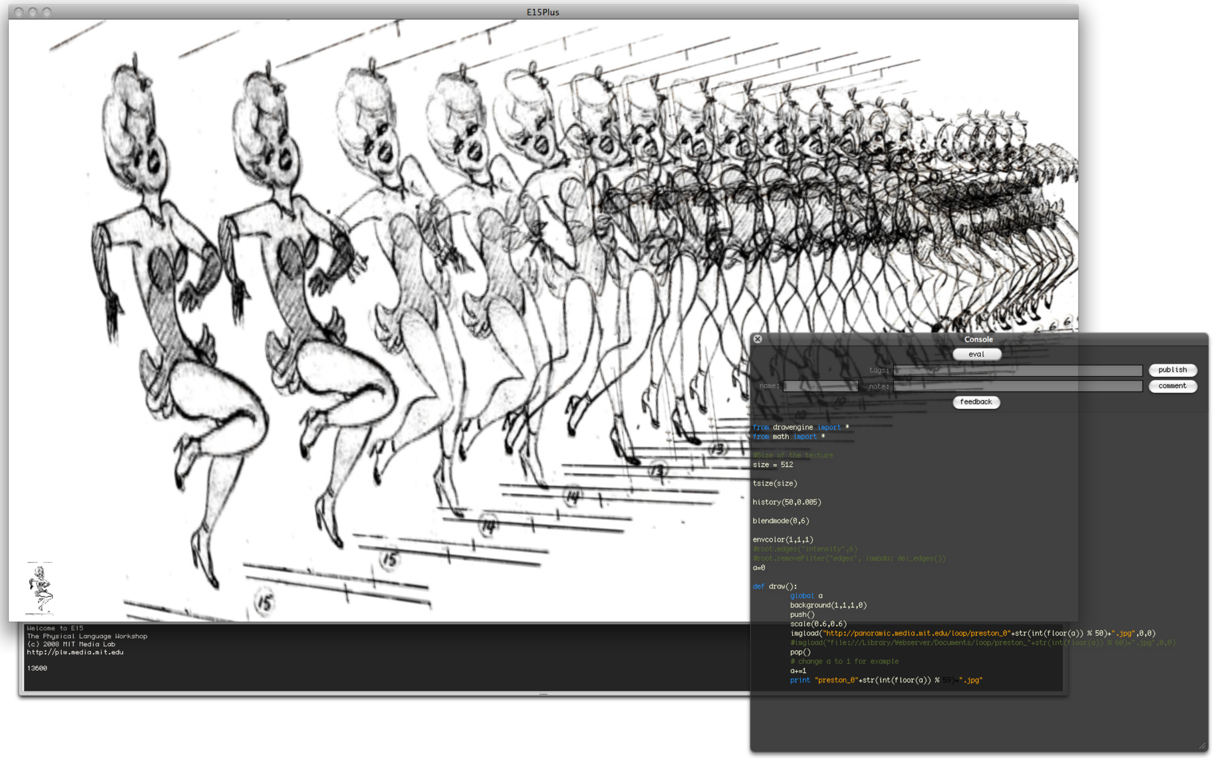
- Interaction with computer graphics in 3D space.

- Interaction with the interpreted Python code can change the structure of the code that is being evaluated.
- Interaction with the World Wide Web, making it accessible to use Web data as a material to render new graphics.

Overall, the kinematic, generative, behavioral, and interactive properties usually associated with computational art can be dynamically influenced by incoming feedback, both from the participants' perception of the actual graphics, and from the participants' message pool. The creative process happens in parallel with a computational process that turns instructions and data into visual input for the participants and the artist himself.

Today, it is not difficult to start a parallel process in the computer that monitors the principal process, and manipulates the original computer time as if it was a new instance of space. In order to look at the evaluated script's process in time as a physical dimension in space, the "history" command was incorporated to the oGFx API, allowing to store a requested variable number of frames from the generated animation into a buffer, rendering them as a sequence in space as the animation runs. The effect, well known by kinematic visual art practitioners since the dawn of animation, becomes an interesting practical tool to study motion (and dynamic process) once it has been set to be manipulated while it is in motion.

Unfolding computational process into higher dimensions is opposite to the experience delivered by the studio system, where only a static visual slice and the corresponding code recipe will be delivered. This projection serves two roles, the first and most important is to communicate steps on the evolution of the coding process performed by the artist. The second is a version log for process, that can eventually let the artist -or other participants- roll back to transitional states of process, recovering effects or resources that could have been lost otherwise, as well as providing a more permanent space for the discussion of issues regarding the artist's creative process.



```
from drawengine import *
from math import *

#Size of the texture
size = 512

tsize(size)

history(50,0.005)

blendmode(0,6)

envcolor(1,1,1)
#root.edges("intensity",6)
#root.removeFilter("edges", lambda: del_edges())
a=0

def draw():
        global a
        background(1,1,1,0)
        push()
        scale(0.6,0.6)
        imgload("http://panoramic.media.mit.edu/loop/preston_0"+str(int(floor(a)) % 50)+".jpg",0,0)
        #imgload("file:///Library/Webserver/Documents/loop/preston_"+str(int(floor(a)) % 60)+".jpg",0,0)
        pop()
        # change a to 1 for example
        a+=1
        print "preston_0"+str(int(floor(a)) % 59)+".jpg"
```

```
Welcome to E15
The Physical Language Workshop
(c) 2008 MIT Media Lab
http://plw.media.mit.edu

13600
```

Fig. 4-2: History command: Pencil test frames from 1943 animation classic "Red hot riding hood", animated by Preston Blair and directed by Tex Avery. The drawings were extracted from Blair's animation recipe books [30], loaded into E15, and projected in motion over 3D space. Graphical representations of time are particularly useful to understand motion.

4.2 Analysis of Process

Regarding performance time, I have already distinguished between two parallel processes. First, the human process that iterates versions of written code after reacting to a combination of the feedback from the computer graphics the code generates, and the feedback provided by the participants. Second, the computational process that evaluates the script that is being written, which is also affected by input from the artist every time he changes the script and evaluates it again, redefining the dynamic data structures that are transformed into graphics onscreen. The intimacy of this human-machine relationship is at the core of the creative process, and it can only be fully communicated if the participants have the resources to run the script themselves. The snapshot objects published online are a good anchor to understand the parallel evolution of code and graphics, but it will be hard for anyone to reconstruct in their heads how these graphics will change and interact with a user overtime. In this regard, the snapshot content that is logged online is more directly related to the communication of the artist's process. It describes how visuals and code change as a reflection of each other, but it conveys poorly the richer aspects of dynamic interaction represented by the digital content at full.

However, the same way a photograph can be said to poorly represent a situation experienced by a group of persons in the real world, where time passage, sound, smell and so many other attributes are discarded, the frozen snapshot published online can gain a different value, letting appreciate details in form and expression that would have otherwise been lost in time. Because of this, it can be inferred that immediacy and persistence play the roles of opposites, shedding light over completely different aspects of the digital content, and at the same time reinforcing the multimedia nature of the digital medium, that can be broadcast as an interactive movie and published as a picture-code pair at the same time, expanding itself in parallel to different forms of reading. It is amusing to imagine a world where one could send away a script for a movie knowing that it will be played back as a unique version of it once it is loaded in the movie-making machine.

Immediacy and persistence determine two different timeline modes that must be considered when evaluating the interaction space for process. Even though both of them are non-linear and asynchronous, the immediacy of the first case forces its timeline to be compressed into the scope of the performance time; interactions that happen within this mode are parallel -if not synchronous- to the process, and establish a relationship with it as it happens. because process is being performed, the artist will not have much time to examine the participants' feedback in detail, and will react mainly by instinct to the comments delivered by them.

The second mode opens itself to a virtually infinite timeline, where there will always be time to go back and review what happened, study the feedback content and provide some more. Both modes, however, face a communication problem that is present -and unresolved- in most contemporary Web communication structures, where the trend seems to be a fixation with the new, and discussions are stuck inside each submission, without it being possible to link between different conversations, that should sometimes be linked through the sequence of submissions.

4.2.1 Initial conditions

The relevant initial conditions to start an evaluation run of the studio system are three:

- Artist.
- Artist's intent.
- Participants.

The only reliable condition to evaluate is the artist's intent, which can be determined by an initial concept to explore and communicate through the development of code. The artist is an important variable that escapes formal analysis, because even when following the same intent he will take different approaches based in his personal state, that could even include the knowledge acquired from having followed the same intent some other time before.

The participants' feedback is not a reliable metric because it depends more on the artist's perception than on the actual participant. As John Maeda pointed out, a robot could generate feedback in ways that make perfect sense to the artist, but that don't reflect anyone's impression on the artist's work. Even though participation is a main goal to the development of this project, it can't be fully evaluated unless there is a reasonable strategy to map the participant's profiles, which is impossible in an anonymous system like this one.

However, it is valid to assume all participants are human and well intentioned. In this case, an explanation to discard the participants intentions as an important variable to evaluate the studio system is harder to articulate. For example, a participant might want to learn from the artist, enjoy the artist's performance, or try to make the artist work in a direction the participant wants. These intents will determine communication patterns that will have different effects on the artist, but they are not essential enough for consideration because it is the artist who has ultimate control, and it is his intent what will allow communication to happen in a certain way. If the artist's intent is to teach about a certain concept, participants that express themselves in favor of a different course of action will redirect the way the expression of that concept is approached, but will not succeed in making the artist abandon his original intent.

Intent is understood as the planning and desire to perform an act, and for this reason the same intent can be taken as a starting point several times. Looking for clarity and simplification, three ideas related to computation process will be taken as inspiration, and the intent will be to elaborate variations around combinations of them, aiming to hide their computational nature under visual forms that can still reveal their origin if examined carefully:

- Grid: matrix
- Oscillation: pulse
- Loop: tangle

The starting point for each case is defined by the same simple setup script -consisting of a few lines of code- over which variations can be performed and published. The course to take with this variations is determined by the artist's first intent and his response to feedback. An intrinsic limitation of the chosen set of intents is that the participants' feedback doesn't directly affect the performed artwork, it can only influence the artist indirectly, sending messages to communicate what they think or feel about the art. The participants can touch a living copy of the art by running the code themselves, but they can't touch the artist's performance.

4.2.2 Feedback

Due to the extreme flexibility permitted by asynchronous communication, it is possible to mock up scenarios that approximate the two communication models that have not been implemented: broadcasting and conversation. The three recipes of control that can be evaluated are the following:

- Publishing or postponed feedback. Feedback is received after the fact.
- Pseudo-broadcasting or delayed feedback. Feedback is requested when wanted.

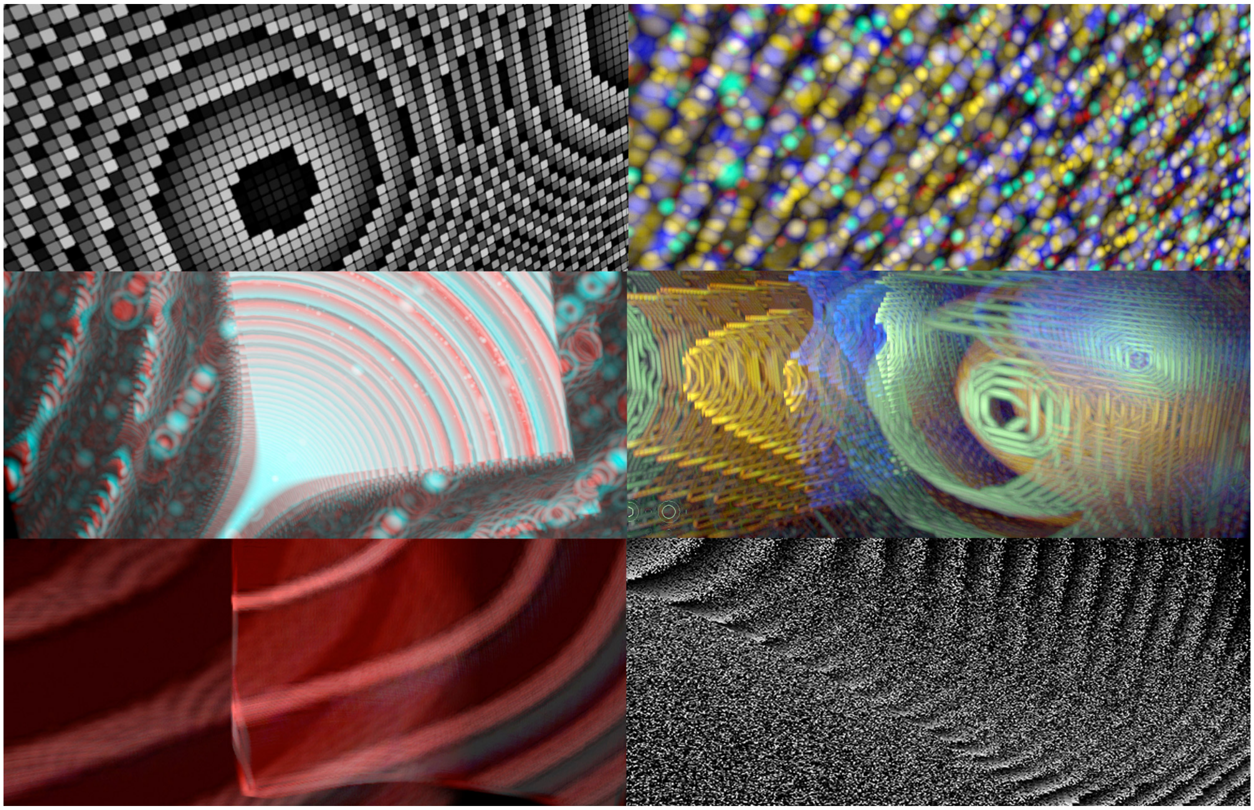


Fig. 4-3: Luis Blackaller. Pulse, 2008, 6 variations of oscillation over a rigid matrix. The 6 images featured in the figure are generated by scripts that differ in nothing more than around 8 lines of code. The scripts are less than 30 lines long. Their apparent simplicity relies on the complex resources supported by the E15 architecture, including state of the art image processing and 3D graphics technologies.

- Pseudo-conversation or immediate feedback. Feedback is received when sent.

In the first case the artist performs unaffected by exterior influences, and has total control over the communication channel. This control remains the same in the second case, but is inverted in the third, where conversation eliminates any attempt to control the feedback frequency.

When performing the three recipes of control over the same intent, it becomes clear that the corresponding outcomes are all different, which means that in the larger timeline, order affects the state of the artist. For example, if we label publishing as P, pseudo-broadcasting as B and pseudo-conversation as C, the sequence of actions PBC will be performed in a very different way than the sequence CBP. In general, because of an implicit knowledge of the sequence order, the artist will most likely take the first step as a preparation for the next ones and this will determine in part the artist's involvement with his performance. In the PBC case, isolated publishing will come first, and the expectation of the future exposed performances might make the artist explore routines that he could repeat later in the presence of a public. The CBP case is perhaps more interesting, because the artist will be forced to expose process without being prepared for it, thus being more prone to miscomprehension. However, this will give him an opportunity to understand the beholder's reaction to his intent, and use this new knowledge to refine the final solitary session of the publishing communication mode.

In other words, this means that the artist can use different degrees of self-control over feedback to learn different things from the participants as a sample of his audience. A system like **MyStudio** collects data, or information, which is different from knowledge. It is up to the artist to understand the reactions from his audience -and his own response to these reactions- in a way that can benefit his own art. Each finished piece can be part of a larger process, that refines the artist's methodology, pointing towards deeper meaning, and hopefully, a larger scope that can eventually touch eternity.

4.2.3 Axes of evaluation

Time, control, and value are ideal axes for mapping an evaluation of the effect the studio system can have on the creation of digital art. Artistic process is deformed through a sequence of control recipes (represented by approximations to the three communication models outlined before) over time. However, value is qualitative as opposed to quantitative, and it is a relative concept that can't be objectively measured. Norbert Weiner's effort to pair aesthetic or artistic value with certain characteristics of information remains unsatisfactory, but my mapping of the artistic process over time permits a degree of speculation in the same direction, perhaps not in terms of value, but by observing singularities or discontinuities along the more or less progressive process of artistic development.

5

Conclusion

In order to face the original questions motivating this thesis regarding the poetics, aesthetics and distribution of digital art, I have proposed to unify them into the same performance action, by taking advantage of the digital medium to open artistic process -usually an isolated act- to participation.

5.1 Result

Following the proposed direction, I have suggested a strategy to incorporate the digital machine as a moderator between the artist and his audience, also serving as a kind of archive that features visual and conversational modes of code classification. If the digital machine becomes good enough to manipulate language and pictures the way humans do, archives pairing procedural code with natural language and images will be of extraordinary value, especially to look back at the behaviors and patterns through which humans develop digital representations of concepts.

In terms of art practice, I have proposed to merge the spaces for the creation and the communication of art, utilizing the current state of digital technology to perform artistic process and open it for participation. Whether an artist will be affected in a positive way depends on a precise definition of art and it social role. If the result of the artistic process is a type of object with an expected quality, then it is reasonable to compare it, but if the result is the process itself, then there will be too many unknowns, and the only certainty left will come from the assumption that any discussion involving unknowns is a good thing.

5.2 Future Work

It is important to find the necessary conditions for a continued refinement of the system developed in this thesis, finishing the implementation of incomplete features mentioned before (like a snapshot browser drawer) and considering the implementation of conversation and broadcasting scenarios. Based on the limitations exhibited by the work developed for this thesis, it is also important to push towards a better integration between digital creative action and communication models. This is the only way the realm of technology can aspire to seamlessly integrate itself with nature.

Bibliography

[1] Bitstrips. [http://bitstrips.com/].

[2] Design by numbers. [http://dbn.media.mit.edu/].

[3] Deviantart. [http://www.deviantart.com/].

[4] E15. [http://e15.media.mit.edu/].

[5] Facebook. [http://www.facebook.com/].

[6] Flat black films. [http://www.flatblackfilms.com/].

[7] Flickr. [http://www.flickr.com/].

[8] Google. [http://www.google.com/].

[9] Google analytics. [http://www.google.com/analytics/].

[10] Max msp. [http://www.cycling74.com/].

[11] Opencode. [http://opencode.media.mit.edu/].

[12] Opengl. [http://opengl.org/].

[13] Perlin zoompad. [http://mrl.nyu.edu/ perlin/experiments/whiteboard/].

[14] Picturexs. [http://pixs.media.mit.edu/].

[15] Plw openstudio. [http://openstudio.media.mit.edu/].

[16] Processing. [http://processing.org/].

[17] Python. [http://www.python.org/].

[18] Quartz composer. [http://www.quartzcompositions.com/].

[19] Rhizome. [http://rhizome.org/].

[20] Ruby on rails. [http://www.rubyonrails.org/].

[21] Sodaplay. [http://sodaplay.com/].

[22] Tiny icon factory. [http://tiny.media.mit.edu/].

[23] Tinydoodle. [http://doodle.media.mit.edu/].

[24] Vvvv. [http://vvvv.org/].

[25] Webkit. [http://webkit.org/].

[26] Youtube. [http://www.youtube.com/].

[27] Christopher Alexander, Sara Ishikawa, and Murray Silverstein. *A Pattern Language: Towns, Buildings, Construction (Center for Environmental Structure Series).* Oxford University Press, 1977.

[28] John Backus. The history of fortran i, ii, and iii. pages 25–74, 1981.

[29] Walter Benjamin. The work of art in the age of mechanical reproduction. In Meenakshi Gigi Durham and Douglas Kellner, editors, *Media and Cultural Studies: Keyworks.* Blackwell Publishers, 2006.

[30] Preston Blair. *Cartooning: Animation 1 & 2.* Walter Foster, 1998.

[31] Benjamin H. D. Buchloh. Conceptual art 1962-1969: From the aesthetic of administration to the critique of institutions. *October*, 55:105–143, 1990.

[32] Alexander Christopher. *Notes on the Synthesis of Form.* Harvard University Press, 1964.

[33] Dave Crane, Eric Pascarello, and Darren James. *Ajax in Action.* Manning Publications, 2005.

[34] Douglas Davis. *Art and the Future.* Praeger Pubishers, 1973.

[35] Guy Debord. *The Society of the Spectacle.* Zone Books, 1973.

[36] Michael Eisenberg. Schemepaint: a programmable application for graphics. *University of Colorado Department of Computer Science Technical Report*, (CU-CS-587-92), 1992.

[37] Douglas Engelbart. Augmenting human intellect: A conceptual framework. *Stanford Research Institute Summary Report*, 1962.

[38] Roy Thomas Fielding. *Architectural styles and the design of network-based software architectures.* PhD thesis, 2000. Chair-Richard N. Taylor.

[39] Alexander Galloway. *Protocol: How control exists after decentralization.* MIT Press, 2004.

[40] Erich Gamma, Richard Helm, Ralph Johnson, and John Vlissides. *Design patterns: elements of reusable object-oriented software.* Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1995.

[41] Karl Gerstner. *Designing Programmes.* Hastings House Publishers, 1968.

[42] Ian Hacking. *The Emergence of Probability: A Philosophical Study of Early Ideas about Probability, Induction and Statistical Inference, second edition.* Cambridge University Press, 2006.

[43] Heraclitus. *The Art and Thought of Heraclitus: An Edition of the Fragments with Translation and Commentary.* Cambridge University Press, 1979.

[44] Aaron Hillegass. *Cocoa Programming for Mac OS X 3rd Edition.* Addison-Wesley Professional, 2008.

[45] Brian W. Kernighan. *The C Programming Language.* Prentice Hall Professional Technical Reference, 1988.

[46] Brian W. Kernighan and Rob Pike. *The practice of programming.* Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1999.

[47] Brian W. Kernighan and P. J. Plauger. *The Elements of Programming Style.* McGraw-Hill, Inc., New York, NY, USA, 1982.

[48] Donald E. Knuth. *The art of computer programming, volume 1 (3rd ed.): fundamental algorithms.* Addison Wesley Longman Publishing Co., Inc., Redwood City, CA, USA, 1997.

[49] Myron Krueger. *Artificial Reality 2.* Addison Wesley, 1991.

[50] Golan Levin. Painterly interfaces for audiovisual performance. Master's thesis, Massachusetts Institute of Technology, 2000.

[51] John Maeda. Maedastudio. [http://maedastudio.com].

[52] John Maeda. *Design by Numbers.* MIT Press, 1999.

[53] James Clerk Maxwell. On governors. *Proceedings of the Royal Society*, 16:270–283, 1867-1868.

[54] John McCarthy. History of lisp. pages 173–185, 1981.

[55] Peter Naur. The european side of the last phase of the development of algol 60. pages 92–139, 1981.

[56] Seymour Papert. A computer laboratory for elementary schools. *MIT-CSAIL Artificial Intelligence Lab Publications AI Memos*, AIM-246, 1971.

[57] Seymour Papert. Teaching children to be mathematicians versus teaching about mathematics. *International Journal of Mathematical Education in Science and Technology*, 3(3):249–262, 1972.

[58] Seymour Papert. *Mindstorms: children, computers, and powerful ideas.* Basic Books, Inc., New York, NY, USA, 1980.

[59] Christiane Paul. *Digital Art.* Thames Hudson, 2003.

[60] K. Perlin and E. M. Hoffert. Hypertexture. *SIGGRAPH Comput. Graph.*, 23(3):253–262, 1989.

[61] Ken Perlin. An image synthesizer. In *SIGGRAPH '85: Proceedings of the 12th annual conference on Computer graphics and interactive techniques*, pages 287–296, New York, NY, USA, 1985. ACM.

[62] Ken Perlin. Improving noise. In *SIGGRAPH '02: Proceedings of the 29th annual conference on Computer graphics and interactive techniques*, pages 681–682, New York, NY, USA, 2002. ACM.

[63] Alan J. Perlis. The american side of the development of algol. pages 75–91, 1981.

[64] Frank Popper. *From technological to virtual art.* MIT Press, 2007.

[65] Casey Reas. Software structures: A text about software art. [http://artport.whitney.org/commissions/softwarestructures/text.html].

[66] Casey Reas and Benjamin Fry. *Processing a programming handbook for visual designers and artists.* MIT Press, 2007.

[67] Michael Rush. *New Media in the late 20th-century Art.* Thames Hudson, 1999.

[68] W.R. Sabiston. Extracting 3d motion from hand drawn animated figures. Master's thesis, Massachusetts Institute of Technology, 1991.

[69] Herbert A. Simon. *The Sciences of the Artificial.* MIT Press, 1969.

[70] Karl Sims. Artificial evolution for computer graphics. *Computer Graphics, Annual Conference Series, SIGGRAPH '91 Proceedings*, 25(4):319–328, 1991.

[71] Karl Sims. Evolving 3d morphology and behavior by competition. *Articial Life IV Proceedings*, pages 28–39, 1994.

[72] Karl Sims. Evolving virtual creatures. *Computer Graphics, Annual Conference Series, SIGGRAPH '94 Proceedings*, pages 15–22, 1994.

[73] Gerald Jay Sussman and Harold Abelson. *Structure and Interpretation of Computer Programs, 2nd Edition.* MIT Press, 1996.

[74] Ivan Edward Sutherland. *Sketchpad: A man-machine graphical communication system.* PhD thesis, Massachusetts Institute of Technology, 1963.

[75] Dag Svanaes. Kinaesthetic thinking: The tacit dimension of interaction design, computers in human behavior. *Norwegian Perspectives on Computing in Complex Domains*, 13(4):443–463, 1997.

[76] Dave Thomas and David Hansson. *Agile Web Development with Rails Second Edition.* Pragmatic Bookshelf, 2006.

[77] Jr. Thomas J. Bergin and Jr. Richard G. Gibson, editors. *History of programming languages—II.* ACM, New York, NY, USA, 1996.

[78] Alfred R. Wallace. On the tendency of varieties to depart indefinitely from the original type. In Jane R. Camerini, editor, *The Alfred Russel Wallace reader: a selection of writings from the field.* Johns Hopkins University Press, 2002.

[79] Norbert Weiner. *Cybernetics or the Control and Communication in the Animal and the Machine.* MIT Press, 1943.

[80] Norbert Weiner. *The Human use of Human Beings: Cybernetics and Society.* Da Capo Press, 1950.

[81] Norbert Weiner. *Nonlinear Problems in Random Theory.* MIT Press, 1966.

[82] Richard L. Wexelblat, editor. *History of programming languages I.* ACM, New York, NY, USA, 1981.

[83] Mason Woo, Jackie Neider, Tom Davis, and Dave Shreiner. *OpenGL Programming Guide: The Official Guide to Learning OpenGL, Version 1.2.* Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1999.