HOW HAS SOFTWARE AFFECTED THE VISUAL ARTS?

WHAT IS THE POTENTIAL FOR SOFTWARE WITHIN THE VISUAL ARTS?

AS A DESIGNER OR ARTIST, WHY WOULD I WANT OR NEED TO WRITE SOFTWARE?

Software influences all aspects of contemporary design and visual culture. Many established artists, such as Gilbert and George, Jeff Koons, and Takashi Murakami, have totally integrated software into their processes. Numerous prominent architects and designers use software extensively and commission custom programs to realize their ideas. The creators of innovative video games and Hollywood animated films also write software to enhance their work.

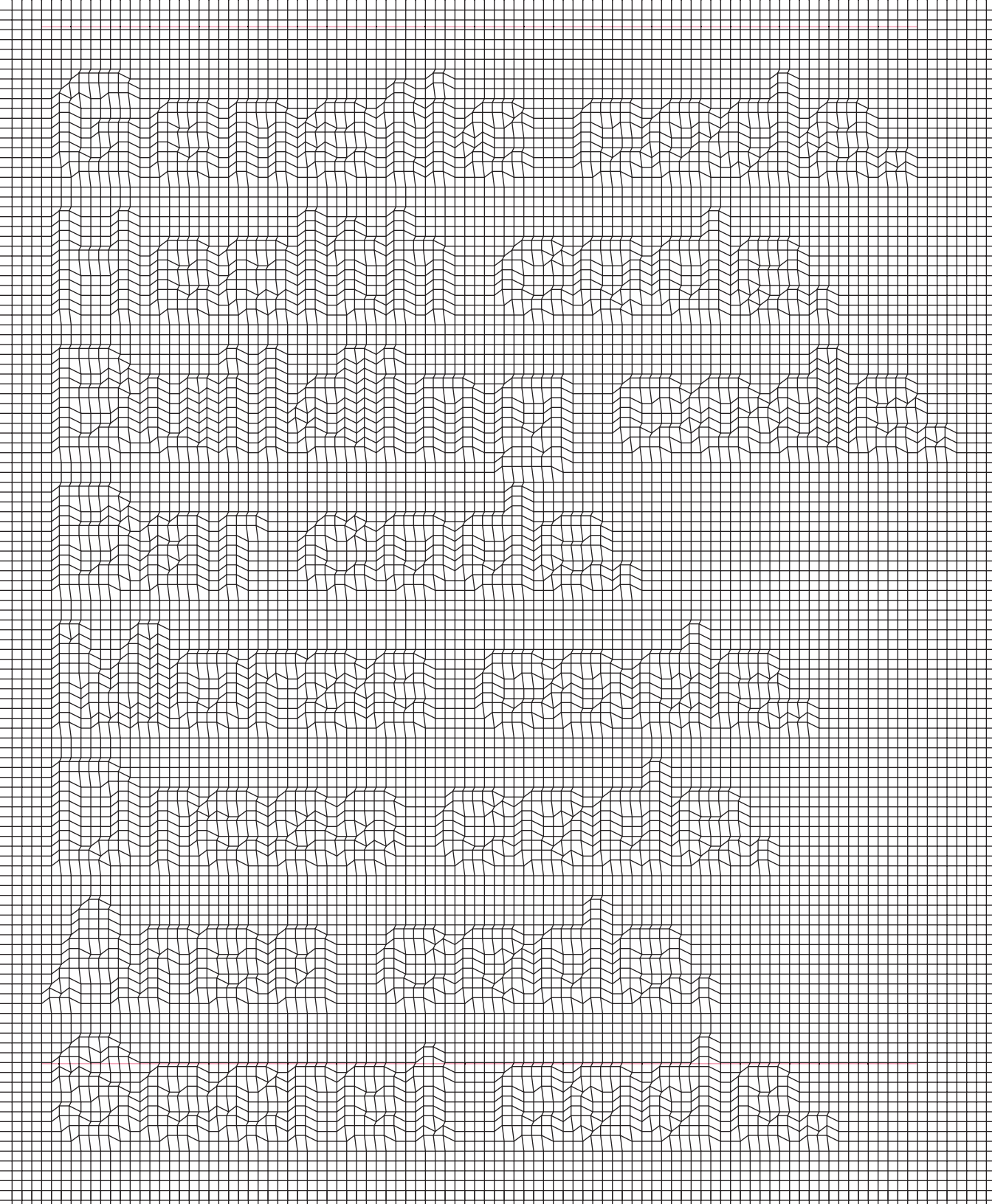
While these exciting developments are taking place at the highest levels of the creative professions, integrating them into design education is a challenge. Even the most motivated student will find the technical boundaries difficult to overcome. As a comprehensive first introduction to software development within the arts, this book seeks to encourage the enthusiasm that the field requires. It will not, however, teach you to program computers. To satisfy that urge, see Processing: A Programming Handbook for Visual Designers and Artists by Casey Reas and Ben Fry.*

In Form+Code in Design, Art, and Architecture we define form as visual and spatial structures; code is defined primarily as computer programs, but we extend the definition to include instructions beyond computer code. The book is organized into seven chapters: What is Code?, Form and Computers, Repeat, Transform, Parameterize, Visualize, and Simulate. The first two chapters set the foundation by defining terms and introducing basic concepts. The themed chapters that follow are deeply linked to code. Each begins with an essay to define the territory, continues with images and captions to clarify and explain each theme, and concludes with two illustrated examples of programs. The corresponding source code is available in multiple programming languages and can be downloaded for free from the book's website: http://formandcode.com.

We're tremendously excited about the potential for creating form with code. We hope this book will inspire readers to think further about the relationships between these topics.

* Casey Reas and Ben Fry, Processing: A Programming Handbook for Visual Designers and Artists (Cambridge, MA: MIT Press, 2007).

The diverse typographic explorations on the title pages of the following chapters began with the same typeface used for this page—Neutral by Kai Bernau. Using Neutral's geometry as a foundation, we developed all of the typefaces by tweaking and writing code using Python, PostScript, and Processing. The title-page typography demonstrates the nature of codification: the embedding of information within a system.



WHAT IS CODE?

Codes typically serve three main purposes. They are used for communication, clarification, or obfuscation.

In Morse code, a word is transformed into short and long pulses so that it can be communicated over a telegraph. The word my is encoded by the sender into "-- -.--"; the resulting sound is then decoded back into my by the receiver.

Genetic information is encoded in sequences of deoxyribonucleic acid (DNA), such as "AAAGTCTGAC," with A standing for adenine, G for guanine, T for thymine, and C for cytosine. The genetic code is a set of rules that use these sequences to build proteins.

The California Health and Safety Code is a set of written laws that codify the rules set forth by the State Legislature. For example, section 12504 states, "Flammable liquid means any liquid whose flashpoint is 100 degrees Fahrenheit, or less."

Ever since the origins of writing, codes have been used to protect messages from unwanted eyes. For example, a code can be as simple as replacing each letter of the English alphabet with a number: A is 1, B is 2, C is 3, etc; with this code, the word secret becomes "19, 5, 3, 18, 5, 20."

The grid on this page began as a simple and evenly spaced pattern. The outline of Neutral was converted to points, and a system was defined in code to map these points on the grid. The point systems are slightly offset from their initial positions to reveal the information.

--. . -. . - .. -.-. / -.-. --- -.. . .-.-.- / .... . .- .-.. - .... / -.-. --- -.. . .-.-.- / -... ..-
.. .-.. -.. .. -. --. / -.-. --- -.. . .-.-.- / -... .- .-. / -.-. --- -.. . .-.-.- / -- --- .-. ... .
/ -.-. --- -.. . .-.-.- / -.. .-. . ... / -.-. --- -.. . .-.-.- / .- .-. . .- / -.-. --- -.. . .-.-
.- / ... . -.-. .-. . - / -.-. --- -.. . .-.-.- / -.-. --- -.. . ... / .- .-. . / ..- ... . -.. / ..-.
--- .-. / -.-. --- -- -- ..- -. .. -.-. .- - .. --- -. --..-- / -.-. .-.. .- .-. .. - -.-- --..-- /
.- -. -.. / --- -... ..-. ..- ... -.-. .- - .. --- -. .-.-.- / .. -. / -- --- .-. ... . / -.-. --- -..
. --..-- / .- / .-- --- .-. -.. / .. ... / - .-. .- -. ... ..-. --- .-. -- . -.. / .. -. - --- / ...
.... --- .-. - / .- -. -.. / .-.. --- -. --. / .--. ..- .-.. ... . / ... --- / - .... .- - / .. - /
-.-. .- -. / -... . / -.-. --- -- -- ..- -. .. -.-. .- - . -.. / --- ...- . .-. / .- / - . .-.. . --.
.-. .- .--. .... .-.-.- / ..-. --- .-. / . -..- .- -- .--. .-.. . --..-- / - .... . / .-- --- .-. -..
/ -- -.-- / .. ... / . -. -.-. --- -.. . -.. / -... -.-- / - .... . / ... . -. -.. . .-. / .. -. - --- /
-....- -....- / -....- .-.-.- -....- -....- --..-- / .-- .... .. -.-. .... / .. ... / - .... . -. / -..
. -.-. --- -.. . -.. / -... .- -.-. -.- / .. -. - --- / -- -.-- / -... -.-- / - .... . / .-. . -.-. .
.. ...- . .-. .-.-.- / ... .. -- .. .-.. .- .-. .-.. -.-- --..-- / --. . -. . - .. -.-. / .. -. ..-.
--- .-. -- .- - .. --- -. / .. ... / . -. -.-. --- -.. . -.. / .. -. / -.. -. .- / ... . --.- ..- . -.
-.-. . ... / -.--.- . .-.-.- --. .-.-.- / .- .- .- --. - -.-. - --. .- -.-. .-.-.- .-.-.- .-.-.- -.-
-.- .-.-.- / - .... . / --. . -. . - .. -.-. / -.-. --- -.. . / .. ... / .- / ... . - / --- ..-. / .-.
..- .-.. . ... / - .... .- - / ..- ... . / - .... / ... . --.- ..- . -. -.-. . ... / - --- / -...
..- .. .-.. -.. / .--. .-. --- - . .. -. ... .-.-.- / -.-. --- -.. . ... / .- .-. . / .- .-.. ... --- /
..- ... . -.. / ..-. --- .-. / -.-. .-.. .- .-. .. - -.-- .-.-.- / ..-. --- .-. / . -..- .- -- .--. .-..
. --..-- / - .... . / -.-. .- .-.. .. ..-. --- .-. -. .. .- / .... . .- .-.. - .... / .- -. -.. / ...
.- ..-. . - -.-- / -.-. --- -.. . / .. ... / .- / -.-. --- -- .--. .. .-.. .- - .. --- -. / --- ..-. /
.-- .-. .. - - . -. / .-.. .- .-- ... / - .... .- - / -.-. --- -.. .. ..-. -.-- / - .... . / .-. ..- .-..
. ... / ... . - / -... -.-- / - .... . / .-.. . --. .. ... .-.. .- - ..- .-. . .-.-.- / ..-. --- .-. / .
-..- .- -- .--. .-.. . --..-- / ... . -.-. - .. --- -. / .---- ..--- ..... ----- ....- / ... - .- - .
... --..-- / ..-. .-.. .- -- -- .- -... .-.. . / .-.. .. --.- ..- .. -.. / -- . .- -. ... / .- -. -.-- /
.-.. .. --.- ..- .. -.. / .-- .... --- ... . / ..-. .-.. .- ... .--. --- .. -. - / .. ... / .----
----- ----- / -.. . --. .-. . . ... / ..-. .- .... .-. . -. .... - --..-- / -.-. --- -.. . ... /
.- .-.. ... --- / --- -... ... -.-. ..- .-. . / .. -. ..-. --- .-. -- .- - .. --- -. / ... .. -. -.-. .
/ - .... . / --- .-. .. --. .. -. ... / --- ..-. / .-- .-. .. - .. -. --. --..-- / - .... . -.-- / ....
.- ...- . / -... . . -. / ..- ... . -.. / .--. .-. --- - . -.-. - / -- . ...- --. . ... / ..-. .-.
--- -- / ..- -. .-- .- -. - . -.. / . -.-- . ... .-.-.- / .- / -.-. --- -.. . / -.-. .- -. / -... . /
.- ... / ... .. -- .--. .-.. . / .- ... / .-. . .--. .-.. .- -.-. .. -. --. / . .- -.-. .... / .-.. . - -
. .-. / .-- .. - .... / .- / -. ..- -- -... . .-. ---... / .- / .. ... / .---- --..-- / -... / .. ... /
..--- --..-- / -.-. / .. ... / ...-- --..-- / . - -.-. .-.-.- / .-- .. - .... / - .... / -.-. ---
-.. . --..-- / - .... . / .-- --- .-. -.. / ... . -.-. .-. . - / -... . -.-. --- -- . ... / .---- ----.
/ ..... / ...-- / .---- ---.. / ..... / ..--- ----- .-.-.-

Morse code, 1840s
In Morse code, every character is encoded as a rhythmic sequence of dots and dashes.

THE ALGORITHM

There are many types of code. Within the context of this book, we're interested primarily in codes that represent a series of instructions. This type of code—often called an algorithm, procedure, or program—defines a specific process with enough detail to allow the instructions to be followed. While the word algorithm may be unfamiliar to you, its meaning is not. It's just a precise way of explaining how to do something. It is commonly used within the context of computer instructions. While most people wouldn't refer to a pattern for knitting a scarf as an algorithm, it's the same idea.

```
Row 1: (RS) *K2, P2* across
Rows 2, 3, & 4: Repeat Row 1
Row 5: (RS) *K2, P2, C8F* Repeat to last
   4 sts, K2, P2
Row 6: Repeat Row 1
Repeat rows 1-6 for desired length,
   ending with row 4
Bind off in K2, P2 pattern
```

Likewise, directions to get from one place to another, instructions for assembling kit-of-parts furniture, and many other types of guidelines are also algorithms.

```
Hiking Directions to Point Break

From the North:
- Follow the trail from the Nature
  Center
- Turn right at the Water Tower,
  walk until you see the Old Oak Tree
- Follow directions from the Old Oak
  Tree

From the South:
- From the Pinic Grove, follow the
  Botany Trail
- Turn right on the South Meadow Trail
- Turn right on the Meadow Ranch Trail,
  walk until you see the Old Oak Tree
- Follow directions from the Old Oak
  Tree

From the Old Oak Tree:
- Follow the path under the tree
- Turn right onto the Long Hill Trail
- Follow the trail until you reach
  Point Break
```

Algorithms can be defined as having four qualities. These qualities can be easily understood when defined in relation to travel directions.

There are many ways to write an algorithm. In other words, there are always multiple ways to get from point A to point B. Different people will create different sets of directions, but they all get the reader to their intended destination.

An algorithm requires assumptions. Hiking directions assume that you know how to hike, from knowing to wear the right shoes, to understanding how to follow a winding trail, to assuming that you know to bring plenty of water. Without this knowledge, the hiker may end up lost and dehydrated with blistered feet.

An algorithm includes decisions. Directions often include instructions from different starting locations. The person reading the directions will need to choose a starting position.

A complex algorithm should be broken down into modular pieces. Directions are often divided into small units to make them easy to follow. There may be separate directions for coming from the North or South, but at a certain point the directions converge and both groups follow the same instructions.

PostScript

```
gsave
        % move to the center of the page
        306 396 translate
        % repeat from 0 to 360 in increments of 12
        0 12 360 {
                pop
                % draw line from (20,0) to (200,0)
                20 0 moveto 200 0 lineto
                % rotate 12 degrees
                12 rotate
        } for
        stroke
grestore
```

Processing

```
void setup() {
  size(800, 600);
  stroke(255, 204);
  smooth();
  background(0);
}

void draw() {
  float thickness = dist(mouseX, mouseY, pmouseX, pmouseY);
  strokeWeight(thickness);
  line(mouseX, mouseY, pmouseX, pmouseY);
}
```

PostScript, 1982
The PostScript language specializes in defining pages for print output. Although PostScript files can be written in a text editor, they are typically created with a graphical user interface (GUI), which makes it easier to create and edit files but removes the more powerful features.

Processing
Processing is a programming language and environment with a focus on coding form, motion, and interaction. It simplifies and extends the Java language. This Processing program draws a line at the position of the mouse; the line thickness is calculated by the speed of the mouse.

CODE AND COMPUTERS

In computer programming, code (also called source code) is used to control the operations of a computer. It is an algorithm written in a programming language. There are thousands of programming languages, and new ones are developed every year.

Although the words and punctuation used in programming languages look different from written English words, the codes they use are intended to be read and understood by people. Specifically, computer-programming languages are designed for the way people are taught to read and write from a young age, with the precision necessary for instructing a computer. Human languages are verbose, ambiguous, and contain large vocabularies. Code is terse, has strict syntactical rules, and small vocabularies. Constructing an essay and a computer program are, however, both forms of writing, as explained in Processing: A Programming Handbook for Visual Designers and Artists:

> Writing in a human language allows the author to utilize the ambiguity of words and to have great flexibility in constructing phrases. These techniques allow multiple interpretations of a single text and give each author a unique voice. Each computer program also reveals the style of its author, but there is far less room for ambiguity."[1]

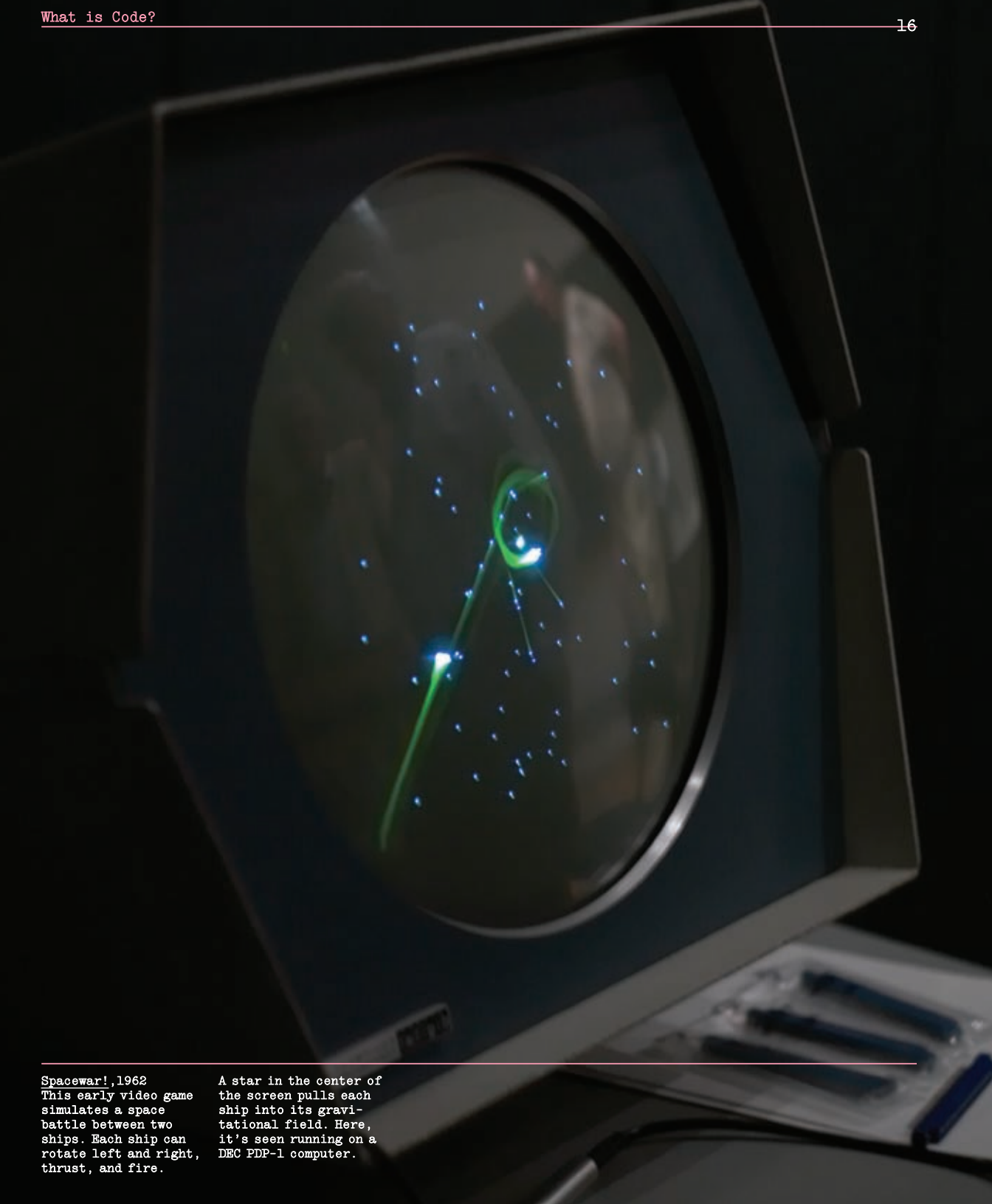
In fact, there can only be one interpretation of every piece of code. Unlike people, computers are not able to guess or interpret a meaning if it's not stated exactly. There are rules of grammar in every language, but if I misspell a wurd or two, you will still understand, but the computer won't. Fortunately (or maybe unfortunately), people are highly adaptable, and for many individuals it's easy to learn how to structure code.

Before a piece of code can be run on a computer, it must be converted from a human-readable format to a computer-executable format; these are sometimes called machine code, binaries, or executables. This conversion transforms the code into software. It can now run (or execute) on a computer. Machine-based code is usually represented as a series of 1s and 0s:

```
0001 0001 0000 1001 0000 0001 0000
1110 0000 1001 1100 1101 0000 0101
0000 0000 1100 1001 0100 1000 0110
0101 0110 1100 0110 1100 0110 1111
0010 0001 0010 0100
```

While this series of 1s and 0s looks different from source code, it's a literal translation of the human-readable code. This translation is required for the computer to be able to follow the instructions. We think you'll agree that understanding what these sequences of 1s and 0s mean is more difficult than reading the source code. This format instructs the computer's operations at the lowest level. Each bit (1 or 0) is grouped into bytes (a sequence of eight bits) that define how the computer makes calculations and moves data into and out of the processor.

[1] Casey Reas and Ben Fry, Processing: A Programming Handbook for Visual Designers and Artists (Cambridge, MA: MIT Press, 2007), 17.



Spacewar!, 1962
This early video game simulates a space battle between two ships. Each ship can rotate left and right, thrust, and fire.

A star in the center of the screen pulls each ship into its gravitational field. Here, it's seen running on a DEC PDP-1 computer.

THINKING IN CODE

Software is a tool for the mind. While the industrial revolution produced tools to augment the body, such as the steam engine and the automobile, the information revolution is producing tools to extend the intellect. The software resources and techniques at our disposal allow us to access and process enormous quantities of information. For example, the science of genomics (the study of the genome) and the collaborative scholarship of Wikipedia were not possible without the aid of software. But using software is not only about increasing our ability to work with large volumes of information; it also encourages new and different ways of thinking.

The term procedural literacy has been used to define this potential. Michael Mateas, an associate professor in the computer science program at the University of California, Santa Cruz, describes procedural literacy as "the ability to read and write processes, to engage procedural representation and aesthetics."[2] One component of procedural literacy is the idea that programming is not strictly a technical task; it's an act of communication and a symbolic way of representing the world. A procedural representation is not static. It's a system of rules that define a space of possible forms or actions. Video game designer Ian Bogost defines this elegantly in his book Persuasive Games: The Expressive Power of Videogames:

> To write procedurally, one authors code that enforces rules to generate some kind of representation, rather than authoring the representation itself. Procedural systems generate behaviors based on rule-based models; they are machines capable of producing many outcomes, each conforming to the same overall guidelines.[3]

A video game like Spacewar! is a good example of a procedural representation. Playing the game requires understanding the spatial and kinetic relationships between two opposing spaceships. Each player controls a ship by rotating left and right and by thrusting the rocket with the goal of shooting down the other ship. To write the game, a procedurally literate individual had to break the behaviors into modules with enough detail so that they could be programmed. The primary complexity involved in creating the game is not technical; it's about choreographing all of the components into a coherent and enjoyable experience. Procedural literacy is a general way of thinking that cuts across all programming languages and even applies to thinking outside the domain of writing source code.

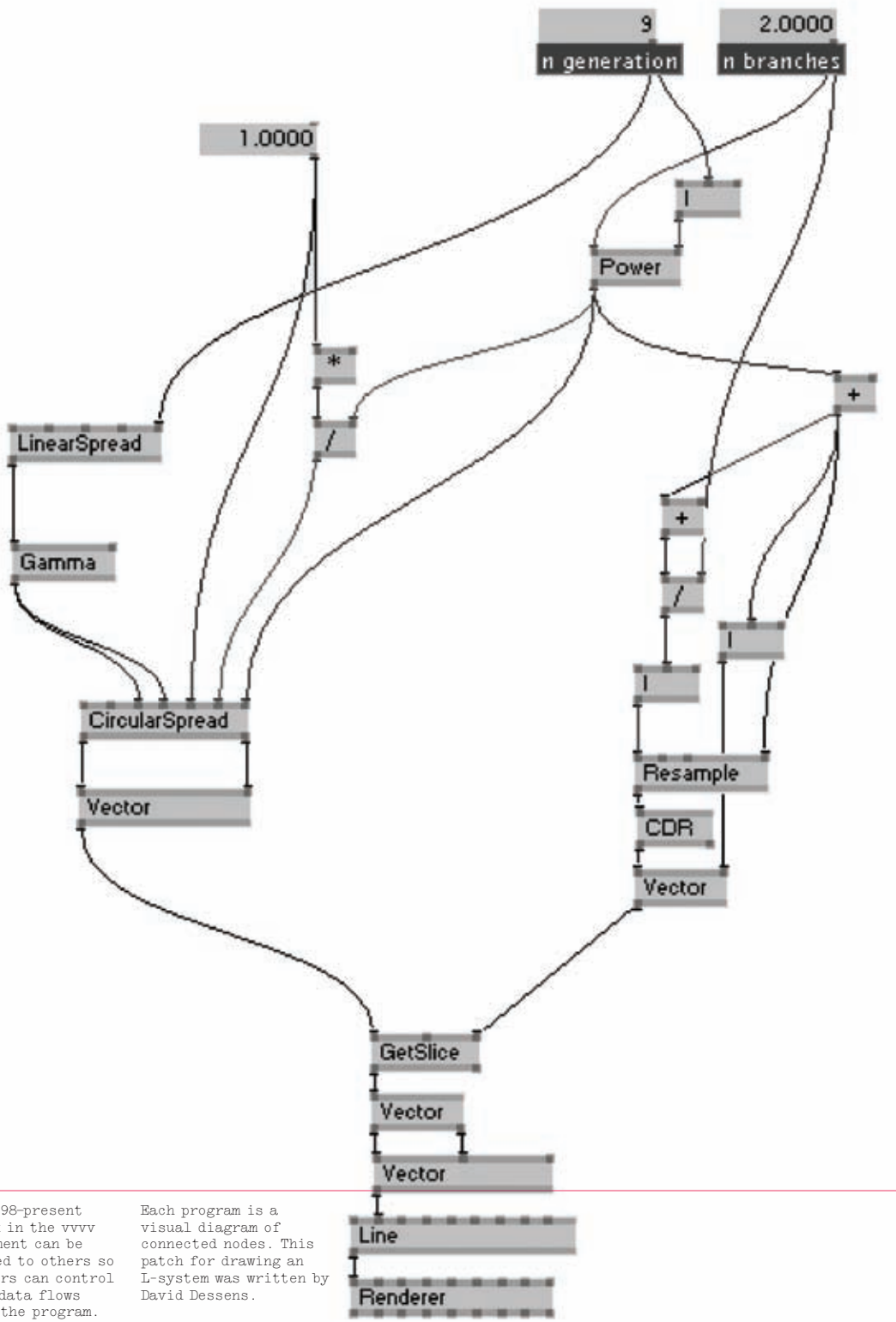
Each programming language is a different kind of material to work and think with. Just as a carpenter knows the unique properties of various woods, including oak, balsa, and pine, a person knowledgeable about software knows the unique aspects of different programming languages. A carpenter building a table will select the wood based on factors such as cost, durability, and aesthetics. A programmer selects a programming language based on the estimated budget, operating system, and aesthetics.[4] The syntax (or grammar) of each programming language structures what is possible within that language. Different programming languages encourage programmers to think about their work through the affordances (or action possibilities) and constraints of that language.

The way that a programming language encourages a certain mode of thinking can be demonstrated by comparing two very different languages: BASIC and LOGO. In each of these programming environments, drawing a triangle requires a different approach and understanding of space. BASIC relies on an established coordinate system and requires the knowledge of coordinates; lines are drawn by connecting one coordinate

[2] Michael Mateas, "Procedural Literacy: Educating the New Media Practitioner," Beyond Fun, ed. Drew Davidson (Pittsburgh, PA: ETC Press, 2008), 67.

[3] Ian Bogost, Persuasive Games: The Expressive Power of Videogames (Cambridge, MA: MIT Press, 2007), 4.

[4] There are many reasons why one programming language may be preferred over another. For example, some languages allow code to be written faster, but this is usually at the expense of how fast the code is capable of running. Some languages are more obscure than others, therefore reducing the chances that another programmer knows of the language.



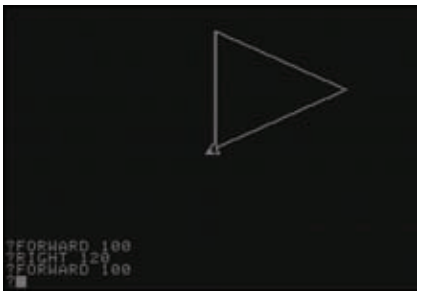
vvvv, 1998–present
Each box in the vvvv environment can be connected to others so that users can control the way data flows through the program.

Each program is a visual diagram of connected nodes. This patch for drawing an L-system was written by David Dessens.

to another.[5] In contrast, LOGO is a language developed for young children who have not yet studied geometry; it allows the user to code a shape with only an understanding of angles and the difference between left and right. In this program, the child draws lines by directing the path of a turtle on screen. The child imagines that he or she is the turtle and moves forward, turns to the right, moves forward, turns to the right again, and then moves forward to complete the triangle.[6] Although both languages allow the same shapes to be drawn, BASIC promotes objectivity, while LOGO fosters exploration. Additionally, LOGO encourages the programmer to run the code mentally, which is a useful skill for developing procedural literacy.

BASIC

```
10 HGR : HCOLOR = 3
20 HPLOT 0, 0 TO 100, 50 TO 0, 100 TO 0, 0
30 END
```



LOGO

```
FORWARD 100
RIGHT 120
FORWARD 100
RIGHT 120
FORWARD 100
```

Visual programming languages (also called graphical programming languages) provide an alternative way of thinking with code. Writing a program with a visual programming language is similar to making a diagram instead of writing a text. Three of the most popular visual programming languages within the arts—Max, Pure Data, and vvvv—were influenced by the way sounds are constructed using patch cables attached to analog synthesizers. Virtual cables, represented as lines on screen, are used to connect programming modules together to define the software. Visual programming languages make it easy to generate and filter images and sounds, but they are often too cumbersome for writing long, complicated programs. For example, the Max program is written in the text programming language C++, not a visual programming language.

[5] The BASIC language was designed in 1964 to teach non-technical university students how to program. Variations of it were used to teach many children and hobbyists how to program in the early era of personal computing.

[6] In LOGO, first developed in 1967, drawings are made by moving a triangular turtle around on screen. Early versions of the language moved a robotic turtle around the room.

PROPOSAL FOR WALL DRAWING, INFORMATION SHOW

Within four adjacent squares,
each 4' by 4',
four draftsmen will be employed
at $4.00/hour
for four hours a day
and for four days to draw straight lines
4 inches long
using four different colored pencils;
9H black, red, yellow and blue.
Each draftsmen will use the same color throughout
the four day period,
working on a different square each day.

Proposal for a Wall Drawing, Information Show, by Sol LeWitt, 1970
The instructions for LeWitt's drawings are executed by a skilled draftsperson who interprets the instructions during the drawing process.

CODE AND THE ARTS

Beginning in the 1940s, code was developed to assist with work in the fields of science and engineering. Seymour Papert, a pioneer in researching computers and creativity, explains the situation at the time:

> The world was at war. Complex calculations had to be done under time pressures not normally felt by mathematicians: numerical calculations related to the design and use of weapons; logical manipulations to break ever more complex codes before information became old news....It's unlikely that they gave even a passing thought to making computers user-friendly to people with softer styles than theirs.[7]

The decisions made about computers and programming languages since that time have, along with other factors, hindered the synthesis of software and the arts. It's an unfortunate fact that many languages used within the arts were not originally designed for those areas. The software desires of designers, architects, and artists are often different from those of scientists, mathematicians, and engineers. The technical skill required to create visual form with the most dominant languages, such as C++ and Java, often takes years to acquire.

An alternative way of considering code is revealed through the work of artists who in the 1950s and 1960s began to experiment with software and themes related to software, such as dematerialization and system aesthetics. These explorations were first presented to the general public in the exhibition Cybernetic Serendipity, held at the Institute of Contemporary Arts in London in 1968; as well as in the shows Software—Information Technology: Its New Meaning for Art, held at the Jewish Museum in New York in 1970; and Information, held at the Museum of Modern Art (MoMA) in 1970. The curator of the Software exhibit, Jack Burnham, described the works on view as "art that is transactional in that they deal with underlying structures of communication and energy exchange."[8] Among the works on view, Hans Haacke's ambitious Visitor's Profile sought to reveal the elite social status of the museum's patrons as a form of critique of the art world. Using a computer interface, it tabulated personal information solicited from visitors. Les Levine exhibited Systems Burn-off X Residual Software, a collection of photographs discussed within the context of software. Levine claimed that images are hardware, and that information about the images is software. He wrote the provocative statement, "All activities which have no connection with object or material mass are the result of software."[9] Similar to Levine's piece, many of the works featured in the Information exhibition at MoMA were characterized as "conceptual art."

At the same time, artists and musicians—including Mel Bochner, John Cage, Allan Kaprow, Sol LeWitt, Yoko Ono, and La Monte Young—created a different type of conceptual and process-based art by writing instructions and creating diagrams as a form of art. For example, instead of physically making the drawings, LeWitt encoded his ideas as instructions that were used to produce drawings. He wrote a series of rules to define the task of a draftsperson, but the rules are open for interpretation; therefore many different results are possible. Ono's artworks, such as Cloud Piece, are instructions for life; each short text asks the reader to perform actions like laughing, drawing, sitting, or flying. Like programmers, these creators all wrote instructions for actions. Through their use of English as the programming language, they introduced ambiguity, interpretation, and even contradiction.

Simultaneous with these conceptually focused explorations, engineers were creating programming systems for the creation of visual images. In 1963 at Bell Laboratories, Kenneth C. Knowlton wrote BEFLIX, a specialized program for constructing animation, which he used

[7] Seymour Papert, The Children's Machine: Rethinking School in the Age of the Computer (New York: Basic Books, 1994), 157.

[8] Jack Burnham, Software—Information Technology: Its New Meaning for Art (New York: The Jewish Museum, 1970), 10.

[9] The New Media Reader (Cambridge, MA: MIT Press, 2003), 255.

CLOUD PIECE
Imagine the clouds dripping. Dig a hole in your garden to put them in.

1963 Spring

Electronic Numerical Integrator And Computer (ENIAC), 1943–46
The first digital computers were very different from modern computers. ENIAC cost almost $500,000 and weighed over thirty tons. This U. S. Army photo shows two of the computer's programmers.

Cloud Piece, by Yoko Ono, 1963
Ono's artwork is the instructions. The reader imagines or performs the actions.

BASIC

```
10 DEFINT A-Z' DRAW100SQ
20 CLS
30 MOVE$="!AX"
40 DRW$="!AY"
50 OPEN "COM1:1200,0,7,1" AS #1
60 PRINT #1, "!AE";
70 FOR ROW=0 TO 90 STEP 10
80  FOR CLM=0 TO 90 STEP 10
90       GOSUB 310
100      PRINT #1, MOVE$+STR$(ROW)+STR$(CLM)
110      GOSUB 310
120      PRINT #1, DRW$+STR$(ROW+10)+STR$(CLM)
130      GOSUB 310
140      PRINT #1, DRW$+STR$(ROW+10)+STR$(CLM+10)
150      GOSUB 310
160      PRINT #1, DRW$+STR$(ROW)+STR$(CLM+10)
170      GOSUB 310
180      PRINT #1, DRW$+STR$(ROW)+STR$(CLM)+";"
190 NEXT CLM
200 NEXT ROW
210 PRINT
220 GOTO 70
230 '
240 '
250 '
260 '
270 'XON/XOFF subroutine
```

Hypertalk

```
on mouseUp
  put "100,100" into pos
  repeat with x = 1 to the number of card buttons
    set the location of card button x to pos
    add 15 to item 1 of pos
  end repeat
end mouseUp
```

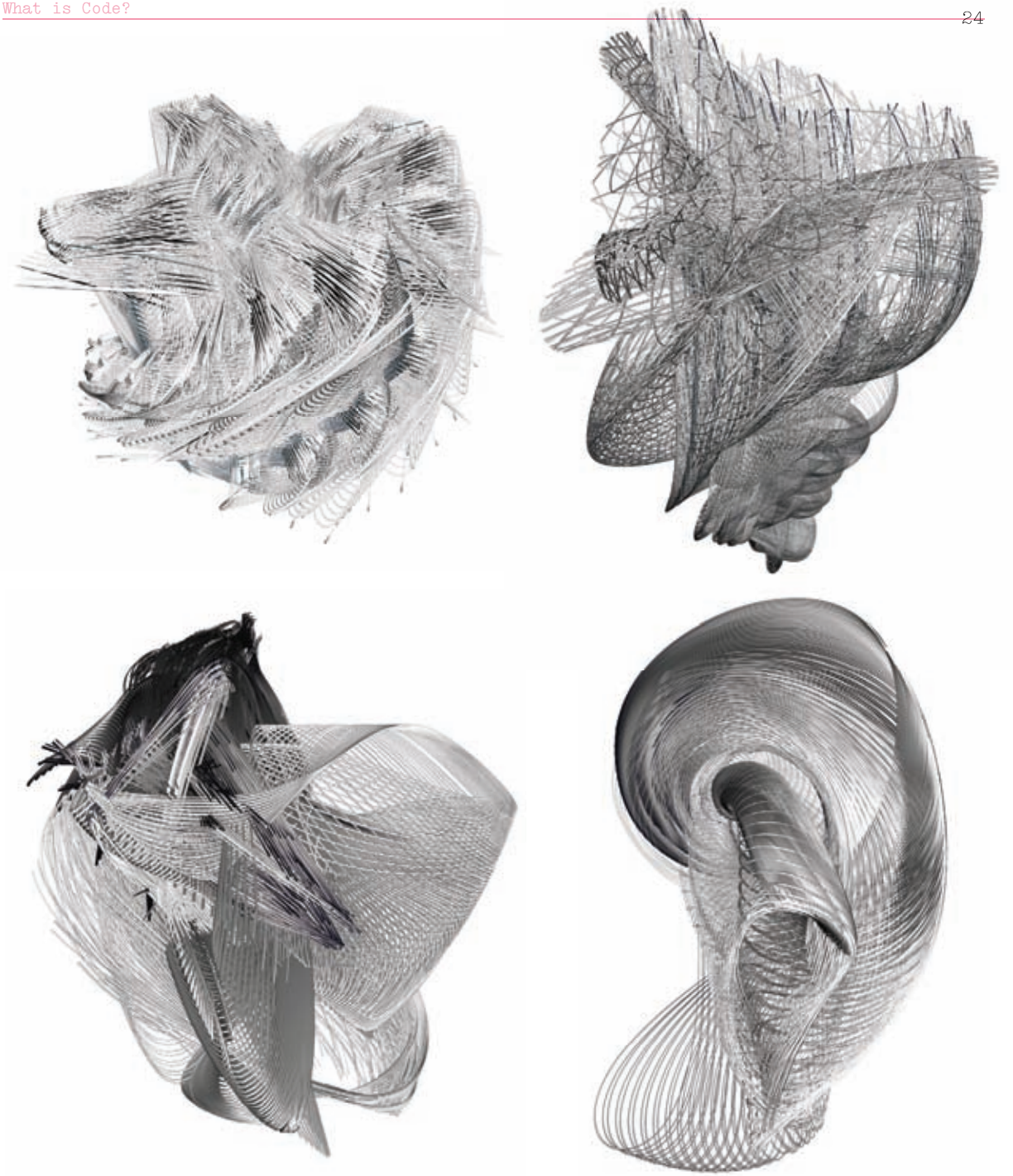
BASIC, 1964
This program draws a grid of 100 squares. It demonstrates how the BASIC language can be used to control a mechanical drawing arm known as a plotter.

HyperTalk, 1987
The HyperTalk language was written to be easy for beginners. It is more similar to English than most other programming languages.

to create early computer films in collaboration with artists Stan VanDerBeek and Lillian F. Schwartz. The computer-generated film, Permutations, was created in 1966 by John Whitney Sr. using GRAF, a programming library developed by Dr. Jack Citron of IBM. Both BEFLIX and GRAF were built on top of the language Fortran. From these and other early explorations, the development of programming languages written expressly for the arts has continued to gain momentum, building toward the current frenzy of activity.

In the 1980s, the proliferation of the personal computer allowed programming to reach a wider audience, which in turn led to the development of HyperTalk, a programming language for Apple's unique HyperCard application (an early hypermedia system). The related Lingo language was developed for the first release of Adobe Director in 1988 (formerly Macromedia Director, and before that MacroMind Director). Lingo was the first programming language used by many designers and artists in the era leading up to the development of the World Wide Web in the early 1990s. The early days of the web fostered intense graphic programming exploration, primarily channeled through the ActionScript language. The rise of programming literacy within the arts and architecture communities has led to the current proliferation of programming options; many are featured within this book.

The influence of code is not limited to the screen and projected image. It is also felt in physical space. Code is used to control elements of products, architecture, and installations. It is used to create files that are output as prints and made physical through computer-controlled machines that cut and assemble materials, including wood, metal, and plastic. Code is rapidly moving outside the boundaries of the screen and is starting to control more aspects of the physical world. There are examples of this in the Producing Form section of "Form and Computers" (p. 37), as well as throughout this book.



Strand Tower, by Testa & Weiser, Architects, 2006
Strand Tower precursors were generated using a purpose-built software (Weaver) and coded through an iterative templating process. Specific fiber behaviors or traits such as fraying, bundling, knitting, and bias patterning are coded into the design. Fiber agency and affiliation is more informal and multidimensional than conventional woven patterns.

WHY CODE?

The use of software in the arts can be separated into two categories: production and conception. In the first category, the computer is used to produce a preconceived form; in the second, the computer participates in the development of the form. Within the context of this book, we're primarily interested in the latter. (It is important to note that this distinction does not imply a value judgment but does impact the types of forms that are created.)

Using the computer to reduce the amount of time needed to create a complex, repetitive composition was often the motivation for the early adoption of software and its integration into the creative process. This was especially important in the field of animation, where subtle changes had to be repeated thousands of times to create the illusion of motion; however, this alluring technical benefit has had a profound effect. If initial production takes one-tenth the time that it would take to execute the work by hand, then the artist can create ten versions in the same amount of time. This way, many versions can be created and the best chosen. Efficiency facilitates the creative process by enabling more time for exploration as less time is needed for the final production. Eventually the computer came to be understood as more than just a production tool. People started to see it as, in the words of computer graphics pioneer A. Michael Noll, "an intellectual and active creative partner that, when fully exploited, could be used to produce wholly new art forms and possibly new aesthetic experiences."[10]

Often, to realize a new or unique vision requires that artists and designers exceed the limitations of existing tools. Proprietary software products are general tools designed for the production of specific types of forms. If you are already using software for your work, why constrain yourself to the expectations of a software company or another programmer? To go beyond these limitations, it is necessary to customize existing applications through programming or to write your own software.

Each existing form of media—whether drawing, printing, or television—is capable of assuming new qualities of expression. For example, video games demonstrate many of the distinct characteristics of software. If you've ever succumbed to the pleasures of a great game (we know some of you have; if you haven't, then what are you waiting for?), you already know that the emotions experienced while playing are different than those felt while watching a film or looking at a drawing. Games can be physically engrossing and socially engaging, and they can sustain intense fascination over a period of months.

In reference to the emerging media of his time, theorist Marshall McLuhan wrote, "Today we're beginning to realize that the new media aren't just mechanical gimmicks for creating worlds of illusion, but new languages with new and unique powers of expression."[11] Writing code is one gateway for realizing these new forms. Learning to program and to engage the computer more directly with code opens the possibility of not only creating tools, but also systems, environments, and entirely new modes of expression. It is here that the computer ceases to be a tool and instead becomes a medium. We hope the following chapters will provide evidence for you to draw your own conclusions about the potential of software in the visual arts.

[10] Jasia Reichardt, Cybernetics, Art, and Ideas (New York: New York Graphic Society, 1971), 143.

[11] Edmund Snow Carpenter and Marshall McLuhan, Explorations in Communication: An Anthology (Boston, MA: Beacon Press, 1960), 2.