Avant.org / Project / Hello, World!

Taeyoon Choi

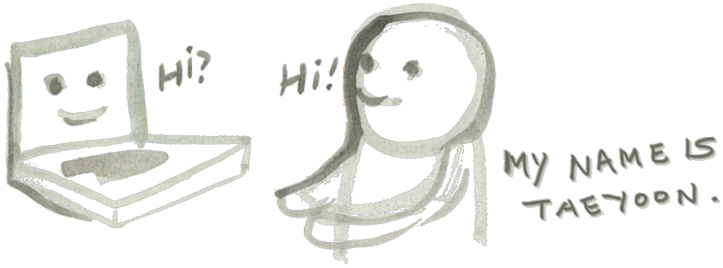
5 June 2017

Handmade Computer

computation

urbanization

poetics



and

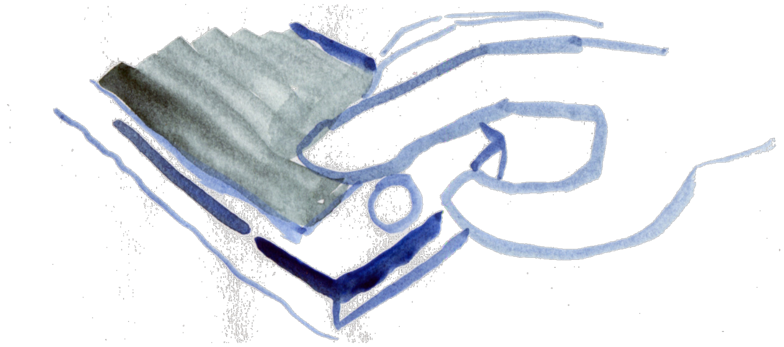
THIS IS A STORY ABOUT MAKING A COMPUTER.

I started in 2011 with the lofty goal of building a computer from scratch. Along the way I learned a lot about how computers work and a little about how humans do too.

Over the next 7 chapters I'd like to share my experiences, learning about the fundamentals of electricity and computation by playing with circuits, making mistakes, and finding a story that's kept me going.

Why learn about computers? *Because...*

IF YOU ARE READING THIS, MOST LIKELY YOU ARE USING A COMPUTER OR A SMART PHONE



WHICH IS A SMALL AND POWERFUL COMPUTER.

...but how many of us *really* know how a computer works?

I didn't, so I decided to try building one of my own. A *Handmade Computer* that would let me see how all the pieces fit together.

THERE ARE MANY STORIES
ABOUT HOW THE COMPUTER
WAS INVENTED. THE COMMON
SiLicon Valley FOLKLORE,
LONE HACKERS IN THEIR
GARAGE... THE MILITARY
HISTORY OF INTELLIGENCE,
CRYPTOGRAPHY...
CYBERPUNK ViSiONS OF
MECHANICAL-CALCULATING
MACHINE-BRAINS. THE
LIST GOES ON. HOWEVER
THIS STORY iS NOT SO MUCH
ABOUT THE PAST. IT'S NOT
ABOUT THE FUTURE EITHER.
IT'S ABOUT THE PRESENT.

There is a Silicon Valley mythology centered around protagonists like Gordon Moore, Bill Gates, and Steve Jobs, a legend of innovation which traces its history to an idealized California counterculture.

But left out of this glossy narrative are the parallel stories of militarization, state power, and mass production, which not only financed, but provided the social conditions for the computing industry as we know it.

The first computers were human.

In fact, we get the term “computer” from the forgotten practice of human computation. Well before computers became “personal,” an entire profession of arithmetic labor (mostly women) were contracted for scientific and military production.

So, understanding the computer requires study of both its essential technologies and the story behind the utopian myth. A history of grand calculating machines, built to accelerate the engines of war, and the so-called “Californian Ideology,”[1] [2] that grew out of the military-industrial complex to produce the Silicon Valley we know today.

We’ll learn all about this as we build a computer of our own, together.

THE COMPUTERS WE USE FOR WORK AND PLEASURE.

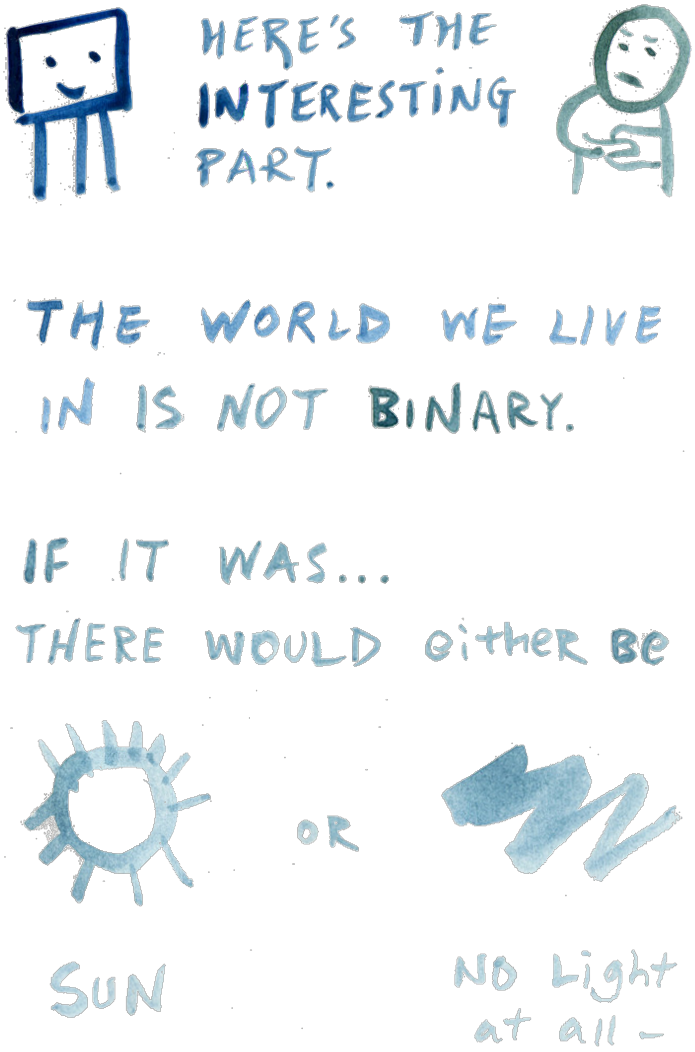
SMALL AND MEGA LARGE...

THEY ALL SHARE A CORE PRINCIPLE: THEY USE

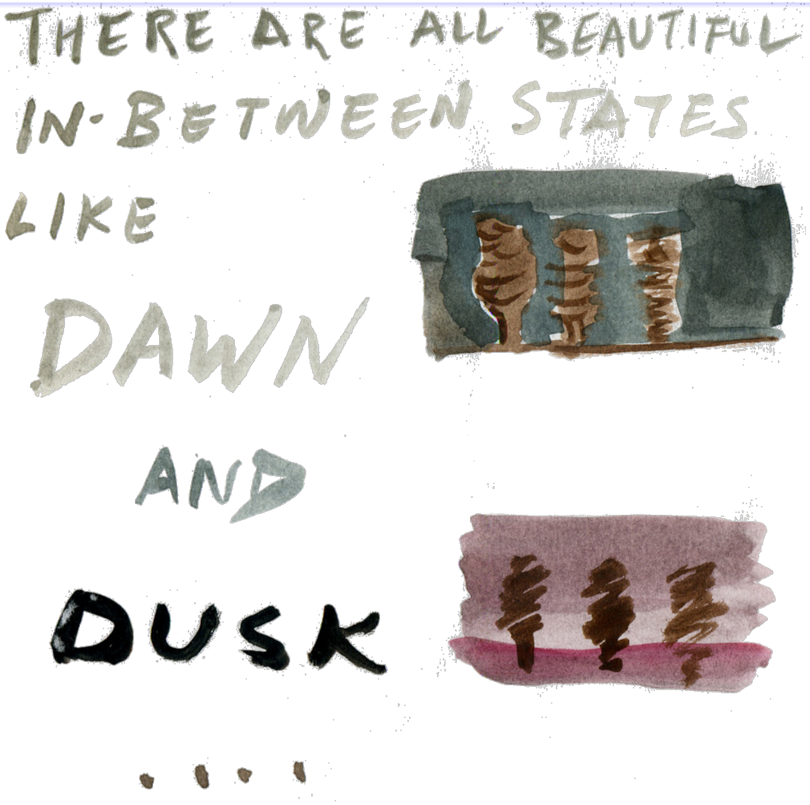
0 1

ZERO AND ONE

TO REPRESENT ALL OTHER NUMBERS. AND ALSO ALL OTHER MATHEMATICAL OPERATIONS, LIKE ADDING, MULTIPLYING, ETC.


HERE'S THE INTERESTING PART.
THE WORLD WE LIVE IN IS NOT BINARY.
IF IT WAS...
THERE WOULD either BE
SUN
OR
NO Light at all -

However...



Life can’t be reduced to binary.

SO HOW DO COMPUTERS ALLOW US TO COMMUNICATE INTRICATE MESSAGES AND DELICATE MEMORIES?

Interface

Taeyoon Choi, *Tablet computer*, Acrylics on canvas, 8 x

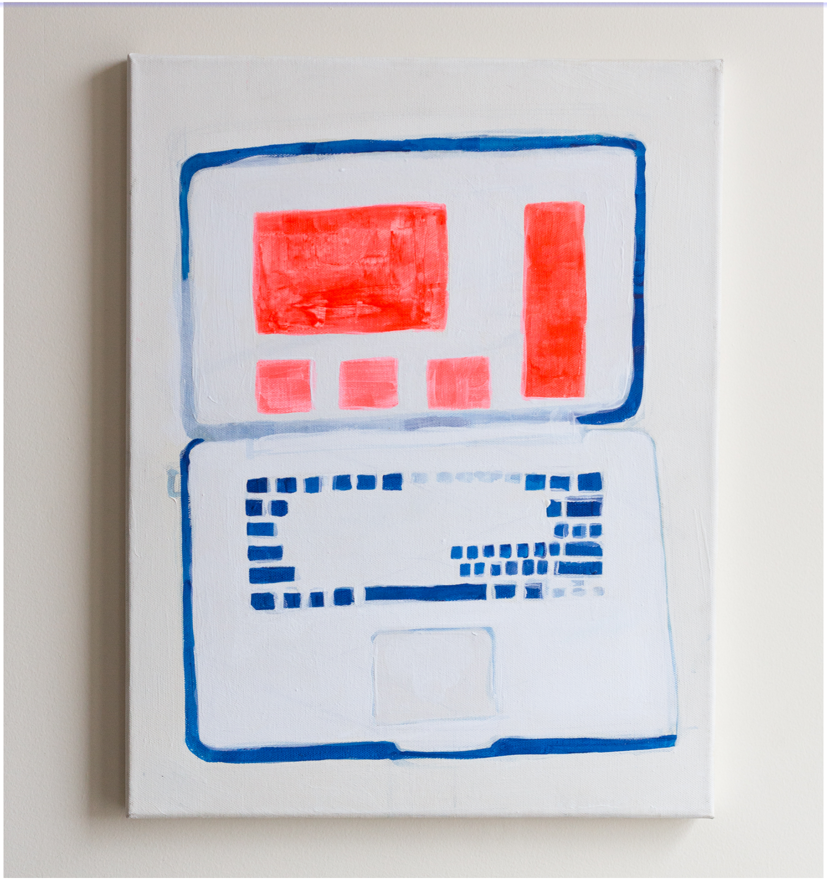
12 inches, 2013

Every day we look at computer screens for a very long time. We touch them, carry them, sleep next to them. Our lives are stored in bits, messages turned to network packets. Living spaces arranged in human-sized pixels.

But behind the familiar interface, the inner workings still seem so distant. For most of us the window into our digital lives is opaque.

How did we get here? Well… as the computer was commoditized, users became the primary source of value. Commercial incentives drove corporations to establish proprietary ecosystems[3] to capture market share, setting off fierce competition for “frictionless” experiences that deliberately concealed the logic inside. And with computers designed to be ever-more complex, while interfaces made ever-simpler, the gap between user and machine only grew wider.

But now we’re getting ahead of ourselves. Before we get into history let’s begin by describing the computer in simple, technical terms, as an object that performs only a few basic operations, repeated and abstracted.



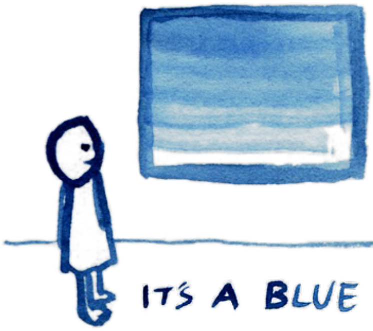
Taeyoon Choi, *Laptop*, Gouache on canvas, 16 x 20 inches, 2013

Most of the time, when we use a computer, we are looking at a *graphical user interface*, GUI for short, that's displayed on our monitor. And when we enter text with a keyboard or move our mouse, the interface software passes that data to a shared system which makes the information available to many applications.

The GUI software, like all computer programs, is made of code. Code is the language of the computer. Sets of instructions for how the hardware should manipulate stored information.

Once the instructions have been completed, the results then flow back to the GUI and can be seen as changes on the screen. Computed at high speed, this interactive loop quickly moves information through many levels of abstraction, the result appearing almost instantaneous to you and me.

THE BEAUTY OF ABSTRACTION IN COMPUTERS IS SIMILAR TO MINIMALIST ART WORKS.



OH, TEXTURES AND SHADOWS.

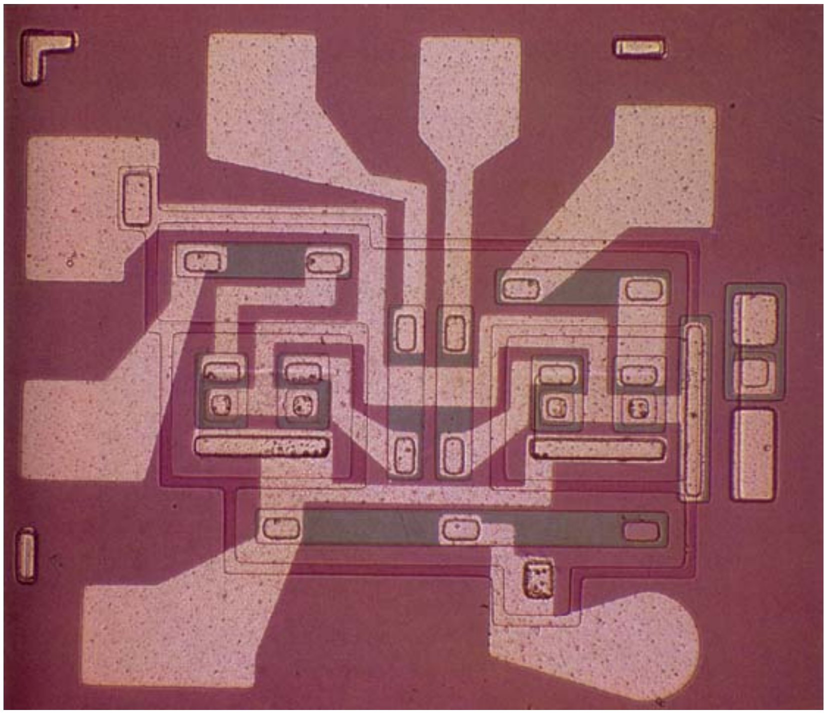
THE
ART OBJECT
REVEALS
ITSELF

IF YOU
GIVE IT

Just like a painting, stepping inside the computer takes patience, but it may be easier than you think.

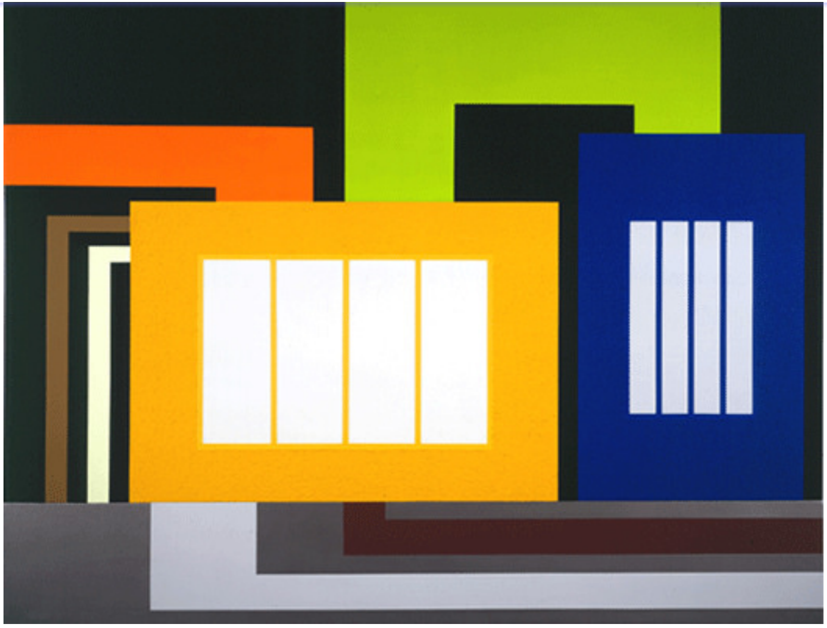
ENOUGH
TIME

Open a computer and you find lots of small electronic components connected to a circuit board which ties them all together. This is an image of a microchip made in 1963 shown at 100x magnification.



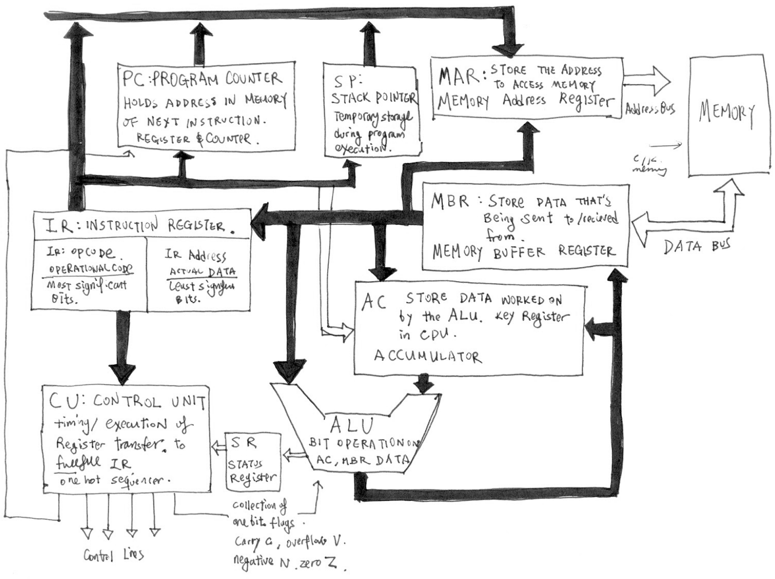
Microchip with resistor-transistor flip-flop logic.[4]

Micrograph photos reveal the unique visual patterns of the computer's interior circuitry. Designed to maximize performance at infinitesimal scale, they can also be quite geometrically appealing. When I look at images like this one I see dynamic works of art, grid plans for a miniature city.

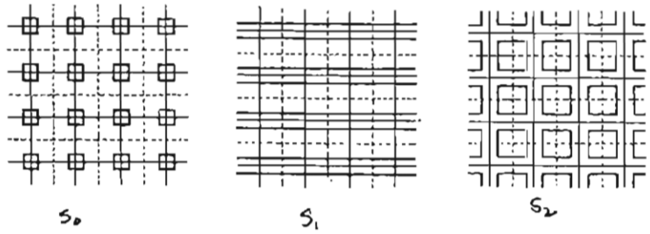


Peter Halley, *Instant City*(1996)[5]

Peter Halley's painting *Instant City* reminds of a computer architecture diagram, information flowing from one box to the next. The painting depicts various levels of abstraction differing in size and color, but with the same essential pattern throughout.


PC : PROGRAM COUNTER
HOLDS ADDRESS IN MEMORY OF NEXT INSTRUCTION.
REGISTER & COUNTER.
SP:
STACK POINTER
Temporary storage during program execution.
MAR: STORE THE ADDRESS TO ACCESS MEMORY
MEMORY Address Register
Address Bus
MEMORY
c/jc memory
MBR : STORE DATA THAT'S Being sent to / recieved from.
MEMORY BUFFER REGISTER
DATA BUS
IR : INSTRUCTION REGISTER.
IR: opcode.
OPERATIONAL CODE
Most significant bits.
IR Address
ACTUAL DATA
Least signifcant bits.
AC STORE DATA WORKED ON by the ALU. key Register in CPU.
ACCUMULATOR
CU: CONTROL UNIT
timing/ execution of Register transfer. to fullfill IR
one hot sequencer.
SR
STATUS Register
ALU
BIT OPERATION ON AC, MBR DATA
collection of one bit flags.
Carry C, overflow V.
negative N. zero Z.
Control Lines

It's surprising how often you'll find connections between art, architecture, and computation. The term "design pattern" for instance, used to describe a computer's repeated software elements, was first introduced to computer engineers by the architectural theorist Christopher Alexander.[6]



Three building types.

Image from Christopher Alexander's *A Pattern Language*[7]

Computation, abstract painting, and Modernist architecture each have their own seperate histories. But all three were affected by the same ethos of industrialization and their visual similarity reflects this shared influence.

- **CPU diagrams visualize the operations and the flow of information within the computer's hardware.**
- **Paintings like *Instant City* render geometric forms of control onto the two dimensional surface.**
- **City plans describe the layout and organization of urban life.**

They are abstractions designed for controlled repetition: repeated signalling, repeated viewing, repeated interaction.

Repetition is a powerful tool that lets us perform complex tasks by breaking them down into a series of a simple operations. It enables a basic pattern to scale to larger, more complex structures. This process of scaling through repetition can be seen from the microscopic details of silicon chips, to the grand scale of urban development.

In the built environment there are also physical loops, infrastructures that move objects and information. At every level of urbanization, from city center to the most remote areas, space is connected through human-scale circuits. From shipping routes to telecommunication networks, the city breathes through these loops, circulating the people, material, and information that bring it to life.

Cities are made by executing urban programs. In her essay *An Internet of Things*,[8] Keller Easterling argues that “activity in a spatial environment is not reliant on the digital environment. It may be enhanced by a code/text-based software, but a spatial software or protocol can be any platform that establishes variables for space as information.” The exchange of culture that flows through a city’s architecture and transportation is the software of contemporary space making. So…

Are cities computers for humans?

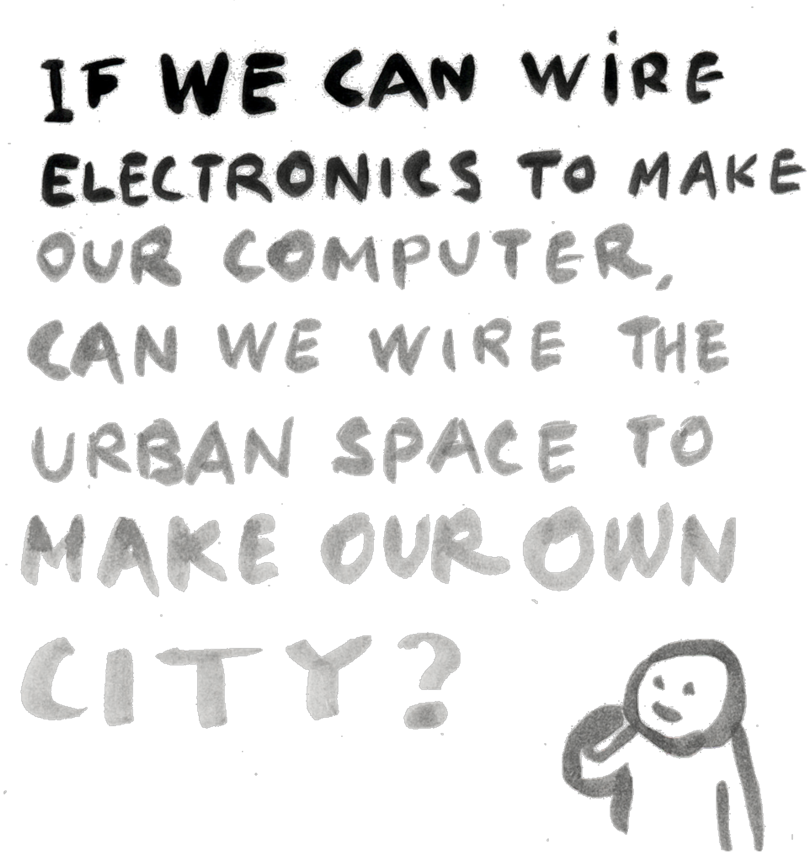
While the binary computer executes programs with precision, cities are most often unpredictable, even unquantifiable. I’m reminded of Jacques Tati’s film *Playtime*, where the choreography of urban life unfolds as programmed chaos and serendipity. If cities are computers for humans, they run buggy software and often fail to compile.

Jacques Tati, *Playtime* Trailer (1967)

Shannon Mattern critiques notions of the “smart city” that conflate data with memory, and decision with execution. Instead of reducing the city to a computer, she challenges us to “recognize spatial intelligence as sensory and experiential; that consider other species’ ways of knowing; that appreciate the wisdom of local crowds and communities; that acknowledge the information embedded in the city's facades, flora, statuary, and stairways; that aim to integrate forms of distributed cognition paralleling our brains' own distributed cognitive processes."[9]

Perhaps some qualities of cities are uncomputable, like the unpredictability of the environment or the ungraspable nature of the human mind.

Thinking about similarities and differences between the city and the computer helps us recognize the cultural complexities that surround them. By opening the computer to look at the inner workings, we may begin to ask: what kind of computers do we want to use and what types of cities do we want to live in?


IF WE CAN WIRE ELECTRONICS TO MAKE OUR COMPUTER, CAN WE WIRE THE URBAN SPACE TO MAKE OUR OWN CITY?

Chapter 2, Skating the Circuits →

Taeyoon Choi is an artist, educator, and activist based in New York and Seoul. His art practice involves performance, electronics, drawings, and installations that often form the basis for storytelling in public spaces. His projects were presented at the Whitney Museum of American Art and Los Angeles County Museum of Art. He co-founded the School for Poetic Computation where he continues to organize sessions and teach classes.

"Hello, World!" is the first of seven chapters from *Handmade Computer*, a book written by Taeyoon Choi and edited by Sam Hart. Chapters will be released bi-weekly during the summer of 2017.

[1] Richard Barbrook and Andy Cameron, *The Californian Ideology*, Mute Magazine, September 1995. Accessed 04 Jun 2017.↥

[2] "The Californian Ideology," Wikipedia, last modified August 27, 2016.↥

[3] "Vendor Lock-in" Wikipedia, last modified March 4, 2017.↥

[4] *Stan Augarten: A Photographic History of the Integrated Circuit*, State of the Art, Ticknor & Fields (1983), USA.↥

[5] Peter Halley, *Instant City* (1996), Acrylic, Day-Glo acrylic, Metallic acrylic & Roll-a-Tex on canvas 58 x 93 inches.↥

[6] Molly Wright Steenson, *Problems Before Patterns: A Different Look at Christopher Alexander and Pattern Languages*, Interactions, Volume 16 Issue 2, March + April 2009 Pages 20-2.↥

[7] Christopher Alexander, *A Pattern language: Towns, Buildings, Construction*, Oxford University Press (1977), USA.↥

[8] Keller Easterling, *An Internet of Things*, Eflux, January 2012. Accessed 04 Jun 2017.↥

[9] Shannon Mattern, *A City Is Not a Computer*, Places Journal, February 2017. Accessed 04 Jun 2017. ↥

SYLLABI INCLUDING THIS TEXT

- The Nature of Code, Processing Foundation