DigitCult | Scientific Journal on Digital Cultures

Prototyping Spatial Interactions

Pierluigi Dalla Rosa
http://www.pierdr.com/

Abstract

A growing field in the interaction design practice is the domain of connected and interactive spaces. The idea of ubiquitous computing, that was originated in the late 1980s and the beginning of the 1990s, is now seeking maturity, and technology trajectories have helped materialize the basic prerequisites for the proliferation of spatial computing. This paper illustrates how the design process for interactive environments can benefit from an emerging set of methodologies and tools, that is the physical counterpart for the successful creation of human-centered interactive experiences in space. When creating interactive spaces, a designer needs to take into consideration new dimensions, like the flow of users within the space, the spatial context, and a new set of sensors and actuators that go well beyond screens, keyboards, mice, or controllers. Unlike other interaction design domains, there is not a defined device, like a smartphone or personal computer, but instead, the technologies of spatial computing are a collection of distributed devices, sensors, and actuators. Due to this configuration the implementation and prototyping of interactive environments requires new tools to deal with the complexity of non-standard components to ease the design process and allow for sketching in hardware and software at scale. With ad hoc software and hardware designed with the purpose of standardizing and making accessible these components, a design practitioner can implement prototypes of interactive and connected spaces to gather inspirations, insights, and validation before investing in creating the complete experience. This paper explains how the process of designing with such tools will transform the work and the outcome of the interaction designer that explores the forefront domains of connected environments, ubiquitous media, and spatial computing.

Published 2 May 2021

Correspondence should be addressed to Pierluigi Dalla Rosa. Email: me@pierdr.com

DigitCult, Scientific Journal on Digital Cultures is an academic journal of international scope, peer-reviewed and open access, aiming to value international research and to present current debate on digital culture, technological innovation and social change. ISSN: 2531-5994. URL: http://www.digitcult.it

 doi:10.53136/979125994120639

Introduction

Everyone that got in contact with digital technologies experienced the fruits of interaction design. From remote villages to tier-one cities, from smartphones to elevators, digital technologies spread into everyday objects. Personal digital devices, first computers than smartphones, became important and indispensable companions to contemporary life, but are just one instantiation of the digital.

Nicholas Negroponte predicted the spread of computation outside computers already in 1998 when he stated that "computers, as we know them today, will be boring, and disappear into things that are first and foremost something else: smart nails, self-cleaning shirts, driverless cars, therapeutic Barbie dolls, intelligent doorknobs [...] Computers will be a sweeping yet invisible part of our everyday lives: We'll live in them, wear them, even eat them".

This trajectory is materializing with rising of embedded computing and connected objects. For example, large consumer electronic manufacturers, today, have in their product lines home assistants, like the Apple Home Pod or Google Home. These are new media touch-points, that blend the digital world with new forms of interactions in spaces. The fact that there are numerous players in the field of connected appliances is an indicator of the maturity of a set of technologies that are enabling computation to spread outside traditional devices like computers and smartphones and become part of objects, like the iRobot Roomba, an autonomous vacuum cleaner, or Philips Hue, a smart lighting system where light bulbs with a micro-controller and wireless connectivity can be programmed from anywhere.

The spread of low-cost networked micro-controllers allows for new possibilities not just in single connected objects but in creating spaces that are characterized by distributed touch-points, enabling multi-user interaction in physical space.

There are many examples of these if we take into consideration interactive museum exhibitions, interactive experiential marketing campaigns, escape rooms, or alternate reality games. More and more of these experiences pioneered in public space are now permeating people's homes as described in the article of December 2020 in the Scientific American where spatial computing is quoted as one of the top emerging technologies and is quoed to be at the heart of the ongoing convergence of the physical and digital worlds (Lathan 2020).

Computing Systems and Spatial Interactions

Every computing system, in the most essential form, has three components: input, processing, output. The Analytical Engine by Charles Babbage, designed in 1837, already manifested these components; in the description of Babbage's computer the input, consisting of programs and data, was to be provided to the machine via punched cards [...]. For output, the machine would have a printer, a curve plotter, and a bell (Collier 1970).

Everyone that uses a computer today is using a system that conforms to this model, having inputs like the keyboard and mouse, and outputs like a monitor or a printer, while the processing happens inside the computing unit. This model has roots in the early days of computing and it is commonly referred as the Von Neumann architecture (Goldstine 1993).

It is possible to apply the label spatial computing to those instances of digital systems that don't have a predefined set of inputs and outputs and break the paradigm of personal digital devices. Aiming to define a growing domain that has divergent characteristics in terms of the user interface but shared digital architecture compared to current digital systems. The wider consequences of the reach, memorability, and engagement of spatial computing systems are a long way off the scope of this paper, so within it, this classification will be used mainly for technical utility.

When designing spatial computing systems without a predefined set of inputs and outputs the focus is on understanding the affordances, mental model, and general usability of different touch-points that populate new devices or interactive spaces; the challenge is also an opportunity to redefine new forms of inputs and outputs that suit the problem, the destination and the purpose of the new digital structure.

A Prototyping Tool for Interactive Spaces

Interaction designers sketch with prototyping tools to get inspiration and test their ideas, before delegating engineering teams the full implementing of a final solution.

When designing interactive spaces, the ideal process follows a similar path, with simulation and prototyping ahead of implementation, but the different nature of interactive spaces, compared with digital products, make it more time consuming and with a higher degree of complexity.

Prototyping tools can help the designer to simulate the experience, or parts of it, so that some test-users can be immersed in the simulated experience and provide early feedback and inspiration to the design team.

The first pioneers to create experimental tools for interactive spaces have been John Underkoffler and Hiroshi Ishii, that started expanding the role of computation with projection mapping; they used a projector to display a computer image that is mapped to real, physical features. One example is their project Urp, developed at MIT, and it is an urban planning tool that shows information about shadows, proximity, reflections, wind flow and visual space using small architectural models with a projection layered on the topographical surfaces.

A movement of designers just after the web-bubble, started hacking keyboards to create physical button that can help create physical interactions. The design consultancy Tellart LLC, pioneered this methodology in RISD, where they taught industrial design students how to use standard electronics to sketch in hardware. The students hacked keyboards to create physical actuation in order to open and close a circuit, triggering keypresses on a personal computer, or connected RFID readers as inputs for a webpage (Noble, 2012).

Today there are different electronic components that simulate a keyboard input. The makeymakey, iPac boards, Bare Conductive touch-boards or Playtronica boards.

These devices allow to simulate keypresses; these keypresses can be used to trigger a slideshow software, like PowerPoint or Keynote, or to play or sequence media, like video or audio. The designer, utilizing these boards, can focus on the creation of physical triggers for an interactive space. The physical space is rich of triggers like a person sitting on a chair, a door opening, a person touching a metal object, a loud sound, a change in light intensity, a gas detected in the air. In this paper physical triggers will be simply referred as buttons.

Buttons and projections are great first steps to prototype interactive spaces, because their purpose is to expand the reach of a single computing unit, both enhancing the inputs of a personal computer with distributed physical buttons, and in outputs, using projections to expand the reach of the screen in space. Of course, this is just the first step towards the creation of engaging interactions in space. Most of the time designers want to enable more complex sensing capabilities and more actuations beyond projections. In fact, actuation expands beyond pixels using motors, lights, relays, etc.

The complexity of prototyping these experiences is proportional to the expensive palette of sensors and actuators.

A simple approach in creating an interactive space, like the one described above, will be to consider every element of the system as just a set of inputs of sensors and actuators, while the central logic is confined in just one single place.

If recalling the above example, it is possible to imagine that at the beginning of the project the designer wants to test their initial concept at scale, or he wants to check that all the stages that have been graphically developed are cohesive and make sense in their logical flows.

There are different low-fi approaches (WOZ, Wizard of Oz) that we can use today that will allow testing these at scale, involving presentation software and manual actuation.

These approaches are great, fast to implement, and do not require the creation of software. They are, on the other side, limited and imprecise, and most of the time they do not convey an experience similar enough to the final one. Here is where *Tramontana* can excel instead.

In fact, the designer-developer could, in a matter of hours, open the *Tramontana* receiver software, to transform any device (tablet, smartphone, personal computer, or electronics board) in a set of inputs and outputs, and from a centralized CPU implement a basic logic that orchestrates all the pieces. The *Tramontana* software could pilot the content of a projection, multiple screens, or physical led lighting. To do that a designer-developer can rely on the *Tramontana* receiver software and the *Tramontana* library.

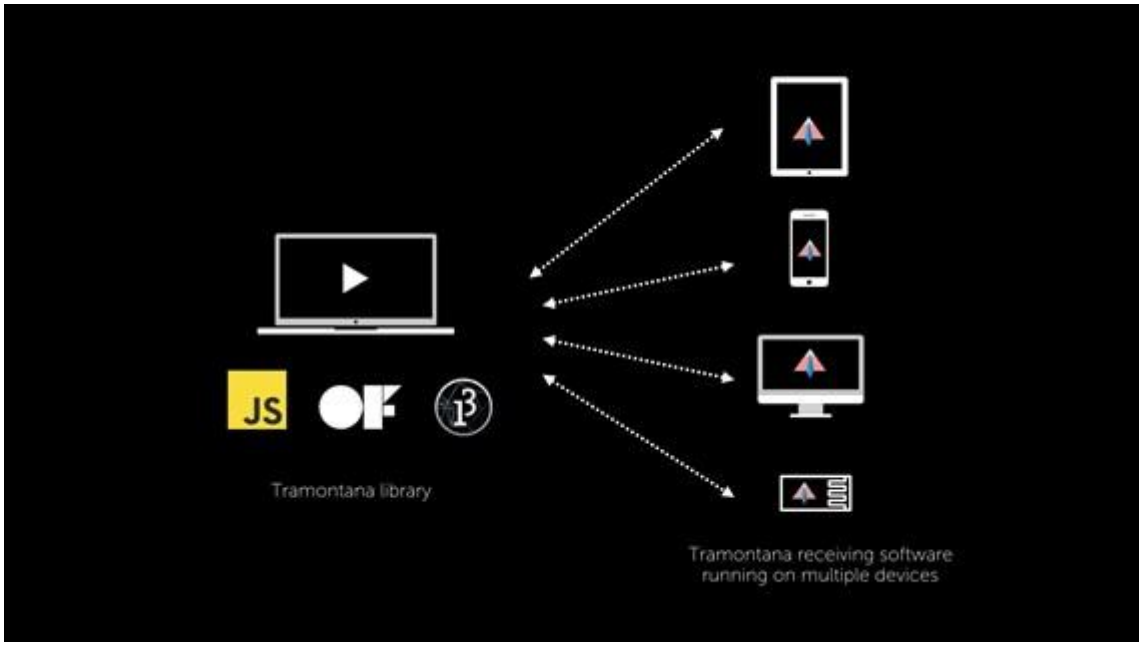
The *Tramontana* receiver software runs on tablets, smartphones, personal computers, or custom electronic-boards that are running the ad-hoc *Tramontana* implementation. The

 doi:10.53136/979125994120639

Tramontana receiver software is distributed in form of an app, and it is available on the major app stores for different platforms like Android, iOS, macOS, and tvOS. The *Tramontana* board instead is pre-loaded with the *Tramontana* receiver software.

The *Tramontana library*, instead, is a framework for creative coding platforms, like openFrameworks, Processing, or JavaScript. Functionally a sketch in any of these creative coding platforms with the *Tramontana* library is the master-mind where all the logic is created. The centrally controlled logic keeps a connection with all the other devices running the *Tramontana* receiver software, and each of those can be used as input or output as desired.

So, within the sketch, a designer can create links to every single node that needs to play a part in the experience.




```
import tramontana.library.*;
Tramontana node1;

void setup(){
      node1 = new Tramontana(this,"192.168.1.3");
}

void mousePressed(){
      node1.makeVibrate();
}
```

In the simplest sketch possible the *Tramontana* library is imported in the desired language (Java/Processing in the example above) and a software object of class `Tramontana` is the interface to access an external device running the *Tramontana* receiver software. This software object (referred to as node-link object) is the interface for the designer to act on the node, changing the behavior of the output or reading sensor data from the external device.

At runtime, the node-link object is initialized, opening the connection with the physical node, reading a parameter that informs the library of the IP address of the device running the *Tramontana* receiving software.

In the background, *Tramontana* creates a network link between the node and sketch that the designer does not need to be knowledgeable about. The designer creates the logic in one single location, the sketch, without any need to create any custom software for the nodes that are simply acting like inputs or outputs of the master-mind, the sketch.

The node-link object exposes the actuation provided by the node, the physical device; for example, if the node is a smartphone the actuation that are possible are:

- manipulate the screen content;

- display media (audio, video, images);
- actuate the vibration motor;
- turn on and off the camera's flashlight.

The *Tramontana* hardware board is a different kind of board and allows a different kind of actuation, like:

- setting relays on and off;
- changing the rotation of a servo motor;
- manipulate the color of a led strip.

In the case of prototyping Rain Room, a designer-developer could seek in *Tramontana* an essential toolkit to start exploring spatial interactions, like motion and movement. A designer-developer in fact could swiftly connect up to 4 relays on a single *Tramontana* board to stop or activate the downpour without needing any embedded software.

The *Tramontana* board can act as a sensor as well. In the case of Rain Room, a designer could connect a motion sensor directly to the *Tramontana* board and access that via the *Tramontana* library.

To perform the sensor reading *Tramontana* conforms to standardized ways to capture a system call or input event.

Tramontana uses the same paradigm and syntax used for listening to the key pressed or mouse events in the available platforms.

In the case of a smartphone, the smartphone itself can be seen as a sensor:

- touch screen;
- accelerometer and gyroscope;
- camera;
- microphone;
- distance sensor;
- audio jack;
- power socket connected/disconnected.

```
import tramontana.library.*;

Tramontana t;
void setup(){
      t = new Tramontana(this,"192.168.1.18");
      t.subscribeAttitude(5);
}
void onAttitudeEvent(String ipAddress, float newRoll, float newPitch,
float newYaw)
{
      …
}
void mousePressed()
{
      …
}
```

Following standard syntax and abstracting different computing units to unified logic, *Tramontana* reduces the technical complexity and allows the implementation of a prototype to follow the behavior of the installation, helping the designer to follow what is important the experience.

Through high-level logical steps instead of an engineering endeavor, it is possible to construct the state machine of the installation or interactive space. The distributed nature of spatial computing is hidden by the *Tramontana* framework and each device running the

 doi:10.53136/979125994120639

Tramontana receiver software is synchronized via network on behalf of the designer-developer without requiring complicated setups and it minimizes the debugging time.

In short, *Tramontana* makes it possible to sketch an interactive experience to test the concepts, before worrying about implementation.

Hardware Platform

Personal digital devices, like smartphones and tablets, are convenient to use for the purpose of prototyping because they are already in the pockets of most, and devices can be great in simulating, not only digital experiences but physical ones as well; in interactive spaces, the distribution of one or more devices can already give the perception of spatiality and those are great to simulate the pressing of a button or the triggering of a light.

To achieve a higher level of fidelity a hardware board can be plugged into the *Tramontana* ecosystem.

Tramontana's hardware platform is a WiFi-enabled board that ships with a firmware that handles a protocol to interact with the *Tramontana* library. The hardware board has an onboard RGB LED, a light sensor, and a set of pre-defined connectors that can be used to control simple hardware components.

The actuators that can be connected are a light strip, two concurrent servo motors, and space for four relays. The sensors available to the plugin are a motion sensor (PIR sensor), two buttons, and two generic analog inputs.

The software syntax to interact with the *Tramontana* hardware board appears to be exactly the same as other devices, the *Tramontana* library, in fact, keeps track of the nodes that are connected. Every *Tramontana* device has different characteristics, for example, a smartphone running the *Tramontana* receiving software will be able to pilot different inputs and outputs then the hardware board running the embedded flavor of the receiving software. The *Tramontana* library keeps track of the kind of device connected and notifies the designer if a wrong set of commands is sent to a wrong node-link object. For example, if a node-link object connects to a hardware board and the user invokes a method to trigger the haptic motor, the platform will notify that the connected device is not able to process that command (reserved for smartphones).

TramontanaCV

The mechanisms that allow *Tramontana* library to connect to different devices, can be utilized to work for a specific use case that is particularly important when working in the field of spatial computing. Understanding people's positioning in space is an important part of the prototyping and creating interactive spaces and it is one of the main barriers when testing interactions in space. *TramontanaCV* is a specialized node that has the aim of sensing people in space using computer vision. *TramontanaCV* includes an interface that allows the designer to harness the GUI, camera, and computing power of a smartphone to perform blob tracking and stream just the relevant information to the *Tramontana* library.

Within the centralized logic, the information regarding users' position can influence the behavior of the interactive space.

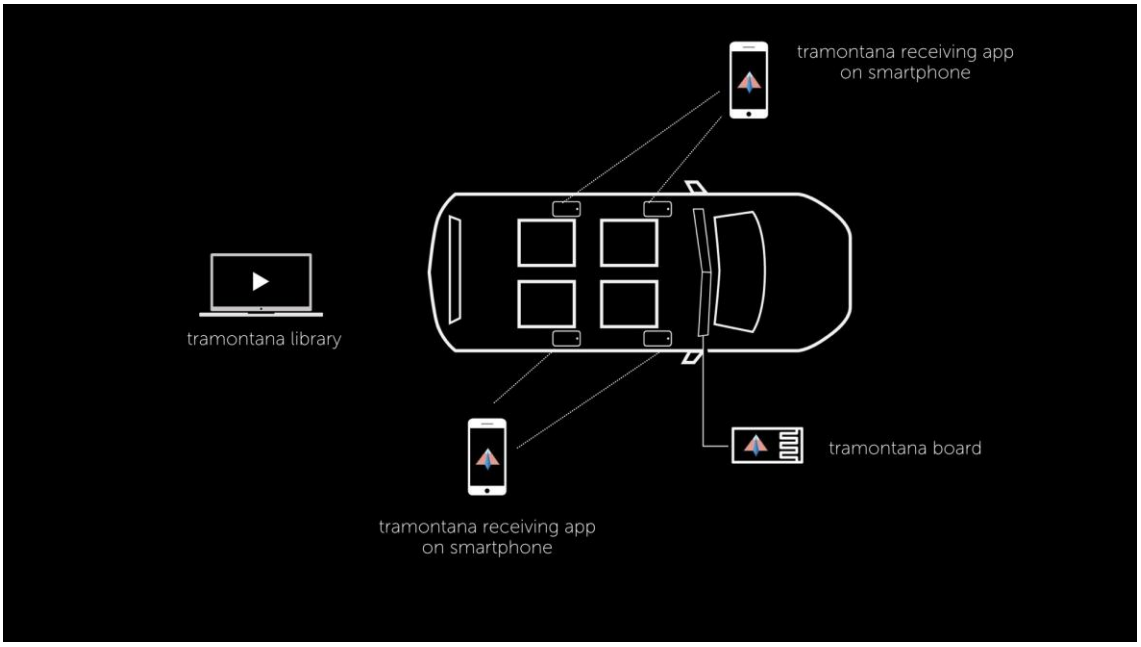
TramontanaCV is just another set of inputs for the interactive system and the syntax, in this case, is compliant with traditional inputs; the software interface to access the data, in fact, follows the same conventions as a keypress or any other *Tramontana* inputs.

```
void onBlobsReceived(LBlobsContainer c,int nBlobs, String ip){
      …
}

void onBoundingBoxReceived(LBBoxContainer c, int nBlobs, String ip){
      …
}
```

An Example of Prototyping Spatial Interactions

A car cockpit is becoming an interactive space filled with digital features and the passenger experience is more and more taken into account.



Let us consider the design of the car system that manages an incoming phone call and let us imagine that the designer identified that each passenger in this scenario should have the means of answering the call. The car notifies the passengers with ambient light and sound of the incoming call and displays the name of the caller on the screen used for the radio. Every passenger has an interface with which they can answer and subsequently terminate the call.

At this stage, the designer is faced with the challenge of prototyping the system. Traditionally the process will involve wiring physical ambient lighting, creating intelligence on different computing units to synchronize the state machine that takes into account the state of the call (idle, call incoming, call in progress, call terminated), connecting the buttons for each passenger with a serial communication protocol.

With the methodology described in this paper, it is possible to skip most of the time consuming and complex steps in this prototype.

The designer architects all the intelligence within one sketch, taking 4 different smartphones running *Tramontana*, nodes, and listens to the press of a node's touch screen as the interface for answering and hanging up the call. The screen of another smartphone or tablet could serve as the screen for the audio system where the name of the caller will be displayed. To simulate the incoming call a *Tramontana* board will activate the onboard ambient lights, and an audio cue can be played by every single node in the space. Every button is now active and ready to listen to the user's action to answer the call. When the call is ongoing every button can be used to hang up. When any of the nodes is triggered, the call is terminated and all the nodes are reset to an idle mode.

All of these interactions and behaviors are managed by the designer in the sketch with a few lines of code and a clear mental model of how the prototyping system behaves, without the need for deep expertise in networking, state machines, or electronics.

With the *Tramontana* ecosystem, the designer can test the experience of spatial computing environments in a very short time and can swap, change and iterate with extreme ease, making the process faster, more iterative, and ultimately more effective.

 doi:10.53136/979125994120639

Conclusions

The most important concept behind *Tramontana* is its architecture. The *Tramontana* ecosystem centralizes the logic of a complex computing system in one location, a code sketch running the *Tramontana* library while keeping multiple inputs and outputs distributed and physically detached from the central logic without adding significant complexity to the system. The ecosystem is extensible thanks to its flexible structure and can host new smart nodes, as demonstrated by the addition of *TramontanaCV*. In fact, nodes can benefit from onboard processing power to execute distributed tasks like computer vision and machine learning. Doing so will be functional to design more complex systems and will enable new interaction opportunities like pattern recognition, pose tracking, identity, and more.

On a different level, the *Tramontana* ecosystem can be expanded with new generic devices, like smartwatches (not yet supported), or with custom devices built for a specific application. Both custom and consumer devices can be part of the *Tramontana* ecosystem if they conform with the universal *Tramontana* API and follow the standardized socket's protocol that connects devices to the *Tramontana* library. These new additions will be no different conceptually than any previous work on the platform and will allow new explorations and possibilities in the field of ubiquitous computing and spatial interactions.

Another area of investigation is the central logic, or the sketch, that at the moment of writing is a program, written within a pre-existing creative coding platform, that uses the *Tramontana* library. In the future, the logic for a spatial computing system can live within a flow-based programming environment, where the logic is not bound to code and can be made even more accessible (Morrison 2010).

The prototyping stage is crucial in the interaction design practice, and while there are many tools that help in the prototyping phase for mobile and web applications, there are very few that are tailored to the emerging field of spatial interactions.

This paper highlights a new way of approaching the prototyping stage in the design of spatial computing systems, allowing the designer to skip complicated steps when approaching the introductions of digital touch-points in prototypes, and allowing the designer to focus on the experience instead of the technology. In the practice of interaction design, it is also essential to understand the digital materials, and prototyping tools are great entry points to explore possibilities and gather inspiration. The *Tramontana* ecosystem allows not only to create quick prototypes but also allows designers to explore, with low entry barriers, the creation of spatial computing systems; in doing so designers build new practical knowledge and nurture inspiration without the burden of the mental complexity and time-consuming execution of computer science techniques that they will require otherwise.

References

Banzi, Massimo, and Michael Shiloh. *Getting Started with Arduino.* Sebastopol, CA: Maker Media, 2014.

Buxton, William. *Sketching User Experiences: Getting the Design Right and the Right Design.* Amsterdam: Elsevier, 2011.

Collier, Bruce. *The Little Engines That Could've: The Calculating Machines of Charles Babbage.* Accessed October 10, 2018. Available at http://robroy.dyndns.info/collier/

Goldstine, Herman H. *The Computer from Pascal to Von Neumann*. Princeton University Press, 1993.

Greenfield, Adam. *Radical Technologies: The Design of Everyday Life*. London: Verso, 2018.

Houde, Stephanie, and Charles Hill. “What Do Prototypes Prototype?” *Handbook of Human-Computer Interaction* (1997): 367-81. doi:10.1016/b978-044481862-1.50082-0.

Ishii, Hiroshi, and John Underkoffler. *Luminous Room/Urp*. 1996. Available at: https://www.media.mit.edu/projects/luminous-roomurp/overview/

Lathan, Corinna E., and Geoffrey Ling. "Spatial Computing Could Be the Next Big Thing." *Scientific American* (November, 2020). Available at: https://www.scientificamerican.com/article/spatial-computing-could-be-the-next-big-thing/

Moggridge, Bill. *Designing Interactions*. Cambridge, MA: MIT Press, 2006.

Morrison, J. Paul. *Flow-Based Programming: A New Approach to Application Development*. International Thomson Computer Press, 2010.

Noble, Joshua. *Programming Interactivity.* O'Reilly Media, 2012.

Verplank, Bill. *Interaction Design Sketchbook*. Accessed online October 9th, 2018. Available at: https://hci.rwth-aachen.de/tiki-download_wiki_attachment.php?attId=797

Wired staff. "Negroponte." *Wired Magazine* (6.12.98). Accessed online October 9th, 2018. Available at https://www.wired.com/1998/12/negroponte-55/