

Bricoleur:

Creative Learning through Video and Computation

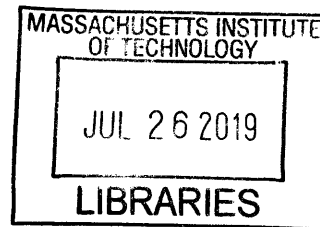
by
Sean Hickey

B.A. Mathematics, Macalester College, 2010

Submitted to the Program in 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
June 2019



© Massachusetts Institute of Technology, 2019. All rights reserved.

Author

Signature redacted

Program in Media Arts and Sciences
May 10, 2019

Certified by

Signature redacted

Mitchel Resnick
LEGO Papert Professor of Learning Research
Program in Media Arts and Sciences
Thesis Advisor

Accepted by

Signature redacted

Tod Machover
Academic Head
Program in Media Arts and Sciences

Bricoleur:

Creative Learning through Video and Computation

by
Sean Hickey

Submitted to the Program in 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
on May 10, 2019.

Abstract

In his theory of constructionism, Seymour Papert asserted the importance of *bricolage* in the learning process. Papert defined bricolage as “a style of organizing work that can be described as negotiational rather than planned in advance.” The bricoleur — one who engages in bricolage — works in dialogue with their materials, continuously experimenting and course-correcting as necessary. Throughout this process, the bricoleur creates relations between materials and in doing so, builds mental relations in their head among ideas and knowledge. In this sense, creating with materials *is* the thinking process and as a result, Papert claimed that “in the most fundamental sense, we, as learners, are all bricoleurs.” In light of this, it is important to develop rich bricolage contexts and experiences to support learners in constructing their own knowledge.

This thesis describes Bricoleur, a new tool for creating expressive projects in a bricolage style. Bricoleur builds off of the programming paradigm developed for the Scratch programming language to allow makers to create a wide variety of dynamic projects by capturing and programming video and audio media on tablet hardware. We describe the design decisions that led to the creation of a *tinkerable* tool that engages both the mind and body. In addition, we outline the types of projects and working styles that emerged during creative workshops in which makers created projects with Bricoleur. We then look at some broad outcomes of the work, noting that Bricoleur enables young makers to engage not just with computation and media, but also with their bodies, their environment, and the people around them. Through this process, makers encounter ideas about space, place, people, and time. We conclude with some reflections on future directions for the tool, mobile programming in general, and new possibilities of creative contexts for bricolage.

Thesis Advisor:
Mitchel Resnick
LEGO Papert Professor of Learning Research
Program in Media Arts and Sciences, MIT Media Lab

Bricoleur:

Creative Learning through Video and Computation

by
Sean Hickey

Signature redacted

Advisor

Mitchel Resnick
LEGO Papert Professor of Learning Research
Program in Media Arts and Sciences, MIT Media Lab

Bricoleur:

Creative Learning through Video and Computation

by
Sean Hickey

Signature redacted

Reader

Glorianna Davenport
President, Living Observatory
Visiting Scientist, MIT Media Lab

Bricoleur:

Creative Learning through Video and Computation

by
Sean Hickey

Signature redacted

Reader

S. Alex Ruthmann
Associate Professor, Music Education and Music Technology, New York University
Director, NYU Music Experience Design Lab (MusEDLab)

Table of Contents

Acknowledgements	20
Preface.....	24
Three Vignettes	28
Introduction	32
Chapter 1: From Bricolage to Bricoleur	38
Chapter 2: Cultural Influences and Inspirations	48
Chapter 3: Design.....	60
Chapter 4: Workshops	70
Chapter 5: Outcomes.....	86
Chapter 6: Reflections and Future Work.....	92
Appendix A: Technical Notes	103
Appendix B: Idea Spark Cards	111
Appendix C: Bricoleur Cards	113
References	115

List of Figures

Figure 1 – (Left) The editor for a complete project. (Right) Diagram of the different parts of the editing interface.....	43
Figure 2 – A blank Bricoleur project.	44
Figure 3 – Drawing a video mask.	44
Figure 4 – Recording a masked video clip.....	44
Figure 5 – Trimming and adding markers to a video clip after capture.	45
Figure 6 – Programming the video clip with blocks.....	45
Figure 7 – Example programming blocks for video clips.	45
Figure 8 – Trimming and adding markers after capturing audio.	46
Figure 9 – Example blocks for audio assets.....	46
Figure 10 – An audio script that loops and layers a sound.....	46
Figure 11 – Blocks for interacting with tablet hardware sensors.	47
Figure 12 – Still from Maya Deren’s <i>Meshes of the Afternoon</i> , shot on a 16mm Bolex camera. (credit: public domain)	52
Figure 13 – A Sony AV-3400 Portapak camera system. (Photo by Wikipedia user Mwf95, 2005. CC BY-SA 4.0).....	53
Figure 14 – Still from <i>Switch! Monitor! Drift!</i> by Steina Vasulka, 1976.	53
Figure 15 – The Fisher-Price PXL-2000 “pixelvision” camera. (Photo by Joe Lillibridge, 2008. CC BY-SA 2.0).....	54
Figure 16 – Still from <i>Me & Rubyfruit</i> by Sadie Benning (age 16), shot on a PXL-2000, 1989.....	54
Figure 17 – The Scratch project editor.....	56
Figure 18 – A ScratchJr project.	57
Figure 19 – The Eisenstein editor.	64
Figure 20 – Capturing an image of a face inside a sprite in ScratchJr.....	66
Figure 21 – Examples of projects made with Steina. (Top) A face mashup. (Bottom) A side-scroller game.....	67
Figure 22 – Materials setup for a Bricoleur workshop.....	74
Figure 23 – Spark Card decks showing an example draw.	74
Figure 24 – Example of a Bricoleur card demonstrating how to loop a video or audio clip.....	75
Figure 25 – Jay Silver’s sample project space. (credit: Jay Silver [15])	76
Figure 26 – A still from a story project about the moon.	76

Figure 27 – A “puppet show” project (also shown in Vignette Two) where each character can be tapped to trigger sound and animated movement..... 76

Figure 28 – A soundboard project where polyrhythms can be explored by tapping on each video clip to start a looped video and sound. 77

Figure 29 – A project exploring collaging together different images of concrete floor texture. 77

Figure 30 – A interactive tree made of body parts. Tilting the device causes the tree branches to sway. Tapping the tree causes apples (made of faces) to fall from the tree..... 77

Figure 31 – An interactive purse project inspired by the carrying case. The video clips respond to the tilt angle of the device. 78

Figure 32 – Two hologram projects running next to each other. 78

Figure 33 – A project where the video clip pans across the xylophone keys. As the device is tilted back and forth, the video frame continuously changes based on the tilt angle. 79

Figure 34 – A project that utilized video masking to define the shapes of the eye. The captured video images simply provide color and texture (e.g., the feathers in the pupil shape)..... 80

Figure 35 – Two makers exhibiting the “capturer” and “performer” division of roles. .. 81

Figure 36 – A video clip demonstrating the framing style of masking. The round drawn shape frames the cow in its surrounding context. 82

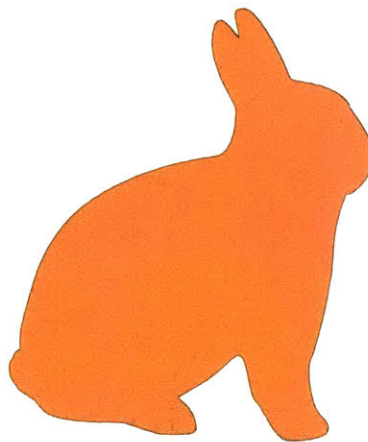
Figure 37 – An isolated character from the puppet show project shown in Figure 27.... 83

Figure 38 – A textured teardrop shape from the eye project shown in Figure 34. 83

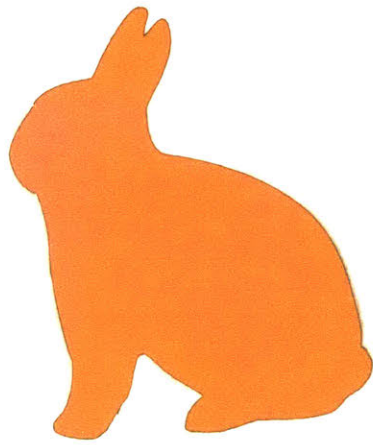
Some (Dis)assembly Required: A Note on Format

Hello, dear reader! I trust that you are well. In case you are reading a PDF version or a scanned, digital copy of this document, I wanted to let you know a few things...

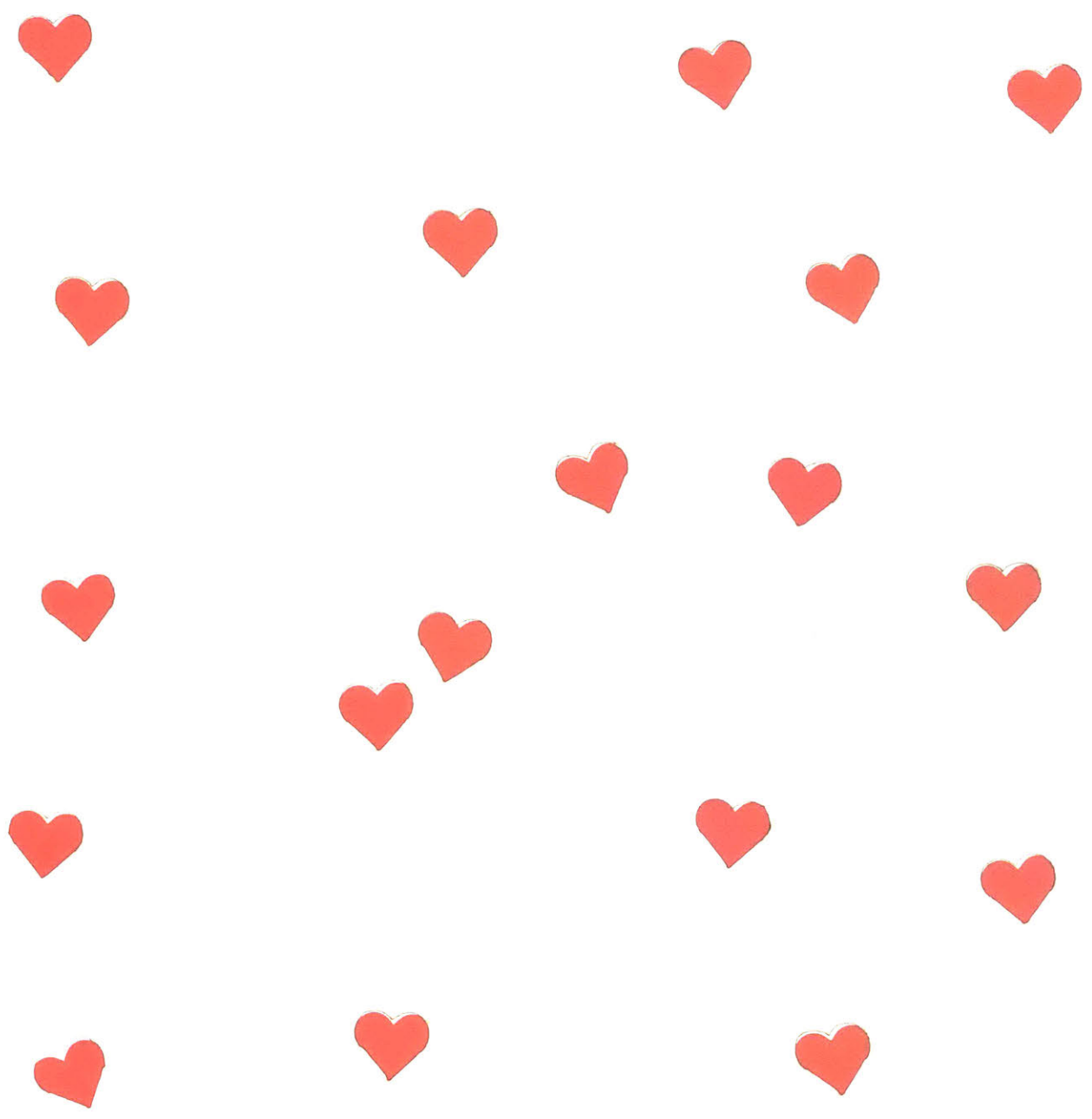
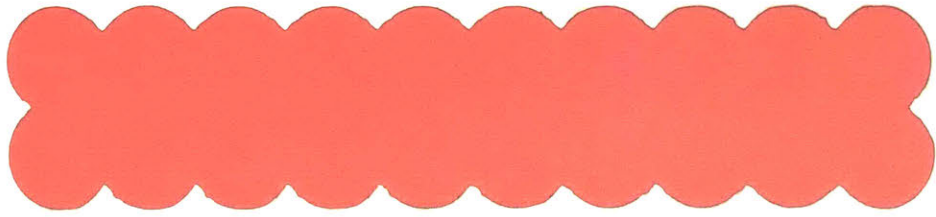
This thesis is, in fact, intended to be read in its physical form. Some pages in the physical version have been altered such that they have shapes cut out of them. When read as a bound, two-sided document, the cutouts allow you, dear reader, to see through to parts of the next (or previous) page. The way these cutouts work is highly intentional and probably only makes sense in the physical copy. In the PDF version, the cutouts appear as black outlines. I don't even know what the cut-outs in the scanned version will look like. So I put one here, shaped like a rabbit, so that you can find out for yourself:



So just know that anytime you encounter a shape like that in the document, it is meant to be a cutout. If you really want to read the thesis as it is intended to be read, feel free to print it out, two-sided, on letter-sized paper. Then cut out where all the cutout shapes appear, staple on the left, and enjoy.



Acknowledgements



Acknowledgements

To Carmelo, Shruti, and Kreg for paving the way ahead of me and guiding me on the journey of writing a master's thesis. And more importantly, for being wonderful friends.

To Jaleesa for going through this wacky two years in parallel with me.

To the folks at Scintillae Atelier in Reggio Emilia — Simona, Federica, Jennifer, and Lorenzo — for bringing this project to children halfway around the world.

To Haystack Mountain School of Crafts for reminding me that it's possible to be an educator, an artist, and a technologist all at once.

To David B, for always being willing to help me in whatever way you can.

To Moran, Marian, Yusuf, Manuj, Tina, Samarth, Lena, Yumiko, and Carolina for your camaraderie.

To Opequon Quaker Camp and the staff, counselors, and campers who make camp what it is and remind me why creative learning is so important. You continue to change my life for the better every year.

To Cynthia Solomon for being this project's biggest fan.

To Eric R for your inspiring playfulness, creativity, and wisdom when it comes to designing creative tools for kids.

To Eric R and Tina for the conversation that led to the hand-drawn shapes in this project.

To the folks from Creative Learning Network, the folks at the Flagship Clubhouse, and the youth at Artists for Humanity for participating in the workshops.

To Anne G for your ongoing mentorship, guidance, and deeply sardonic humor.

To Agnes, David, and Hane for testing early versions of this project.

To Jennifer J your help in thinking through this grad school thing.

To all the folks in School of HONK for providing some much needed balance and fun in my life in Cambridge.

To my Thesis Pants™ which provided the uniform I needed to get these words onto these pages.

To Willa for letting me be a part of your young life.

To Judith for your suggestion of getting a Thesis Uniform to write in.

Authorship is weird. No work exists in a vacuum and no project belongs to just one person. Here, presented in not much of a particular order, are folks who can claim co-authorship on this work for their help and guidance in ways big and small. I feel incredibly fortunate to have you all in my life.

- 
- To the Scratch Team for playtesting early versions of this project and for being an incredibly lively, funny, and creative group of people to work with everyday.
- 
- To the partners in the TiDA project (friends from the Tinkering Studio, LEGO Idea Studio, and Reggio Children Foundation) for seeing the potential in this project.
- 
- To you, dear reader, for bringing this document to life by reading it.
- 
- To Glorianna and Alex for being readers on this project, providing me with rich feedback, and helping get it across the finish line.
- 
- To Tom, my undergraduate advisor, for your support in getting me here.
- 
- To Mitch for so so much: for taking a chance on me by bringing me to Lifelong Kindergarten, for supporting and guiding me along the way, and for the incredible amount of trust and respect you give to all your students
- 
- To Natalie for your wisdom, your wit, your passion, and your compassion. I always smile when I see you.
- 
- To Eric S and Carl for talking video with me whenever.
- 
- To Lily for always remembering to bring an aesthetic sense to your work and for being a great collaborator.
- 
- To Andrew for your thoughts, feedback, and guidance on early versions of this project.
- 
- To Lizz and Kali for being my Destiny Friends, helping me grow as an educator and as a human, for keeping me close from afar, and for my Prince mug.
- 
- To my video friends from back in the day for making weird stuff with me.
- 
- To Sheila Pepe for helping me see that this is all worthwhile.
- 
- To Jake for inviting me to Haystack and for always being a friend.
- 
- To my family: Mom, Dad, Jimmy, Kate, Siobhan, and Kevin for your unconditional love and support.
- 
- To Steph, Brendan, Charles, and Anand for helping create a wonderful place to live these past two years.
- 
- To my siblings, the funniest people I know and the closest friends I will ever have.
- 
- To my parents, for everything.

Preface





Preface

One of my earliest memories is waking up on the morning of my third birthday. I was wearing blue pajamas and as I climbed out of bed and stepped out of the room I shared with my older brother, Jimmy, I already knew that it was a special day. I climbed down the stairs and walked into the kitchen where I saw — for perhaps the first time — a video camera. My parents had rented the bulky VHS contraption just for the occasion, and I was fascinated by it. It sat on the counter and I remember wanting to press the big red “record” button just to see what happened. The tape we made that day still sits on the shelf in my parent’s house.

We got our own family video camera a few years later — a little mini-VHS handheld with a black-and-white viewfinder. We pulled it out at every family birthday (of which there were many, having grown up with three siblings) and I always volunteered



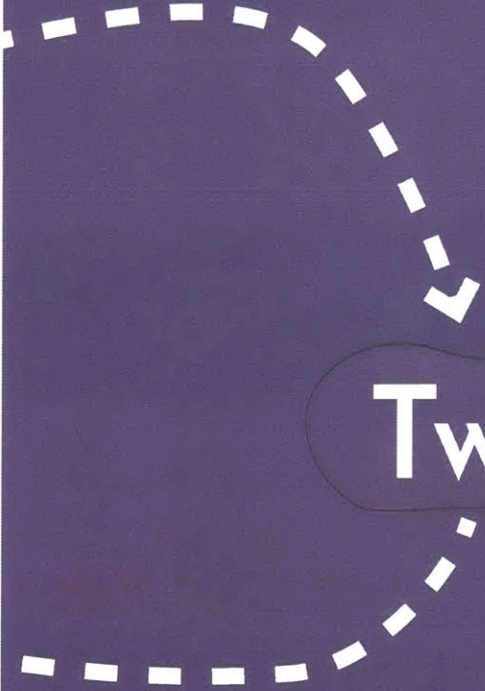
to be the one to capture the event of blowing out the candles. For school projects, I would always ask if I could make a video instead of doing a presentation. I loved being able to create in this medium that felt so intuitive and fluid to me. But the feeling I remember most — and it's a feeling I still have to this day — is how looking through the viewfinder instantly changed the way I could look at the world around me. By turning the real world into images and sounds, I could tell stories and explore ideas in a way that was otherwise impossible.

This thesis is ultimately about trying to find new ways to elicit that feeling in myself, and in others.

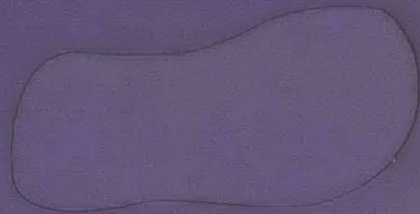
One...



Three Vignettes

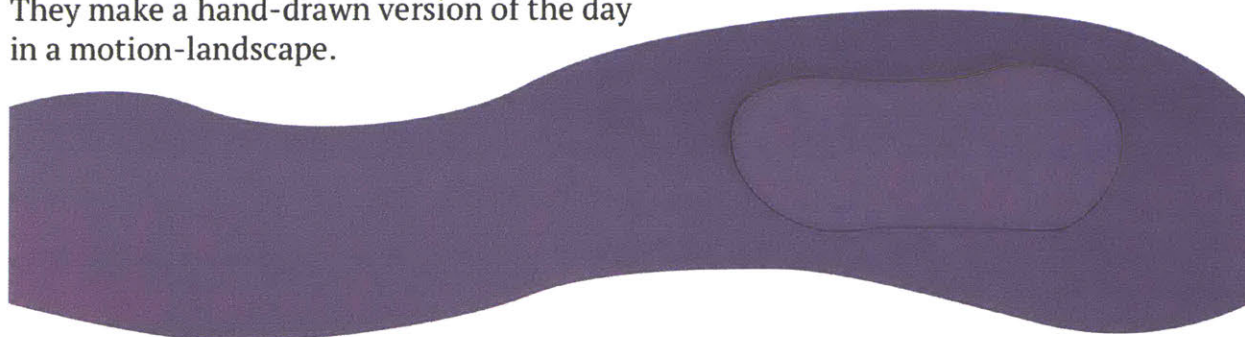


Two...



It's a beautiful, crisp, sunny day outside and two friends sit by the river, watching the gentle waves lap up on the bank. The colors are rich and they want to capture the moment to remember later. A photograph won't do, because the motion of the river and trees are too important, and a video simply can't fit all the detail in a single rectangular frame. They realize they can capture each piece of the scenery independently and recombine them into the scene they're admiring. They trace the shape of each part of the environment — the sun, the sky, the trees, the sand, the river — and capture a short video clip. They program each clip, so the sun shines and turns while the waves lap on the shore. They make a hand-drawn version of the day in a motion-landscape.

One...



Three



They love playing the little xylophone they keep on the desk. It's easy to pick out a quick melody on and the clarity of the notes always sounds beautiful. What if they could build their own instrument to play? They begin by capturing video and audio of the xylophone. By programming the video and sound, they can tap on the images of the instrument to get the sounds to play, but it seems like a dull recreation of the physical object. Maybe they can sing instead? They capture a video of a friend's face opening and closing their mouth, but forget to change the sound they've recorded. The xylophone tone comes out when the person opens their mouth! The effect is too rich to discard, so they continue capturing faces until all eight notes of the scale are accounted for. They've made a playable Xylo-Face!

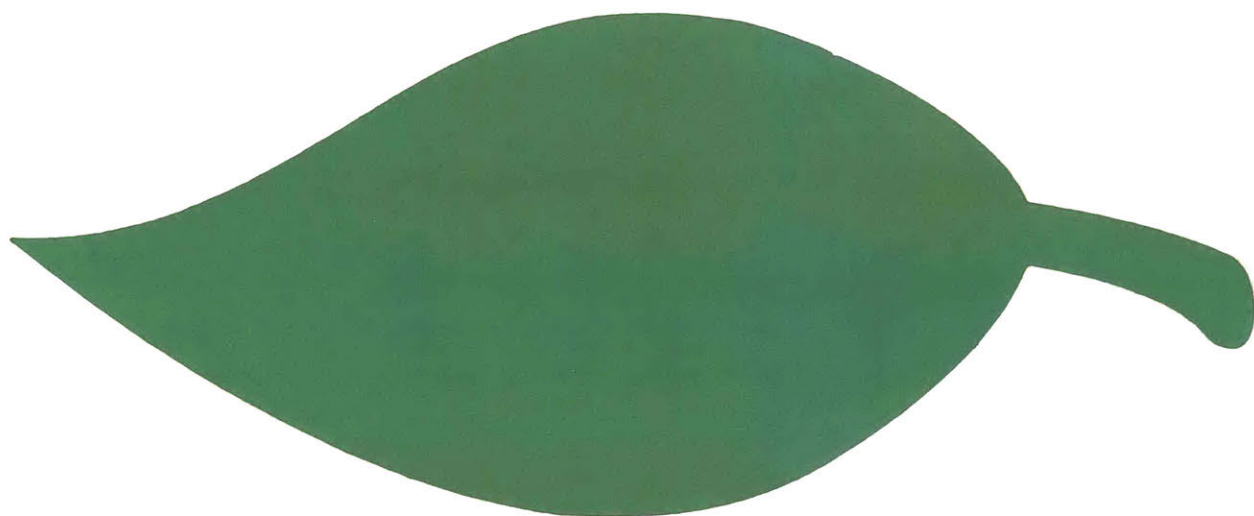
She rolls the clay between her hands forming a rough ball. Poking and prodding here and there, a face starts to emerge. A couple of pipe cleaners for arms and suddenly a character appears! She captures the blob creature from a few different angles, creating a collection of video clips.

Two...

A scene slowly takes shape with the characters on screen. By programming each creature, they move around the screen and interact when tapped. But what does a blob creature say? She records a couple of clips of gibberish language and programs the video clips so that tapping on each plays one of the sounds as well. A story unfolds: a performable blob dialogue!



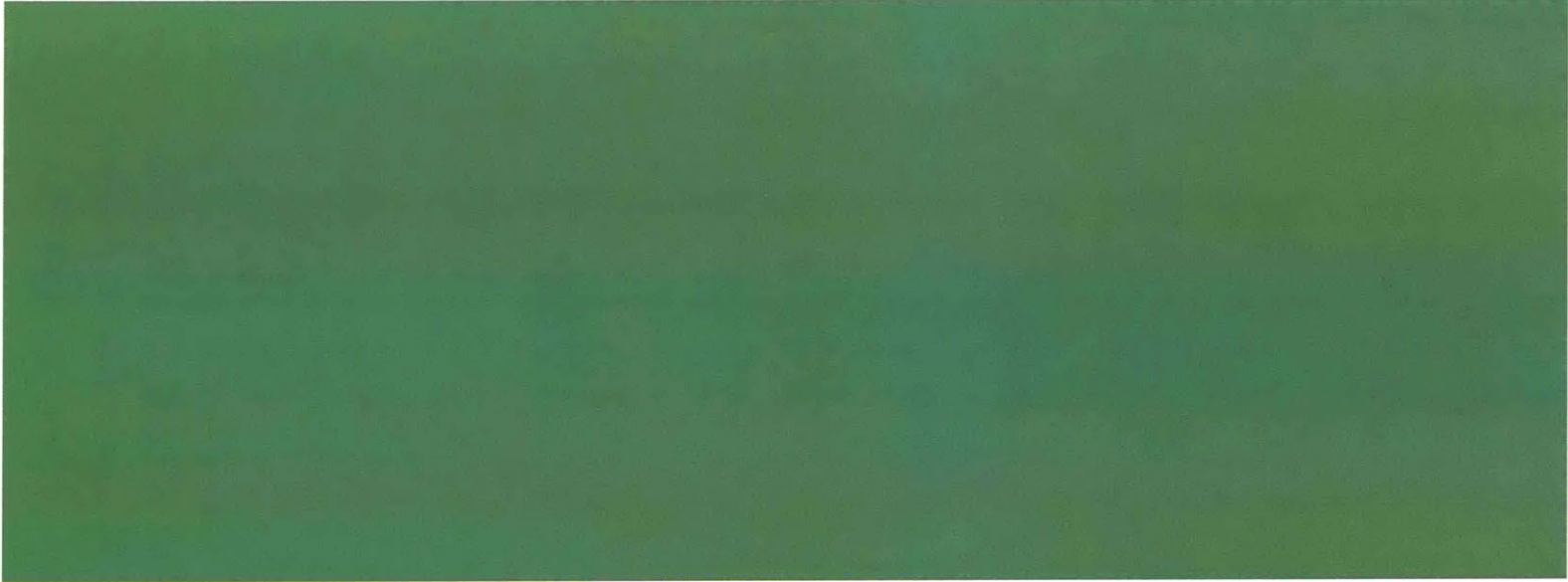
Introduction





A Walk In The Woods

To sum it up in a single sentence, this thesis documents the design, development, testing, and outcomes of a tool called Bricoleur and the new creative possibilities it affords. Bricoleur combines video, audio, programming, and mobility to open up a new space for creative expression. One challenge when writing about a new space of possibilities is that it can be hard to go into great depth about any one aspect of that space. Newness abounds and often times what is most interesting are the variety of ideas and connections between them rather than any one specific concept. In talking about this with my advisor, Mitch Resnick, he made a suggestion which I found useful in thinking about how to frame the work: think of it like traveling to a new land, exploring the terrain, then coming back to tell others about what you discovered.



I've been going to summer camp for nearly every year since I was nine years old — first as a camper, then a counselor, then a staff member, then the camp cook, and most recently, as the director. I've participated in, organized, and led my fair share of wilderness trips in the process, and know quite well the feeling of traveling in unknown and unexplored places. A wilderness trip involves some preparation and packing, traveling to the location, exploring the area, and eventually coming back to share stories of the experience with others. In many ways, this thesis aims to bring you, the reader, on exactly this kind of journey.

At the risk of overloading the analogy, we will add one more step to this journey. After exploring the area and before coming back to share our experience with others, we will do some work toward making a map of the space. Maps are relational in nature. They depict a simplified set of points that represent an infinitely more complex landscape and place those points in relation to each other. Simultaneously abstract and concrete, maps attempt to give a sense of the overall shape of the terrain they represent. In this sense, we can think of our mapping process as a kind of surveying, in the geographic sense of the word. We'll identify some key points in the space of creative possibilities, make connections between them, and thereby triangulate the terrain into something understandable and readable.

In total, our trip will entail:

Preparing (Chapter 1) in which we provide some theoretical underpinnings as well as an overview of the tool.

Packing Up (Chapter 2) in which we examine a set of tools that have served as cultural influences and inspirations for Bricoleur.

Heading Out (Chapter 3) in which we look at the design process that brought us to our new Bricoleur landscape.

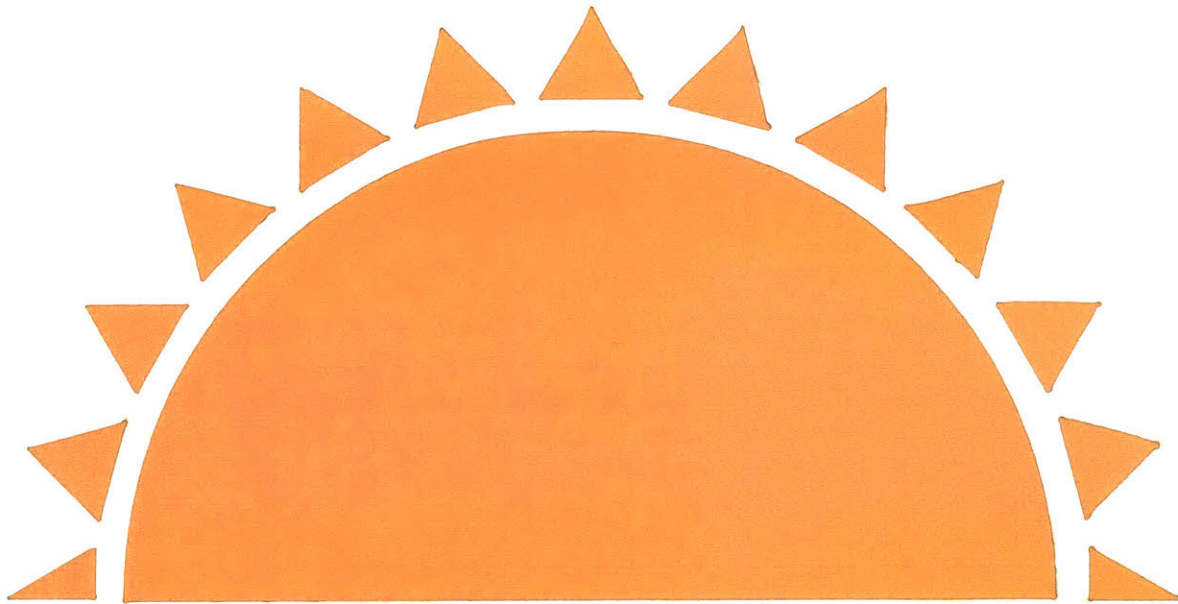
Exploring (Chapter 4) in which we examine how and what makers create in that landscape.

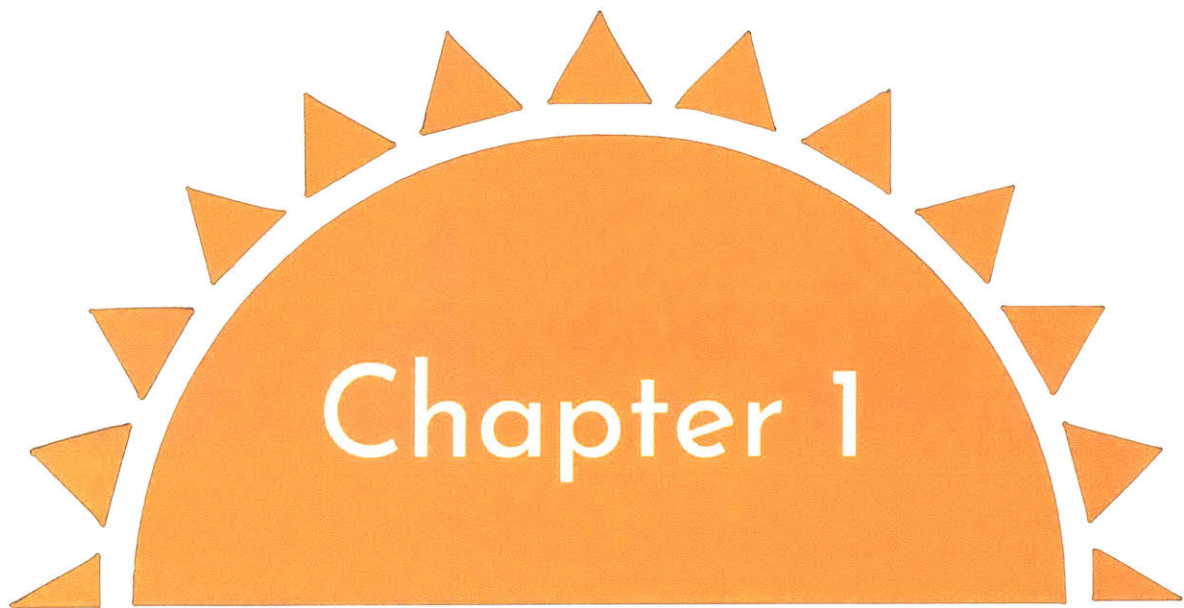
Surveying (Chapter 5) in which we identify and map the types of relational concepts that arise within the landscape.

Coming Home (Chapter 6) in which we consider takeaways and future work based on the project.

This document isn't meant to be an exhaustive treatise on the entire geography, let alone history and culture, of this new landscape, but simply a look at some of the unique hills and valleys of the terrain. Think of it as a walk in the woods. I hope you enjoy the trip.

Chapter 1





Preparing

From Bricolage to Bricoleur

We begin preparing for our journey by looking at some of the theoretical ideas that underlie Bricoleur. In particular, we look at the relationship between making and learning and its implications for designing creative tools. We ground this theory with an overview of Bricoleur and its creative potentials.

Bricolage

This project is about engaging people in *bricolage*, in the sense that Seymour Papert used the term, which he adapted from Claude Lévi-Strauss. Papert defined bricolage as “a style of organizing work that can be described as negotiational rather than planned in advance” [8]. Bricoleurs — i.e., those who engage in bricolage — approach their work by experimenting with materials, using the tools they have at hand, and course-correcting as necessary. Bricoleurs are often positioned in contrast to “planners,” which Papert implies in his definition of the word. That is, the planner does their thinking up front and simply executes the plan when they work. The bricoleur, on the other hand, is in continuous dialogue with their materials and also puts their materials in dialogue with each other. The bricoleur creates relations between materials and in doing so, builds mental relations in their head among ideas and knowledge. In this sense, creating with materials is the thinking process.

Bricolage is in fact just one word for a working style that goes by a variety of other names. When makers talk about “tinkering,” they are talking about bricolage. Artists talk about engaging with the “materiality” of their media. John Dunnigan and his students developed a term at RISD — “thingking” — which acknowledges both the cognitive and the concrete aspect of this way of working, as well as the relationship between them [16].

It is exactly this relationship between making and thinking that lies at the root of Papert's theory of *constructionism* which builds off of Piaget's earlier epistemological theory of *constructivism*. In short, Piaget's theory says that knowledge is not "acquired" by a learner (whether through transmission of information from another person, a book, or otherwise), but rather is always actively constructed in the mind of the learner, by the learner. When a learner encounters new knowledge, that knowledge must be integrated into the learner's existing mental structures. If that knowledge fits well with the existing mental structures, the knowledge can be assimilated directly. On the other hand, when new knowledge comes in conflict with the learner's current mental model, the model must be altered to accommodate the new knowledge to maintain a coherent structure [9]. In either case, the knowledge construction process is relational — putting new knowledge in relation to existing knowledge.

Papert's constructionism took Piaget's theory further by asserting that constructing concrete artifacts in the world is one of the best ways to construct knowledge. It is for this reason that he suggested that, "in the most fundamental sense, we, as learners, are all bricoleurs" [7]. That is, putting materials in dialogue with one another is a rich way to construct knowledge. Moreover, Papert posited that the computer — due to its programmability — is a tool uniquely suited to bricolage, especially for children. When people are engaged in creative programming activities, they are able to access "powerful ideas," to "think about thinking," and to reflect upon their own learning processes [7].

But how do learners choose which connections to make among ideas, which relations to construct? Vea Vecchi, a career *atelierista* in the Reggio Emilia schools, notes that aesthetics offer an especially potent tool for learners to bring disparate ideas into relation with one another. Here, Vecchi refers to the "aesthetic dimension" of learning in which care and attention to quality and a dedication toward creating meaning allows one to make connections between the imaginative and the rational in one's work [17]. Through this process, learners encounter not just the already known, but the newly possible. This view of learning as an aesthetic process has much to offer to the idea of bricolage. As makers dialogue with materials and the world around them, they are, in fact, putting ideas in relation with each other according to their own aesthetic senses. In this way, aesthetics provides a particular avenue for understanding learning through the process of putting materials together in new and unique ways — that is, through a bricolage style of making.

Given that learners are fundamentally bricoleurs, and that aesthetics can be a powerful mechanism for learning by enabling learners to connect disparate concepts in a bricolage style, it is important to develop rich aesthetic bricolage experiences for those learners to engage in. This thesis focuses on a particular context for bricolage which consists of a tool — appropriately called "Bricoleur" — along with materials created to support the tool's use. The following chapters in this thesis describe the design and development of Bricoleur, along with analysis of makers' experiences as they create with it. As a starting point, we give an overview of the tool itself to provide a concrete anchor to the rest of the work.

Bricoleur: An Aesthetic Context for Bricolage

Bricoleur is a mobile application that allows makers to create projects by capturing video and sound clips and programming those captured assets through a blocks-based programming interface. This tool combines video and audio assets with programming and mobility to open new creative possibilities for makers. Rather than capturing traditional rectangular video assets, Bricoleur requires the maker to hand draw a video mask that defines the area of pixels to be captured, enabling them to quickly create assets with complex and organic shapes. The maker can capture multiple video assets which then appear in a common canvas, ready to be programmed. By design, video assets do not contain any audio information. That is, a “video” in Bricoleur consists only of a sequence of images (unlike traditional video). Audio assets are captured separately and programmed independently of video assets. Programming entails snapping together visual blocks of code into “scripts” which define behaviors for each asset (e.g., to play back a section of video or audio, or to move a video asset across the canvas). The visual and sound output is rendered in realtime which eliminates the need for an “editing” mode and a “playback” mode. Edits to the assets, composition, and code are reflected immediately in the canvas and sound output.

By capturing and programming a collection of assets, makers can create a wide variety of rich interactive projects including dynamic artworks, stories, and musical instruments, to name a few. Examples of these types of projects are seen in the Vignettes section at the beginning of this thesis. Figure 1 shows the editor for a complete project (the motion landscape from Vignette One) along with a diagram of the basic parts of the interface.

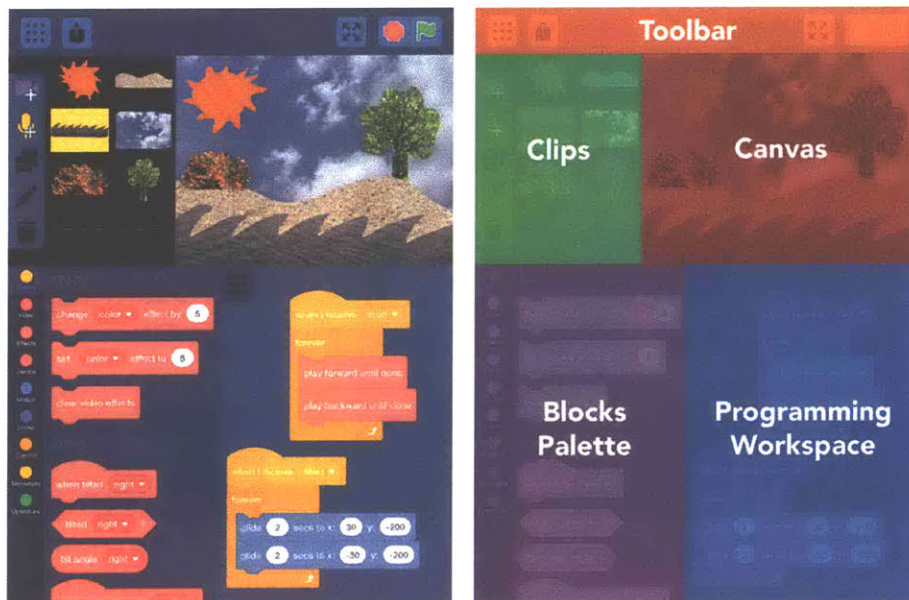


Figure 1 – (Left) The editor for a complete project. (Right) Diagram of the different parts of the editing interface

Making a Project

Makers get started by creating a new, empty project in Bricoleur. The interface guides them to begin by capturing either a video or audio asset (Figure 2).

Video

When capturing a video asset, the maker is prompted to draw the shape of the video they want to capture. Using their finger, the maker defines the masked area for where pixels will be captured via the device camera (Figure 3). The maker can also choose to use either the front-facing camera (e.g., to capture their own image) or the rear-facing camera. While recording, the pixels outside the masked area are overlaid with grey to clearly indicate the pixels that are being captured (Figure 4). Pixels outside the masked area are still visible through the semi-transparent grey overlay so that the maker can see the surrounding image just outside the video shape, which can provide helpful context when capturing. Clip recording is limited to 10 seconds (i.e., 300 total frames at 30 frames per second).

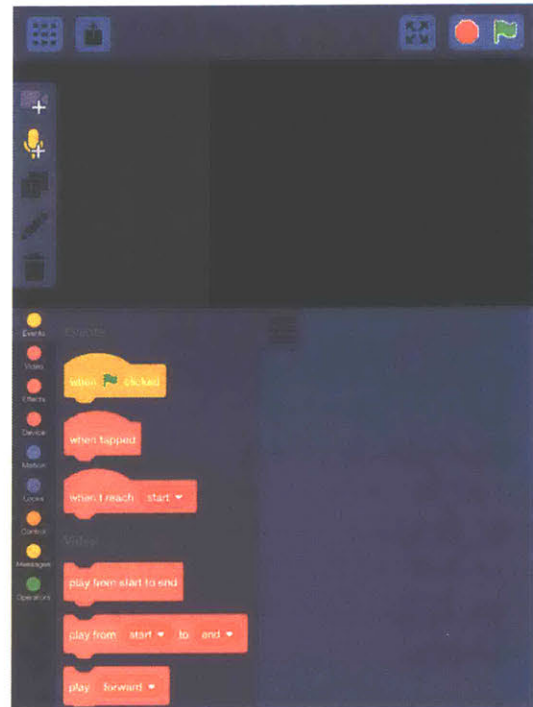


Figure 2 – A blank Bricoleur project.

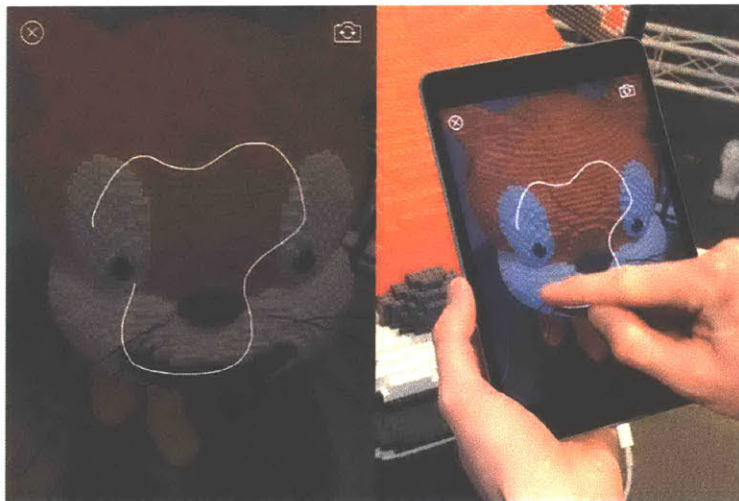


Figure 3 – Drawing a video mask.

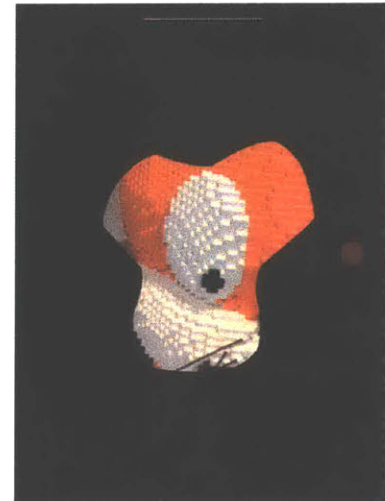


Figure 4 – Recording a masked video clip.

After recording a video clip, the maker is presented with a simple editing interface where they can playback the captured clip for review, trim the clip to a smaller subset of frames, as well as add markers to indicate specific frames of interest, which can later

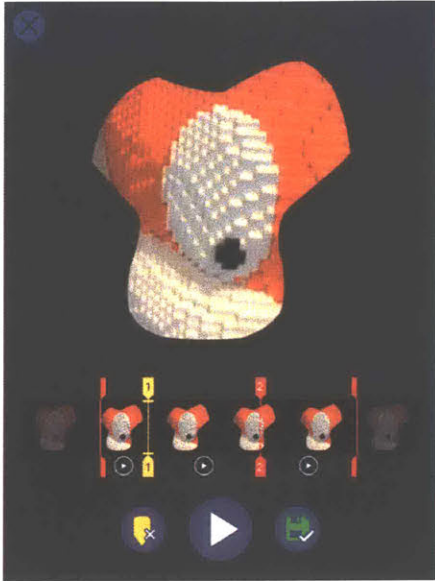


Figure 5 – Trimming and adding markers to a video clip after capture.

be referenced in the programming blocks (Figure 5). From here, the maker can choose to discard the clip, rerecord the clip, or save the clip into the project.

When a clip is saved, the maker is brought back to the programming editor where the clip appears in the canvas. A blocks-based programming interface becomes visible on the lower portion of the interface where the maker can snap programming blocks together to create scripts which define behaviors for the video clip (Figure 6). The palette of programming blocks contains blocks to perform video-specific behaviors (e.g., play from marker [A] to marker [B], or jump to frame [X]), blocks to control motion and state (e.g., turn [D] degrees, or change size by [P]%), as well as programming flow blocks to control the execution path (e.g., a forever loop or an event trigger when tapped) (Figure 7).

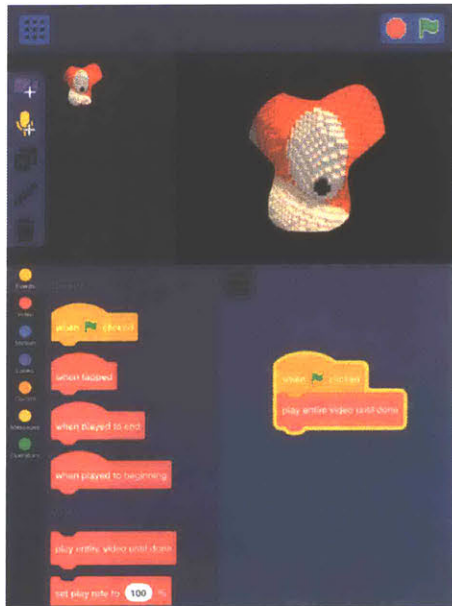


Figure 6 – Programming the video clip with blocks.

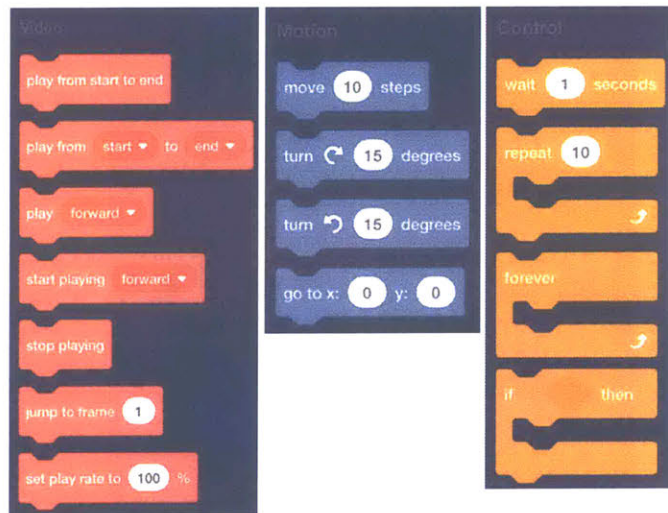


Figure 7 – Example programming blocks for video clips.

Makers can continue to capture and program additional video clips, each of which appears alongside the others in the rendering canvas. Each captured clip has its own set of scripts associated with it and can thus be programmed independently of the others. Behaviors can be coordinated between clips through a set of message passing programming blocks.

Audio

In addition to video, makers can capture audio assets with the device's on-board microphone. Similarly to capturing video, after an audio clip is captured the maker can playback, trim, and place markers at particular locations in the audio stream (Figure 8). Once the audio clip is saved, it appears alongside the video clips in the asset selector in the editor interface. Naturally, it has no rendered representation in the canvas, but like a video clip, it has its own set of programming blocks associated with it and is programmed independently of the other assets. The audio blocks palette contains a set of audio-specific programming blocks, analogous to the video-specific blocks (e.g., `play from start to end` or `set play rate to [P]%`) (Figure 9).

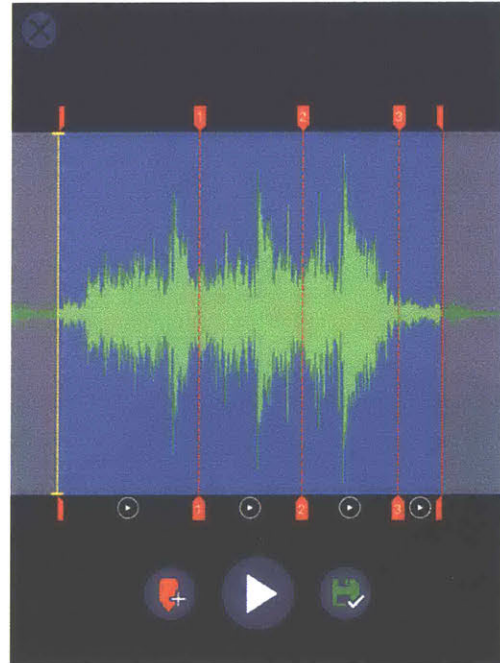


Figure 8 – Trimming and adding markers after capturing audio.

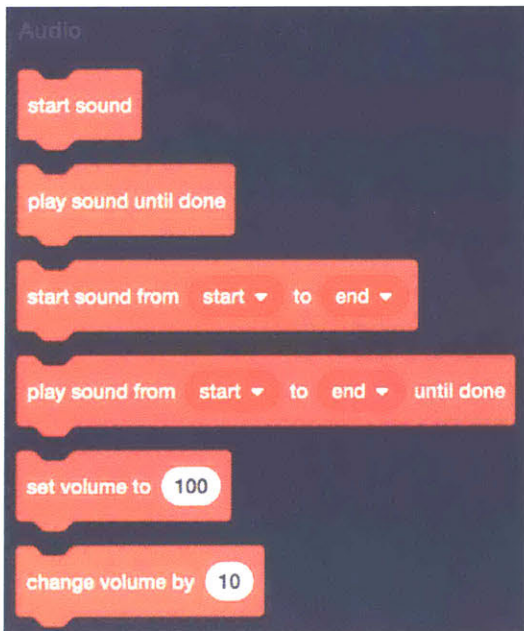


Figure 9 – Example blocks for audio assets.

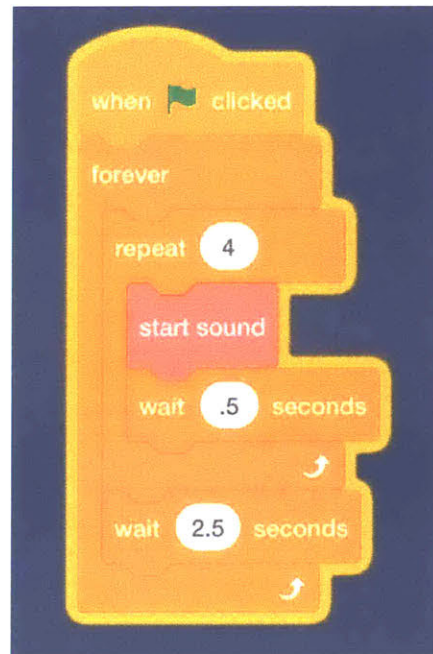


Figure 10 – An audio script that loops and layers a sound.

Bricoleur takes a slightly unusual model to sound playback. When a programming block triggers playback of an audio clip, the tool essentially adds a new copy of that sound to the currently outputting audio stream. Which is to say, if a script contains two blocks that cause audio to start playing, then each will start a new copy of the sound playing and the

audio will mix together. This allows for interesting potentials for audio programming, such as programming a custom repeat/echo effect by layering sound, or even creating harmony between musical sounds (Figure 10). Since a video clip can only ever render a single one of its frame at a time, there is no current analog for this type of programming for video assets.

Blocks for Interactivity

In addition to the blocks for audio, video, motion, etc., Bricoleur presents a set of blocks for interacting with the hardware of the tablet device. At the simplest level, a maker can build a script that responds to a user tapping on a video asset. Bricoleur also has blocks to utilize sensor hardware so that the maker can make use of the tilt angle of the device or the compass direction in their programming scripts.

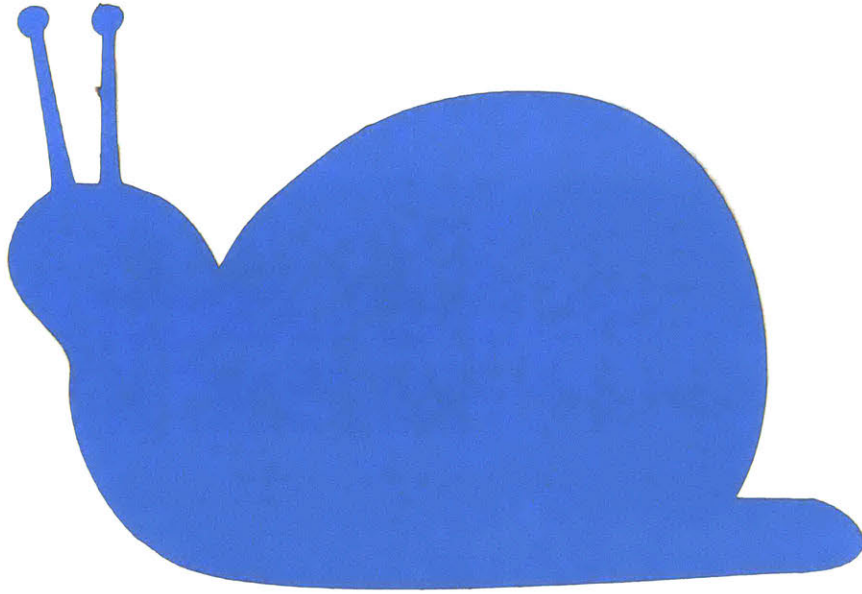
Each of these hardware interfaces have corresponding “event” trigger blocks that enable makers to respond to particular actions (e.g., when tapped, when tilted left, or when pointed toward north) (Figure 11). The video blocks also provide certain event blocks to respond to moments in the video stream (when I reach marker [A]).

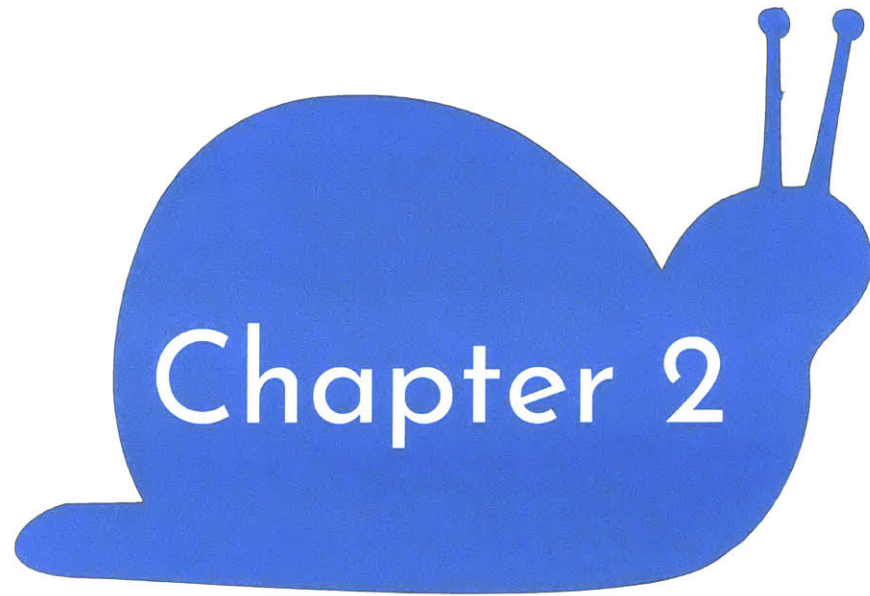
These hardware and event blocks allow for whole new classes of projects to be created. Rather than simply making a “video” that plays from start to finish, makers can create projects that respond to a user’s actions directly. Thus projects can become truly interactive.



Figure 11 – Blocks for interacting with tablet hardware sensors.

Chapter 2

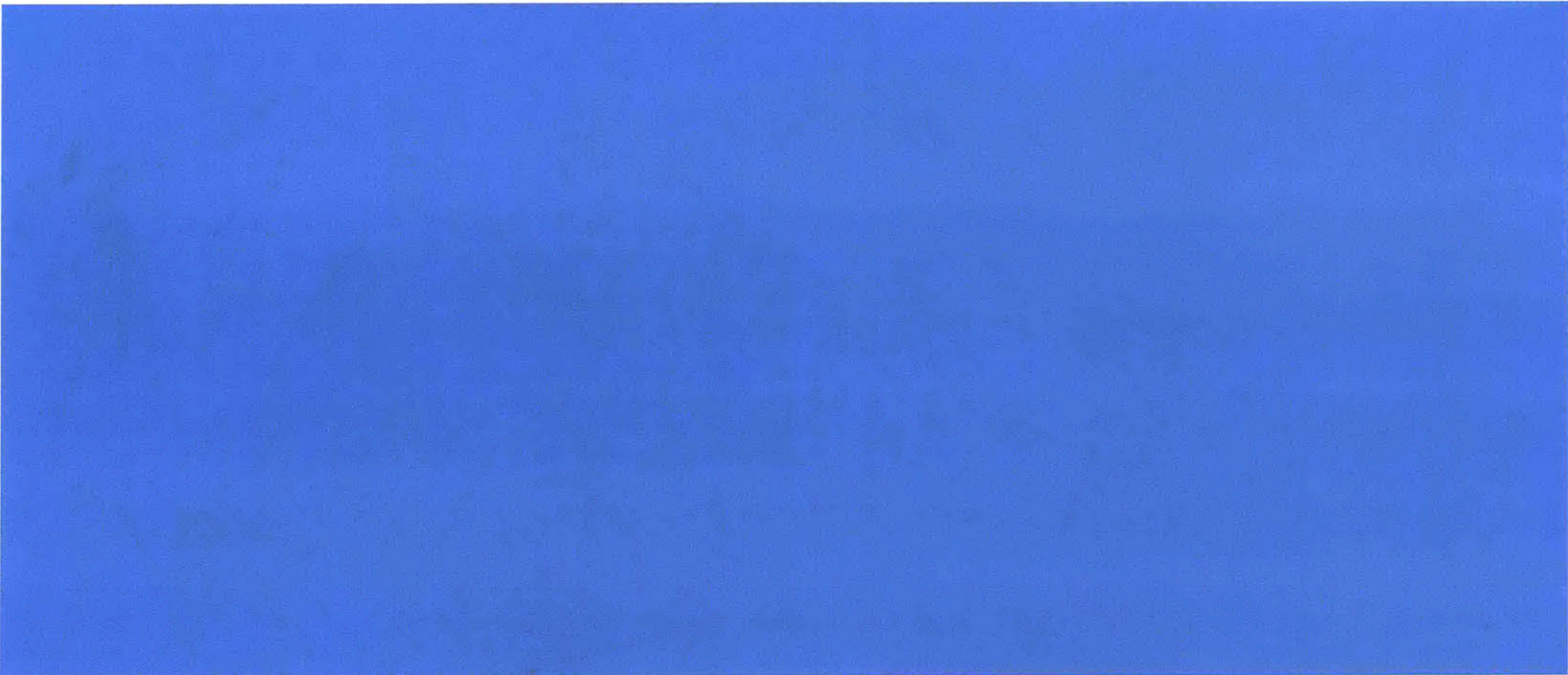




Packing Up

Cultural Influences and Inspirations

At this point in the journey, we turn to look at a variety of tools and practices that have influenced and inspired the design of Bricoleur. We will take these inspirations with us on our journey as we move into the design phase.



New creative practices often develop hand-in-hand with the emergence of new creative tools. The design of Bricoleur draws from two trajectories of creative tools in particular: film/video cameras and creative programming platforms. Each of these histories provides references and inspirations for designing a new creative tool at the nexus of these two streams, and particularly with young makers in mind. Moreover, we focus on tools that might be described as “convivial” in the sense that Illich uses the word. That is, we draw inspiration from tools “which give each person who uses them the greatest opportunity to enrich the environment with the fruits of his or her vision” [4]. These tools are ones that engender new generations of makers with the power to create new realities.

Film and Video

It is undeniable that the advent of the camera (both the still camera, and later motion) had an enormous impact on the way that culture and cultural artifacts are created and consumed. Rather than focusing on the entire history of film and video cameras, we highlight just three specific examples that shaped the way that individual makers (especially on a low budget) could create their own moving image works. Namely, these are the Bolex 16mm film camera, the Sony Portapak family of video cameras, and the Fisher-Price PXL-2000 (“Pixelvision”) camera. Each of these examples has not only empowered a new population of makers to create work, but has also inspired new creative practices to emerge as a result. For this reason, they served as strong inspirations for the design and development of Bricoleur.

Bolex

The Bolex 16mm film camera was first introduced in the early 1940s. This camera was handheld, able to be operated by a single person, and had adjustments for experimenting with film speed and focal length. Each of these features points to an important aspect that allowed a new type of cinema to arise [2]. The handheld, single person operation of the camera made it such that individual makers could make film without the needs of a larger crew. Furthermore, the portability allowed makers to take the camera into their environment and capture footage of the world around them. Adjustments for film speed and focal length allowed for particular effects to be created on film, which served to inspire a new class of experimental work. The film speed adjustment, specifically, allowed makers to create works that experimented with time in unique ways.

Quickly, the Bolex became a staple tool for experimental filmmakers and a new culture of making sprang up around this independent cinema practice [2]. Makers were able to produce work more quickly and perhaps most importantly, cheaply. With the lower cost of production — both materials-wise and crew-wise — makers had more leeway to experiment with work that wouldn't necessarily be commercially viable. It is no accident that the era produced some of the earliest experimental cinema classics. Maya Deren's *Meshes of the Afternoon* is a particularly well-known example of independent experimental cinema shot on a Bolex 16mm camera (Figure 12).



Figure 12 – Still from Maya Deren's *Meshes of the Afternoon*, shot on a 16mm Bolex camera. (credit: public domain)

Portapak

Video — film's low-fi electronic cousin — is at first glance, superficially similar to film. Both are media for capturing and playing back moving images. Nonetheless, film and video have been culturally connected to very different practices and institutions since the inception of each. Film grew naturally out of photography practice. Once a camera could take enough photographs per second, the images could be stitched together to produce a motion picture by projecting each in sequence. Video, on the other hand, was deeply connected with the production of television from the start. It was a mass medium and looked at as a mechanism for broadcasting information from huge corporations to the wider citizenry.

However, this mass media role of video began to shift at the introduction of the Sony Portapak in the 1960s, the first portable video camera system (Figure 13). In many ways, what the Bolex did for film, the Portapak did for video. Like the Bolex, this camera was portable and able to be operated by a single person (or team of two) and brought video

into the hands of individual makers — a medium previously only accessible to major television broadcast corporations. And like the Bolex, a new practice of making emerged around the tool: video art [2].

From the beginning, video art practice was connected to a variety of other media and art forms. Many of the earliest video artists worked in other media including dance, music, performance, and visual arts, to name a few. Joan Jonas' video work grew out of her performance art



Figure 13 – A Sony AV-3400 Portapak camera system. (Photo by Wikipedia user Mwf95, 2005. CC BY-SA 4.0)



Figure 14 – Still from *Switch! Monitor! Drift!* by Steina Vasulka, 1976.

background while Nam June Paik used video as a sculptural material throughout his career [2]. Other makers took the affordances of the Portapak as an opportunity to experiment with the medium of video itself. Steina Vasulka, in particular, created work around the apparatus of the camera itself, experimenting with mirrors, motion, and more to create unique images (Figure 14). Still others utilized the portable camera as a tool for documentary and for activism, using the camera to capture political and social movements on the streets of cities [1]. In each case, the portability and accessibility of the Portapak tools of production helped create new modes of creative expression.

PXL-2000

While the Bolex and Portapak enabled new expressive possibilities, both of these technologies were designed for and marketed toward adult makers. The Fisher-Price PXL-2000, on the other hand, was a video camera expressly designed for children (Figure 15). The camera recorded in high-contrast black-and-white onto standard audio cassette tapes in a small letter-boxed format in the middle of the screen. The aesthetic — dubbed “pixelvision” — became highly sought after in later years as video artists began to use this “toy” camera for their own work, long after the PXL-2000 stopped being produced due to lack of sales [6].

Despite few units being sold, a handful of young makers got their hands on the camera

and had the chance to perform their own video experiments and document their lives. The best known youth pixelvision maker is undoubtedly Sadie Benning who used her pixelvision camera to create a series of semi-fictitious diary-esque video works in the confines of her home bedroom. These early experiments allowed her to explore themes of relationships, anxiety, and queerness to name a few [5]. Benning’s work highlights the power that creative tools can give to children: to express themselves and explore the world they inhabit.

Each of these cameras allowed for new practices and forms of expression to emerge. Specifically, the portability, accessibility, and the ability to experiment with each camera afforded a new population of makers to create expressive media. Bricoleur was designed with these aspects in mind in order to try to engender the kind of expressiveness and experimentalism that each of these cameras brought to film and video.



Figure 15 – The Fisher-Price PXL-2000 “pixelvision” camera. (Photo by Joe Lillibridge, 2008. CC BY-SA 2.0)

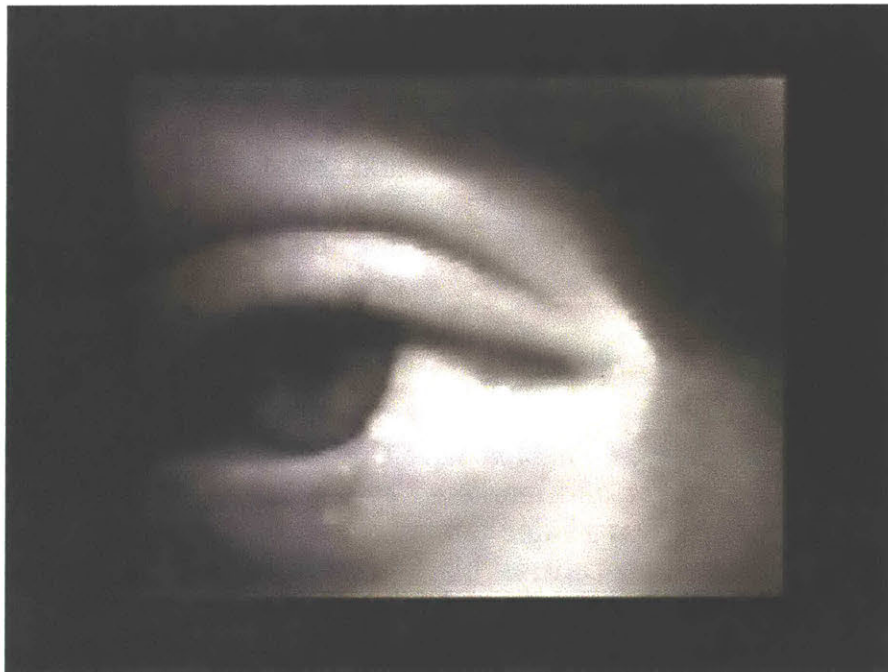


Figure 16 – Still from *Me & Rubyfruit* by Sadie Benning (age 16), shot on a PXL-2000, 1989.

Creative Programming

In addition to cameras, Bricoleur draws from the history of programming languages to inform its design. Programming languages have a wide variety of aims and uses and as a result, are designed with different purposes in mind. Some languages are geared toward scientific research, others for application development, and still others for automation. Bricoleur draws particularly from a lineage of languages designed for children which emphasize creativity and experimentation as part of the programming process. Here, we look at these languages in addition to some languages aimed specifically at media programming.

LOGO

LOGO, designed by a team led by Seymour Papert, was the first language developed specifically for children and came directly out of Papert's thinking and research [7]. The language was designed as a "microworld" in which children would have access to powerful ideas in mathematics by programming the behavior of a physical or digital "turtle" through text-based commands and procedures. By writing and executing procedures for the turtle, children could think about their own procedural (or non-procedural) thought processes, thus giving them a way to "think about thinking." Furthermore, the turtle was chosen intentionally so that children could draw a parallel between the movement of the turtle and the movement of their own bodies — a process of identification that Papert referred to as "body syntonicity." Moreover, Papert was also insistent on the role of the personal and affective nature of creating. In light of this, LOGO was designed so that children could be expressive in a variety of different ways and bring their personal interests into their projects.

Bricoleur draws from LOGO in several ways. First, the design of the programming blocks aims to give makers access to thinking about the way the underlying media is represented and how it operates in time (e.g, by referring to a particular frame of a video clip). Second, Bricoleur aims to bring the body of the maker into the creation of the work. As one example, the maker uses their hand to draw the shape of each video. This act is also tied to the third way that LOGO has influenced Bricoleur in that it allows makers to bring their own personal ideas and interests into their projects.

Scratch

Scratch built upon the ideas of LOGO and offers a more recent creative programming environment for children. Scratch is a web based programming environment and also an online community where young makers can share their projects with each other. Rather than text, Scratch employs a drag-and-drop blocks-based programming paradigm in which makers can create scripts by snapping together programming blocks, much in the spirit of snapping together LEGO bricks. This programming system not only removes the burden of syntax errors for new programmers (i.e., only blocks that can actually go together will snap together), but also allows for a more "tinkerable" style of coding — a topic that will be investigated at length in the next chapter. For example, a maker can take a block out of a script and simply set it aside elsewhere in the workspace rather than

deleting the block all together [13].

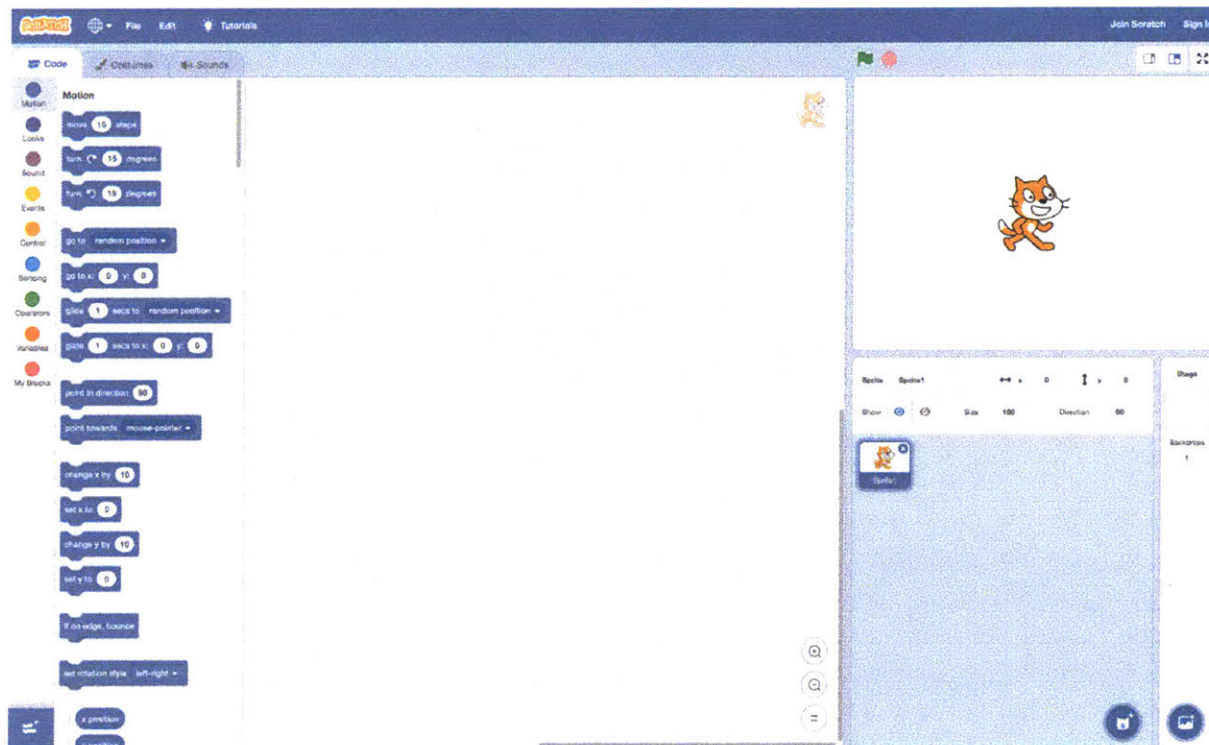


Figure 17 – The Scratch project editor.

Scratch projects, by nature, are media oriented. Scratch programmers create projects by building scripts of code that belongs to “sprites” – two-dimensional visual assets which are then rendered onto an on-screen canvas. Sprites not only have code associated with them, but can also have sounds and “costumes” (different on-screen representations of the same sprite) which are also controlled through programming blocks.

Beyond the technical aspects, though, Scratch is designed to enable children with all kinds of interests and backgrounds to be creatively expressive. One way in which is this talked about is the framework of “low floor, high ceiling, and wide walls” [11]. In the context of Scratch, this means that makers should be able to get started making a project quickly (low floor), they should be able to grow their projects in scope and complexity over time (high ceiling), and that makers should be able to bring a wide variety of interests and create a highly diverse set of projects with the tool (wide walls).

Bricoleur draws heavily on Scratch’s design and programming paradigm. In some cases, the influence is highly concrete. For example, the blocks-based programming interface from Scratch is integrated directly into Bricoleur and makers enact a similar sort of project building strategy in that each video or audio asset is the equivalent of a sprite in Scratch. Moreover, though, the philosophy and design frameworks underlying Scratch have provided considerable guidance in the design and development of Bricoleur. The low floor, high ceiling, wide walls framework, in particular has helped, as well as Scratch’s goal of being as tinkerable as possible.

ScratchJr

Scratch is designed for makers 8 years old and older. One specific reason for the minimum age of 8 is that using Scratch requires the ability to read, since the programming blocks are labeled with text explaining their functions. As a result, ScratchJr was developed and designed for children ages 5 to 7 to make creative media projects with icon-based programming blocks. ScratchJr runs solely on tablet hardware which has provided some important design ideas for Bricoleur. In particular, ScratchJr makers can utilize the on-board camera to insert images of themselves into sprites. By selecting one of the shapes that comprises a vector-based sprite, the maker can capture an image to fill the pixels only within that area. This provided the inspiration for the hand-drawn video masking in Bricoleur as well as the idea of utilizing the on-board hardware to create projects.

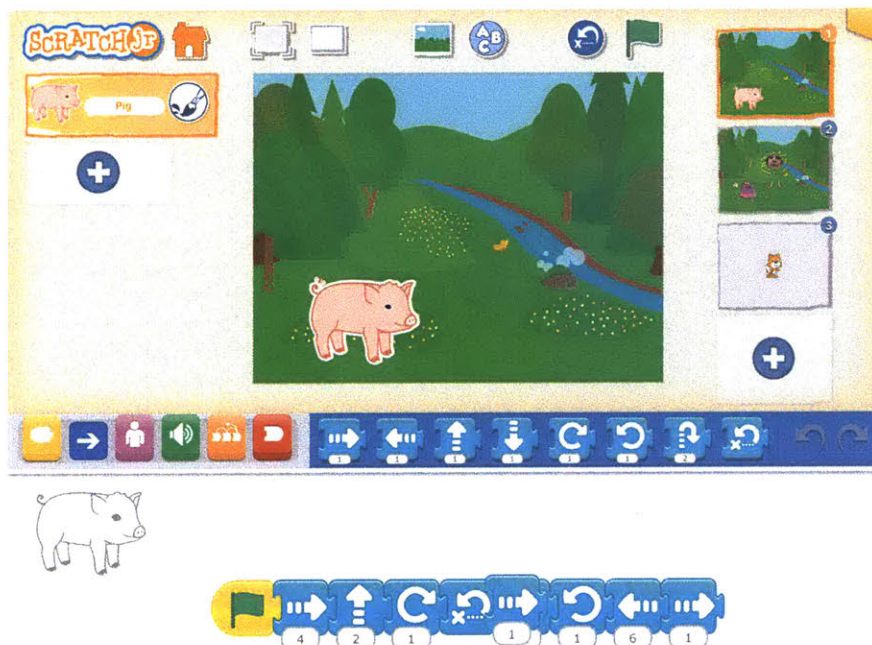


Figure 18 – A ScratchJr project.

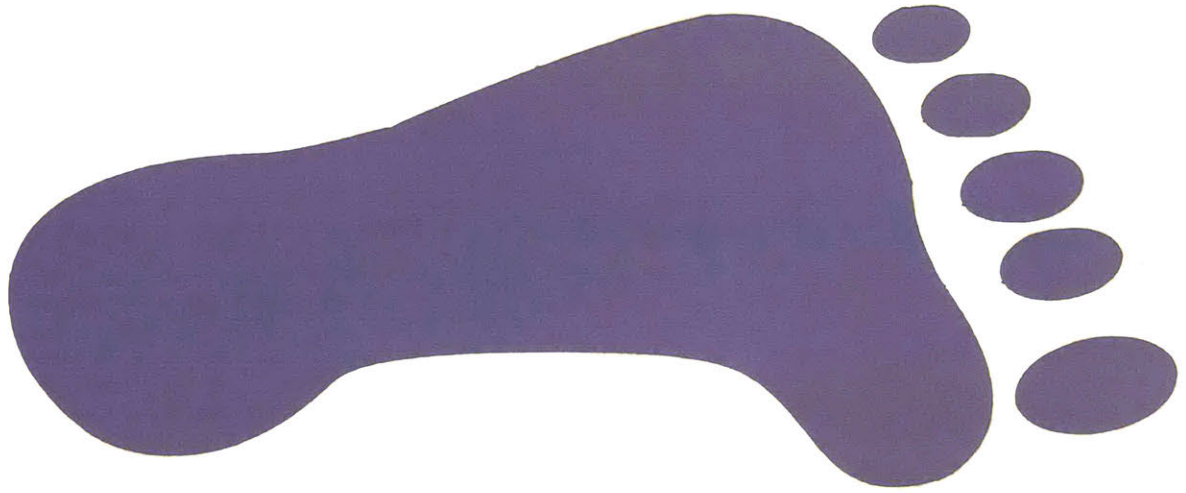
Max

The previous three examples are all languages designed for children and have certainly been the most influential on Bricoleur's design and philosophy. However, there are a wide variety of media programming languages not targeted at children which have explored the space of programmable media as well. Max, in particular, helped inform some of the early ideas of Bricoleur. Max (along with its extensions MSP and Jitter) is a visual programming tool for creating interactive projects with data, audio, and video [10]. Max emphasizes realtime computation and also sets up a paradigm in which audio and video are treated as computational objects. While the design and usage of Max is significantly different than that of Bricoleur, these aspects of realtime computation and video/audio as computational objects provided some initial strategies for creating early prototypes of a blocks-based video and audio programming tool.

A Note on Video and Audio Editing Software

It is perhaps unusual to not mention any traditional video or audio editing tools when describing the design of a new creative video/audio tool. While these types of tools are very powerful and have been carefully designed to do what they do, ultimately their goal is significantly different from that of Bricoleur and ended up not providing much design insight. Editing tools are designed to allow makers to create video and/or audio files as output. As we will see in subsequent chapters, the types of projects that makers create with Bricoleur are not video and audio artifacts, but rather interactive computational projects. Which is to say, many of the things created with Bricoleur simply could not be made with a traditional editing tool. For this reason, Bricoleur focused on the aforementioned camera systems and programming tools as inspirations.

Chapter 3

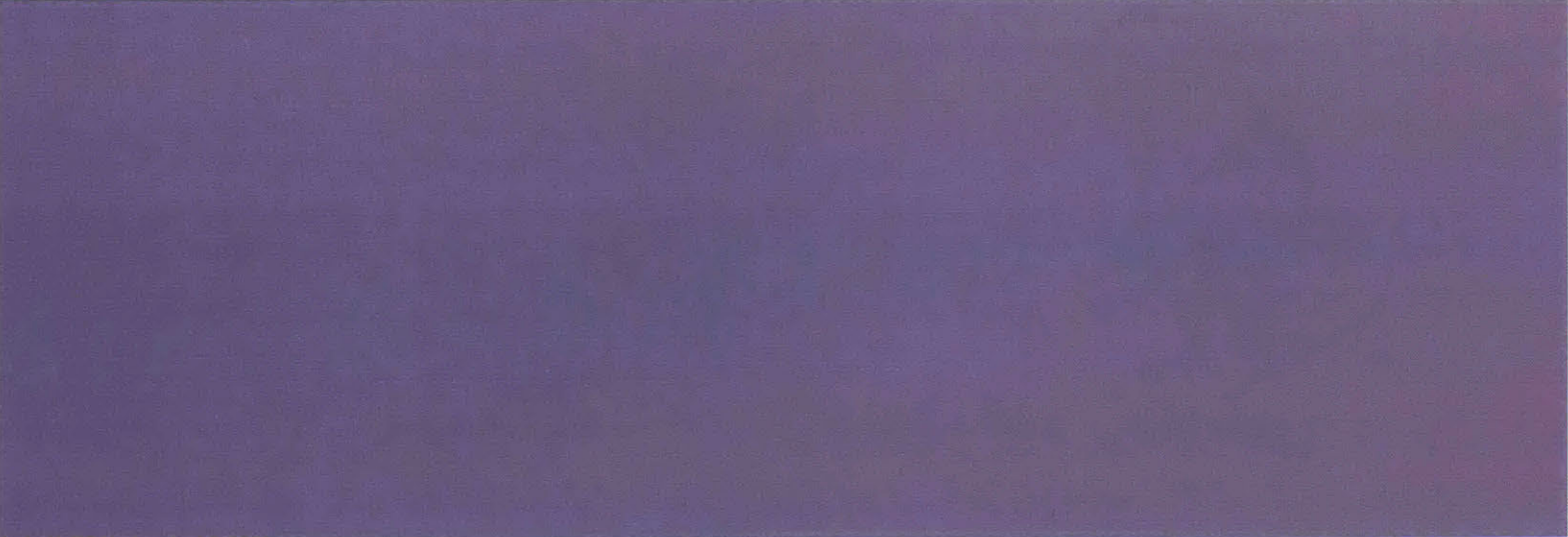




Heading Out

Design

With our preparations in order and our bags packed, we are now ready to embark on our journey to our new landscape. In this chapter, we describe the design iterations that will bring us there.



Like any design project, Bricoleur developed over the course of many iterative cycles. The initial design experiments were based mostly on intuition and taking cues from the designs of the tools in the previous chapter, especially Scratch. Once it became clear that the project could develop further, a design framework was adopted to guide subsequent design iterations toward making a tool that truly supports bricolage styles of making. In this chapter, we present the design framework and several iterations of Bricoleur from its initial conception into its current form.

Tinkerability

As noted in the introduction, the idea of bricolage as a way of working is largely analogous to that of “tinkering.” Given the aim of this project, we drew upon Resnick and Rosenbaum’s design criteria for creating tinkerable technologies and activities to inform the design of Bricoleur [12]. In particular, this framework provides three criteria for tinkerability: (1) immediate feedback, (2) fluid experimentation, and (3) open exploration. We contextualize each of these criteria to this specific design project.

The first criterion, immediate feedback, suggests that tinkerable tools should reflect any action a user makes as quickly as possible back to the user. By showing the results of an action in the shortest time possible, makers can more quickly experiment with ideas or materials in the bricolage style. Bricoleur builds off the strong work that Scratch has already done in this area. For example, in both Scratch and Bricoleur, code blocks execute immediately when they are tapped and running scripts update as soon as any change is made to their block stack.

Fluid experimentation, the second criterion, refers to the way in which the tool facilitates the iterative making process. One way to think about this is how makers are able (or not able) to dialogue back and forth with the tool. Can a maker develop a meaningful project over time, even if they don't begin with a specific goal in mind and/or shift directions multiple times? In light of this, fluid experimentation implies that it should be easy for makers to get started making a project in Bricoleur and furthermore, it suggests that projects should be able to naturally grow in complexity over time.

Finally, the open exploration criterion refers to whether or not the tool can support a wide variety of materials, subject matter, and personal interests. In thinking back to the Portapak camera, video already provides a step in the right direction given its ability to combine with other materials and practices (i.e., music, dance, performance, visual arts, etc.). Building off of this, the design of Bricoleur aimed at allowing a wide variety of project types to emerge. Makers can create more traditional linear video context, but they can also create musical instruments, performances, and other interactive types of projects.

We now look at the iterative development of Bricoleur over time and how this framework informed the design along the way.

Iteration 1 – Eisenstein

The genesis of Bricoleur started with an idle wondering: “What would the combination of video and creative computing produce?” Or, to post it as an open-ended design question, “What would a blocks-based programming paradigm afford to the medium of video, and what would the medium of video afford to a blocks-based programming paradigm?”

Eisenstein, the very first prototype of Bricoleur, was about as simple as the idea could get and still address this question. Built in less than a week, the app consisted of a square video canvas in the upper right corner of the interface and a blocks-based programming workspace that took up the remainder of the screen space. The maker could capture a collection of video assets and then use the blocks to program simple video behaviors such as playing a video forward or backward as well as standard Scratch behaviors like rotating or scaling the video within the canvas frame (Figure 19).

A few important design decisions were made even at this early stage of the project. Crucially, the maker was only able to capture images with the camera, i.e., without sound. The intention here was to highlight the unique aspects of the

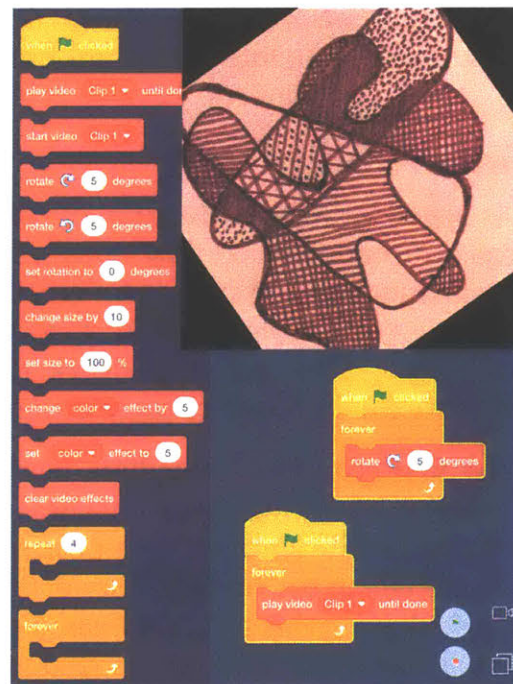


Figure 19 – The Eisenstein editor.

video medium and encourage the maker to explore those in and of themselves without needing to think about the relationship between images and sound. The focus on aspects of the medium also informed the way the video blocks were designed. The blocks allowed simple behaviors like playing a video from start to end, but also allowed the maker to interact with and program using individual frames of the video, the playback rate, and image properties like color.

Importantly, while the maker could capture multiple videos, only a single video could be shown on the canvas at a time. This was partially a technical choice (rendering multiple assets is naturally more complex than rendering a single asset) but also partially a design choice. The thought was that the most common usage would be to sequence clips together,* much like a traditional video editor, so the interface and blocks were designed to facilitate this type of making experience. However, after a quick initial test of the prototype, it was clear that this assumption was wrong. Makers were far more interested in playing with the blocks that enabled different kinds of outcomes that were difficult to produce in traditional video editing software. For example, capturing a video while rotating the camera, then using the rotation blocks to “cancel out” the rotation within the frame.

Moreover, it became clear that makers were not producing videos as such. The programming blocks made it quick and easy to create media that did not play from start to finish in the way a traditional video does. Even with the simple introduction of the `forever` loop block, makers could create work that was essentially infinite in run length — an impossibility in traditional editing software. Each project was not “a video made by programming” but an entirely new type of digital work that expanded beyond the linearity constraint of video.

The outcomes from this quick initial testing proved to be interesting enough to warrant a second prototype. The programming blocks seemed to break the maker out of the time constraints of traditional video, which led to the question of how to break the maker out of the space constraints as well. Two different, but related space constraints were considered. First, and probably most obvious, was the aforementioned aspect of having a single image on the screen at once. Second was the fact that video, by its nature, is captured and edited in a rectangular format. This constraint is ultimately related to the first since in putting multiple images on the screen, the question arises as to what their boundary should look like. Certainly, video editing software can composite multiple video streams together in a variety of configurations and shapes, but this is a complex operation not often accessible to beginners and also typically requires considerable rendering time to generate such an image. These ideas formed the basis of the second prototype.

Iteration 2 – Steina

The major distinction between Eisenstein and the second prototype, Steina, was in addressing these space constraints of traditional video. ScratchJr turned out to be a source of inspiration, along with input from some colleagues. When editing a sprite image in

* Hence the name “Eisenstein” after the Soviet director and his theory of montage.

ScratchJr, the maker can select a closed shape (circle, square, hand-drawn, etc.) and then use the camera on the device to capture an image and fill the shape. A common use of this is to put one's face in place of a sprite's face, thus enabling a maker to put themselves in the scene of their ScratchJr project (Figure 20). This led to the idea that when capturing a video, rather than capturing all the pixels within the frame's bounds, the maker could use their finger to trace the shape of the image they want to create and then only capture the pixels within that hand-drawn shape.

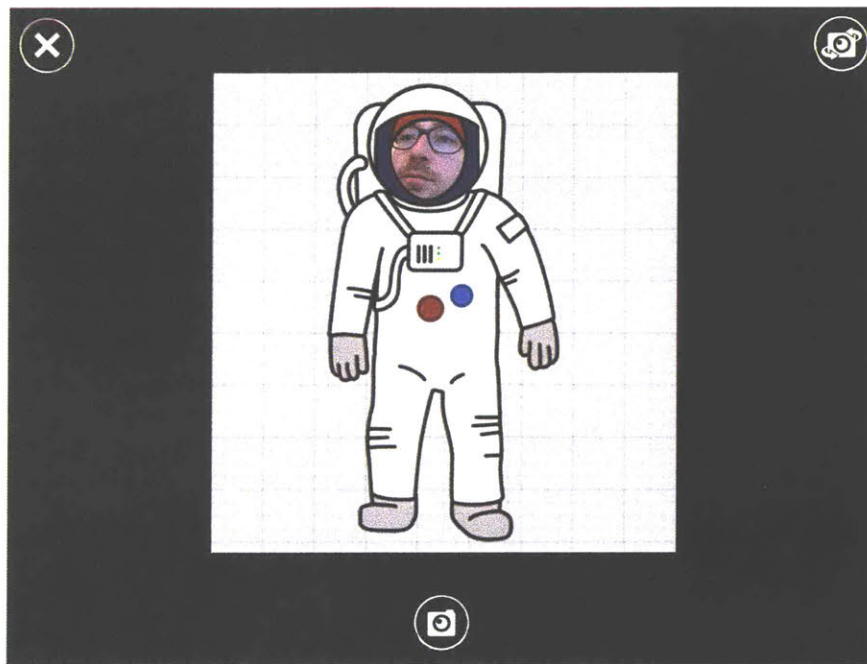


Figure 20 – Capturing an image of a face inside a sprite in ScratchJr.

This idea not only addressed the rectangular video asset dilemma, but since each video asset was only a subset of its original pixels, by nature it would not take up the whole canvas space. This naturally led to being able to layer and program multiple video assets within the same canvas space. Taking a cue from Scratch, each captured video could now exist as its own “sprite” and thus have its own programming blocks associated with it, in contrast with the first prototype.

With the addition of this draw-capture-program workflow, a whole new variety of projects could be made very quickly. The first project consisted of going outside and creating a “motion landscape” in about ten minutes. This project comprised six separate video assets assembled together in realtime to depict a beach scene and can be seen in Vignette One at the beginning of this thesis. Between the programmability, the hand-drawn asset capture, and realtime compositing, it seemed that makers would be able to experiment with breaking the constraints of both time and space of traditional video.

Through a set of in-house workshops at Lifelong Kindergarten, a few threads started to emerge about what makers could make, and how they would make it. One aspect, already apparent from the first prototype but reinforced with the second, was the new

space of possibilities afforded by being able to take the device into the world to capture footage of one's surroundings and then program those clips in situ. Furthermore, the ability to isolate and capture only of a part of the video frame led to a series of projects about disassembly and reassembly of entities from the physical world. It became clear that a common first project to make was to capture different body parts (especially parts of the face) of either one person, or several people, and then reassemble them into a kind of body-mashup (Figure 21). A related aspect that quickly arose was the collaborative potential of the tool. As makers teamed up to create projects together, the mobile nature of the device made it such that the tool could quickly be passed back and forth, thus enabling multiple people to work on a single project with ease. In comparison to standard code sharing practices, this allowed a much, much tighter iteration loop between different makers. Projects also continued to explore the ideas from the first prototype (e.g., matching or canceling video motion with motion block programming) (Figure 21).

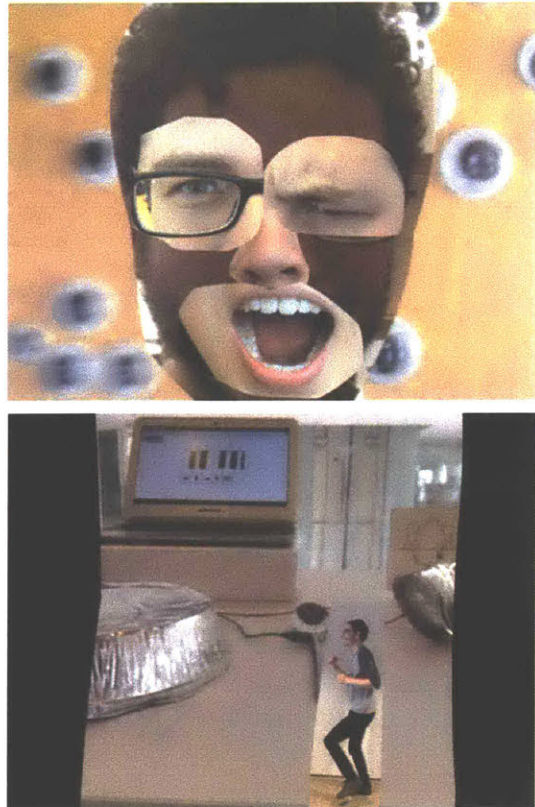


Figure 21 – Examples of projects made with Steina. (Top) A face mashup. (Bottom) A side-scroller game.

These workshops also pointed toward next steps for another prototype. One feature which seemed to be glaringly absent was audio. As makers continued to make projects, the desire to capture sound as well as image arose over and over again. Secondly, while the programming blocks helped to break projects out of linear form, the fact remained that most projects were started by the maker pressing start, and then simply playing the project to completion according to the programming. This pointed toward the idea of introducing interactivity into projects. Blocks could be added to the palette that would enable video behaviors to trigger based on a whole variety of inputs.

Iteration 3 – Bricoleur

As previously mentioned, audio was intentionally left out of early prototypes of the tool to encourage makers to think about video and its unique properties as a medium without the added complexity of synced sound. The thought was that the ability to capture sound simultaneously would take away from the experience of playing with video on its own terms. But after much feedback on the second prototype, this was reconsidered and framed as: how might audio be added to the tool in a way that still respected the original intent to explore video as its own medium? To that end, the choice was made to allow audio capture, but always separate from video. In this way, makers would be able

to play with each medium independently and also have the opportunity to think about and construct relationships between the audio and video in their project. Similarly to video, each audio clip existed as the equivalent of a sprite in Scratch. That is, each audio clip had its own set of programming blocks associated with it. This is different than the Scratch model of a sound “belonging” to a sprite, as a way to further encourage makers to explore the many possible relationships between a particular video clip and audio clip.

With the introduction of audio came a new set of questions about common usages and blocks design. A simple usage is to play a specific section of an audio asset, from a particular starting point to a particular ending point. This can be used both as a way of trimming an audio clip, as well as a way to create an audio loop. With this in mind, an audio capture interface was created that allowed the maker to add markers at particular locations in the audio stream which could then be referenced in programming blocks (e.g., `play from marker [a] to marker [b]`).

As a first step in adding blocks to support interactivity, a single `when tapped` hat block was introduced to the blocks palette. This block would trigger a script when the maker touched the video inside the canvas.

Between these two new features, the space of project possibilities expanded further. Audio added the ability to tell stories with one’s voice, add atmospheric and ambient background noise to a project, sound effects, and more. The touch interaction further allowed makers to create projects that not only play, but are playable, which is to say, performable. As an example of a use of both audio and touch features, a maker could now create a playable musical instrument by triggering a sound when a particular video sprite is tapped. The “Xyloface” project — depicted in Vignette Three at the beginning of this thesis — became a go-to example for showing new makers the possibilities of the tool. The project consists of eight video assets, each depicting someone’s face. When a face is tapped, the video plays showing the person’s face opening and closing their mouth. Simultaneously, the sound of a particular xylophone note is played. In this way, a user can “play” melodies or even chords on the Xyloface by tapping on the faces in whatever order, timing, or combination they choose.

At this point, the tool was renamed Bricoleur to highlight the aspect of capturing assets (now both audio and video) from the world and combining them into unique works. With a rich set of example projects already built, this prototype laid the groundwork to continue developing a tool designed explicitly for tinkering, and in accordance with the tinkerability criteria.

Iteration n + 1

As development of Bricoleur continued, iteration cycles became smaller and quicker. Workshops and user testing — described at length in the next chapter — provided a lot of information which informed the next several cycles of development. Here we highlight some of the most important outcomes of these iterations.

Early workshops highlighted some usability challenges with the tool. Makers were

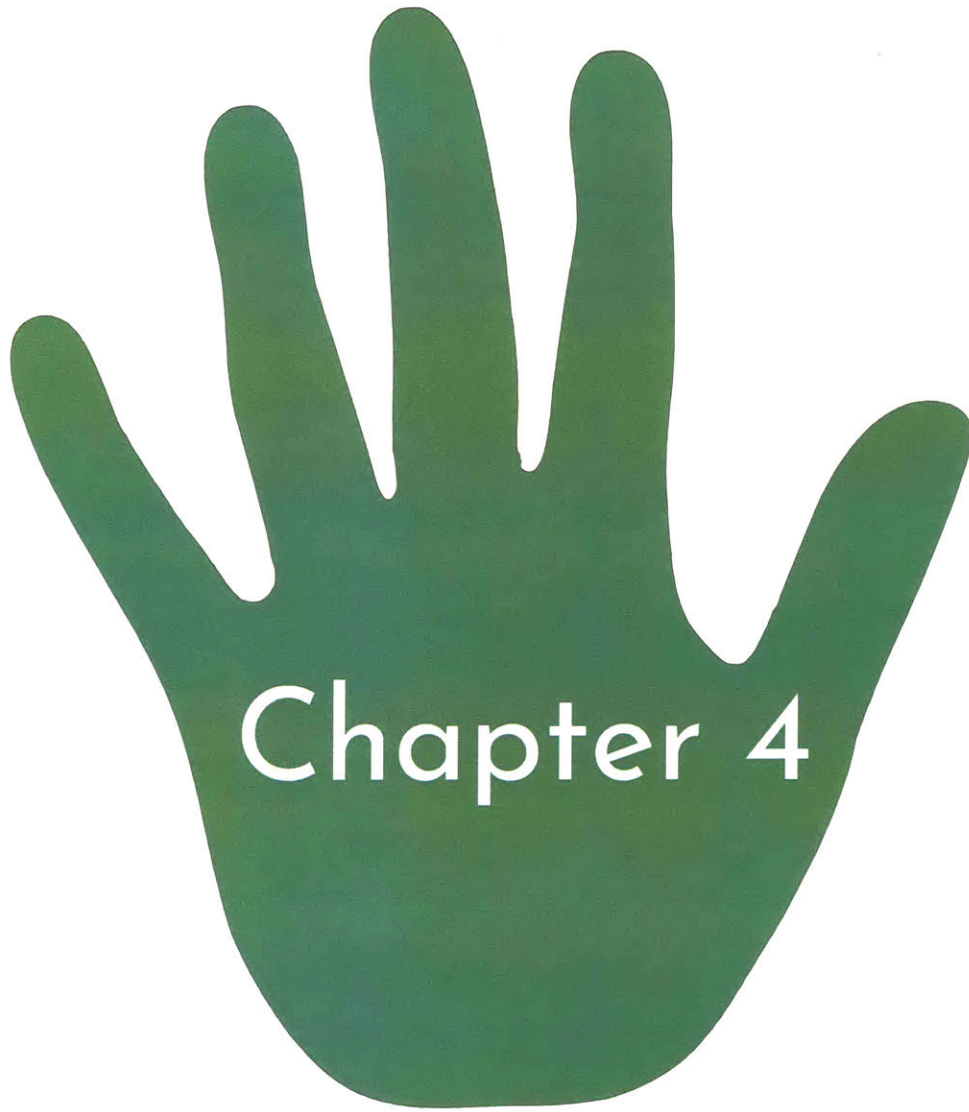
sometimes unsure how to get started and at other times were unable to perform actions that they felt were intuitive. Clearly, this was working against the design principles of both fluid experimentation and immediate feedback. As a result, one iteration focused entirely on the refining the usability of the interface. A few important changes emerged from that iteration and improved the tinkerability of the tool. First, visual signposts were added to help makers get started in making a project (for an example, see the arrows in the editing tool shown in Figure 2). Second, two-finger “pinch” gestures were implemented which allowed makers to scale, rotate, and translate video clips without needing to do so through the code blocks interface. Further workshops confirmed the importance of these changes toward fluid experimentation and immediate feedback.

After some workshops highlighted the creative affordances of the tap recognition blocks, another iteration aimed at prototyping new ways to build interactivity into projects. This iteration produced the tilt and compass blocks which opened up a wide variety of new creative possibilities for subsequent makers. This helped encourage open exploration in the workshops that came after their introduction.

Other iterations focused on adding the video and audio editing interfaces, the ability to send projects between devices as a simple sharing mechanism, an expansion of the audio blocks palette to include features like playing audio backward or at different playback rates, and the ability to duplicate and/or delete assets. These changes had a smaller impact compared to the first two examples, but each proved to be useful in making Bricoleur more usable and more tinkerable.

Chapter 4






Chapter 4

Exploring

Workshops

At last, we have arrived at our new landscape and are ready to explore the terrain. We look specifically at what and how makers create with Bricoleur.



Each design iteration described in the previous chapter not only consisted of design work, but also user testing in the form of workshops. Early in the design process, user testing was done mostly with adult-aged makers while the tool was rougher and less clearly defined. As Bricoleur became more refined over time, more workshops were done with youth (mostly teenaged).

A typical Bricoleur workshop ran for 1.5 to 2 hours. Prior to the workshop beginning, the environment would be set up with physical materials to support the making process. Physical materials were selected to spark ideas and provide points of entry into the making process. Since all projects entail capturing video and/or audio, materials were select-

ed that would have potentially interesting uses in those media. For example, a variety of noise makers (percussive, melodic, textural, etc.) were selected to generate a wide range of sounds. Specifically for video, materials were chosen that had interesting textures, colors, patterns, and/or optical properties (reflective materials, translucent materials, etc.). A few anthropomorphic materials were added to the mix as well to spark ideas about characters or stories. Craft materials like paper and markers, pipe cleaners, and play-doh were also brought to encourage participants to create objects in the physical world as well (which could then be captured into Bricoleur). These materials would be laid out on a table or other nearby surface and presented such that participants could easily access and use them in whatever way they see fit for their project (i.e., materials were not kept in packages). This process takes a cue from Reggio Emilia atelier contexts in which materials are carefully laid out so that they can be seen, considered, and used as easily as possible (Figure 22).



Figure 22 – Materials setup for a Bricoleur workshop.

In addition to physical materials, other supporting materials were brought to help spark project ideas and to assist in programming. A set of “idea spark” cards were developed for an early workshop and proved to be useful enough to bring to subsequent play tests. The idea spark cards are divided into two subsets: a set of project “structures” (e.g., “a poem”) and a set of project “strategies” (e.g., “for two or more people”). Participants could choose a structure and strategy at random in order to provide some ideas for getting started on a project (Figure 23). The cards were designed to provide “open constraints” – i.e., constraints that could be interpreted in multiple ways and allow for a variety of unique outcomes. A full list of the idea spark cards is given in Appendix B.

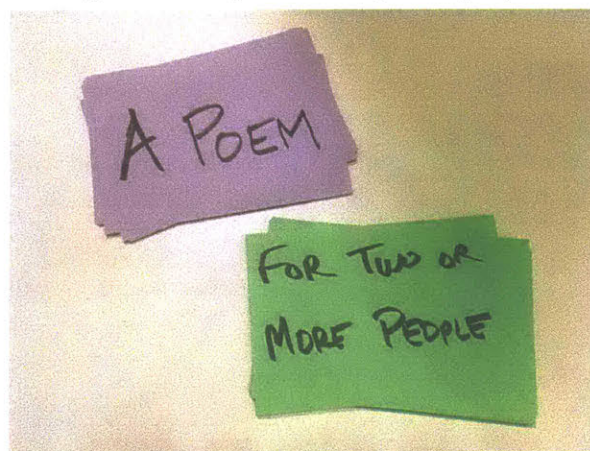


Figure 23 – Spark Card decks showing an example draw.

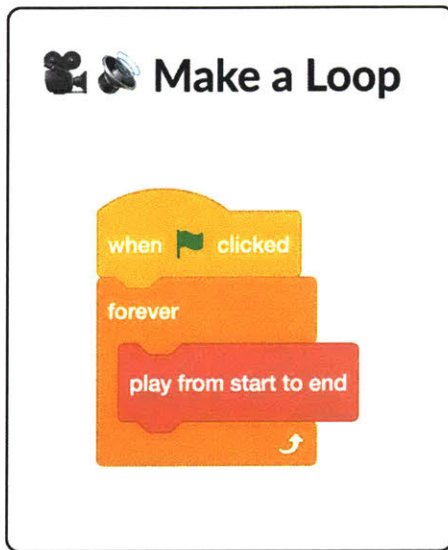


Figure 24 – Example of a Bricoleur card demonstrating how to loop a video or audio clip.

A set of coding cards (in the spirit of Scratch Coding Cards [14]) were also created to support the project making process. Each coding card was titled with a simple, common task that maker might want to perform with code (e.g., “play a video” or “make a loop”) along with a minimal example of a block stack that would accomplish that task (Figure 24). Icons at the top indicated if the code was applicable to video clips, audio clips, or both. A complete set of Bricoleur coding cards is given in Appendix C.

The general structure of a workshop followed a model we use in Lifelong Kindergarten called Image-Create-Share. For Bricoleur workshops, this consisted of an ice breaker activity, a quick demo of the tool including showing a few examples of projects (the Imagine phase), an extended time for project making in groups of 2 to 3 (the Create phase), and finally a time to show off projects and reflect on the experience with the other participants (the

Share phase). Occasionally, a workshop would have a specific focus like making projects using the tilt or compass features, but most were open-ended in terms of project theme.

As a way of examining these workshops, we use Mitch Resnick’s 4 P’s of creative learning – projects, passion, peers, and play – to provide four lenses into the workshop experiences. The 4 P’s assert that creative learning experiences are characterized by makers “working on projects based on their passions, in collaboration with peers and in a playful spirit” [11]. While the word “play” is most directly analogous to bricolage and tinkering, all four P’s are certainly present in any rich bricolage experience.

Projects

What kinds of projects do people make with Bricoleur? Moreover, what kinds of projects can people make with Bricoleur. To investigate this, we draw upon Jay Silver’s notion of a “sample project space” [15]. The idea of the sample project space is that one concrete sample project made with a given tool defines a point in the space of all possible projects. Once there are two sample projects (ideally, as different from each other as possible), the space of project possibilities expands to not only the first point and the new second project point, but the entire continuum of projects on the line connecting the two. Creating a third (non-collinear) project now establishes an entire space in the triangular area between the three points as a space of possible projects to make with the tool (Figure 25). As more unique examples are created, the space becomes wider and wider. In this spirit, we describe a few types of projects that hopefully provide a sense of the possible project space of Bricoleur. Which isn’t to say these are the only possible types of projects, but are merely examples of parts of the terrain that have been explored so far.

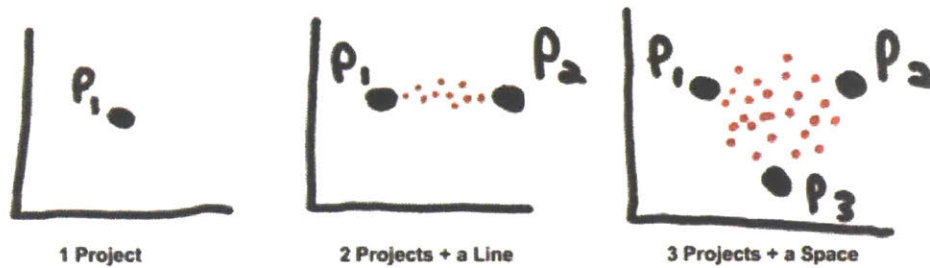


Figure 25 – Jay Silver’s sample project space. (credit: Jay Silver [15])

Stories

Certainly, it is possible to create projects that operate the way that traditional video media does. That is, makers can create projects that consist of sequenced video and audio clips that are played in order from beginning to end. While these projects do not make use of the interactive capabilities of Bricoleur, a rich space of project possibilities still exists in this realm. In particular, makers can create stories through sequencing video clips and syncing audio with the sequence (Figure 26).

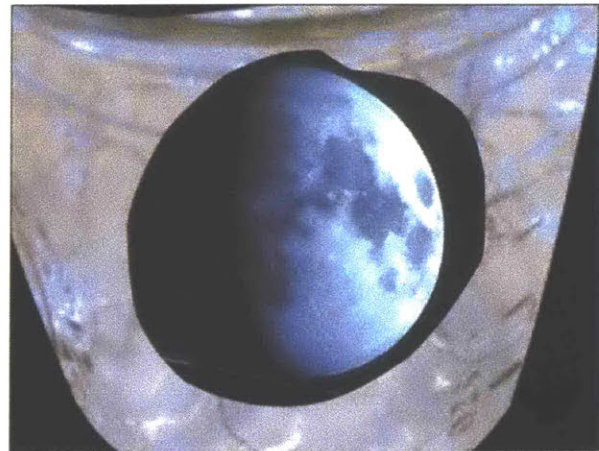


Figure 26 – A still from a story project about the moon.



Figure 27 – A “puppet show” project (also shown in Vignette Two) where each character can be tapped to trigger sound and animated movement.

Performances

As opposed to traditional video media that is simply played from start to finish, many Bricoleur projects are performed rather than played. That is, makers take advantage of the blocks that allow for project interactivity to create works that must be interacted with in order to be experienced. Here, we look at two specific subtypes of performance-based projects.

Puppet Shows

A typical way makers create performance projects is by putting multiple video clips on the canvas that each start playing only when they are tapped. The maker must then perform the project for others by tapping on each clip to trigger the behavior (Figure 27).

During workshops, this was sometimes referred to as a “puppet show” project since the maker had to perform each clip/character/etc. with their hands. Puppet projects can also be performed through movement by utilizing the tilt sensor and compass blocks.

Musical Instruments

Musical instruments form another class of performed Bricoleur projects. By using (primarily) the tap recognition blocks, makers can create melodic instruments, drum machines, and soundboards, to name a few (Figure 28). By connecting the code of each tapped video clip to trigger an audio clip to play, makers create projects that are playable in the sense that a piano is playable. This is slightly different from the “puppet show” type of performance examined above since the focus is on creating a new tool within Bricoleur that itself can be expressive and musical (Figure 28).



Figure 28 – A soundboard project where polyrhythms can be explored by tapping on each video clip to start a looped video and sound.

Collage

The design of Bricoleur quickly enables makers to put a variety of visual images into a single composition — a type of collage. While we see collage aspects in some of the examples above, the primary focus of many makers’ projects is collage itself. In some instances, makers create abstract collages that explore shape, color, texture, etc. for their own sake (Figure 29). In other instances, makers create a specific composition by collaging video clips (and audio) together. These collages can be standalone artworks or dynamic projects that can be interacted with (Figure 30).

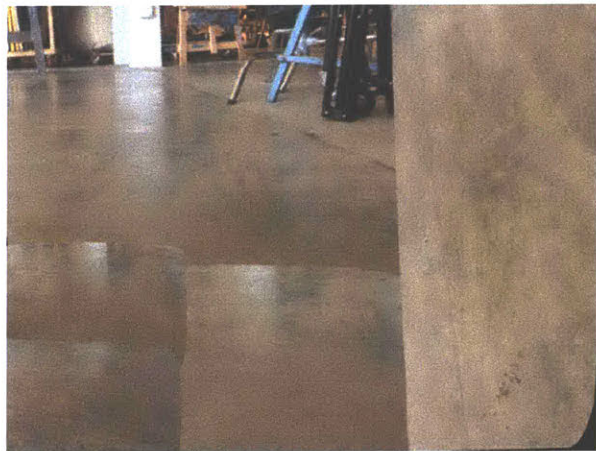


Figure 29 – A project exploring collaging together different images of concrete floor texture.



Figure 30 – A interactive tree made of body parts. Tilting the device causes the tree branches to sway. Tapping the tree causes apples (made of faces) to fall from the tree.

Dynamic Objects

The portable nature of the tablet hardware allows for projects to be embedded in the physical world in unique ways. In one workshop, participants were inspired by the shape of the protective iPad case. Its size and carrying handle evoked the idea of a handbag for the participants, so they created a “interactive purse” project that could be carried around. By using the tilt and compass blocks, video clips on the canvas would respond to the movement of the person carrying the purse (Figure 31).



Figure 31 – An interactive purse project inspired by the carrying case. The video clips respond to the tilt angle of the device.

To share one other unique example, one group of makers used Bricoleur as a projector for a makeshift hologram setup. By constructing a plastic reflector and placing it on top of their playing Bricoleur project, they could look into the reflector and see a “floating” version of their captured video images (Figure 32).



Figure 32 – Two hologram projects running next to each other.

Passion

Beyond types of projects, it’s also useful to look at the subject matter and themes that arise as makers create projects. These themes tend to cut across the project types described above.

Body

The body, as a subject, tends to arise in many Bricoleur projects. Video seems to be a particularly salient medium for this (and has been true of video throughout its existence). When makers capture video images in Bricoleur, they can use the front-facing camera to capture images of themselves, or the back-facing camera to capture images of others. A common first project in Bricoleur involves capturing images of the body and collaging them together. Often, this takes the form of a body-mashup as described above and seen in (Figure 21). In other examples like the Body-Tree shown in (Figure 30), bodies become materials that makers use to create other objects and images.

Movement and Motion

Makers investigate movement and motion in a few different ways with Bricoleur. One way is simply through capturing images of different kinds of motion. Dance projects offer one example while other projects may investigate something like the movement of an object across the camera's field of view.

Motion can also be brought into Bricoleur projects through use of the tilt and compass blocks. For example, a clip can be triggered to start playing when the device is pointing in a particular cardinal direction. Makers can also use the value reporting blocks to experiment with "continuous" motion of the device to do things like scrub through a video clip's frames by tilting the device back and forth (Figure 33).

A third way makers investigate motion involves exploring motion within video clips themselves. A video clip can, in some sense, have "internal" motion. That is, when the maker moves the camera while capturing a video clip, that clip will carry a sense of motion when played back, even if the video clip remains stationary on the canvas. Some Bricoleur projects investigate this by using programming blocks to "cancel out" the motion. For example, if a video is captured while rotating the camera, programming blocks can be used to "undo" the rotation by rotating the clip in the opposite direction.

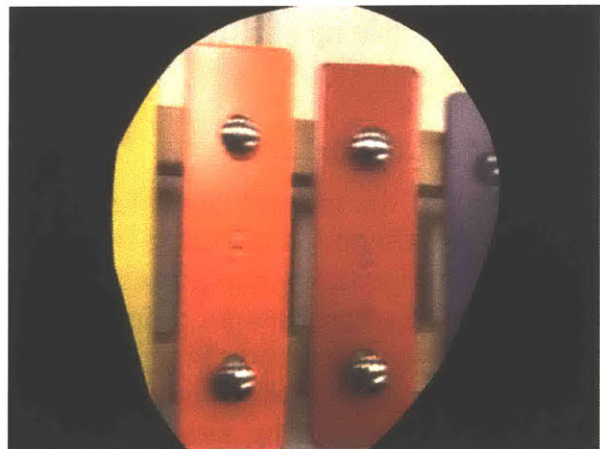


Figure 33 – A project where the video clip pans across the xylophone keys. As the device is tilted back and forth, the video frame continuously changes based on the tilt angle.

Materials

Especially when makers are first using the tool, they may create projects that experiment with a variety of materials and material properties. This includes material properties like color, texture, and pattern when it comes to video, but also includes various sound textures in the realm of audio (e.g., percussive sounds, vocalizations, music notes,

etc.). These projects tend to fall under the umbrella of “collages” as described above, but these types of material elements can also make their way into other types of projects. This is seen in other project types, for example, when makers fill the negative space behind the clips in their story with a “background” image of a particular color or texture. Sometimes, a particular object will be the focus of a project, be it an anthropomorphic character or something chosen purely for aesthetic reasons. In other cases, the maker will trace a video shape that is itself representative of an object and then use textural materials to fill in the pixels. An example of this kind of video texturing can be seen in (Figure 34) where the pupil of the eye is made up of a texture of colored feathers.



Figure 34 – A project that utilized video masking to define the shapes of the eye. The captured video images simply provide color and texture (e.g., the feathers in the pupil shape).

The Natural World

Elements of the natural world tend to arise in projects as well. Trees, water, animals, weather, and the moon are just a few that arose over the course of the workshops in various projects. Certainly, the portability of the camera facilitates the process of capturing images of one’s environment. Some makers took advantage of exactly that and captured natural and environmental images from the world around them. In other cases, makers would capture representations of natural phenomena, for example, through drawing on paper and then capturing a video clip of their drawing. In still other cases, makers would draw a video mask in the shape of the object they wanted to capture and then use images only to provide texture to the shape, as described in the previous section. That is, the shape itself was used to represent the phenomenon the maker wanted to explore.

Peers

One of the most surprising aspects of the workshops was the social nature of the making process. While we often think of programming as a solitary activity, creating and programming with Bricoleur seemed to be anything but that. Collaboration played out in several different ways, and at different stages of the making process, of which we highlight a few here.

Capturing

The first task in making any Bricoleur project is to capture either an audio or video clip. In both cases, a common way for a team of makers to split up roles is for one to act as the “performer” for the camera/microphone while another takes on the role of “capturer”

who operates the camera or microphone during the recording process (Figure 35). When capturing a video clip, the performer may act in front of the camera, but in other cases they make take on more of a role of “puppeteer” for the camera, i.e., controlling objects within the shot. Depending on the shot, the capturer might remain stationary, or they might move themselves and/or the camera to take a motion shot. Makers often collaborate on selecting and/or building physical materials for the shots as well.



Figure 35 – Two makers exhibiting the “capturer” and “performer” division of roles.

Makers switch roles fluidly, an affordance allowed by the portability of the tool. That is, makers can switch performer/capturer roles simply by passing the device. This allows for styles of making in which a project can develop by each maker capturing a clip in turn while another performs.

Programming

Once makers have captured clips and begin to program, other modes of collaboration emerge. In many cases, the classic “pair programming” model develops naturally. That is, one maker will be physically operating the tool and snapping blocks together while another (or others) help to decide what scripts to build. Similarly to the role switching mentioned above, makers fluidly switch roles by passing the device among themselves. In this way, they can continue in a pair/group programming model, cycling through who is “driving” and who is “guiding.” At other times, makers will pass the tool between themselves in more of a “pass-and-program” model where each maker works on programming a single clip or script and then passes it to the next person to program something else.

One other unique way that makers collaborate on code is by comparing code between each other’s projects. When a group gets stuck, they will simply walk their device over to another group and compare code back and forth as a way of getting unstuck. While this is possible with laptop computers, the true portability of the tablet devices seems to allow this type of collaborating more easily.

Sharing

As described above in the Projects section, many Bricoleur projects need to be “performed” to be experienced by an audience. In this way, makers engage with peers when doing this kind of performance for each other to show off their creations. Furthermore, a maker can give their project to a peer and allow them to perform the project for themselves.

Sharing also happened throughout the workshops in a somewhat unexpected way. Many times when new workshop participants would first encounter Bricoleur, even before making a new project, they would go and look at the existing projects on the device (i.e.,

projects from past workshops). This was a nice way for new makers to see the possibilities of what they could make within the timeframe of a workshop as well as to gather ideas for types of clips to capture and see various programming strategies in action.

Play

Of the four P's, play is the one most directly analogous to tinkering and bricolage. "Play" doesn't simply refer to having fun, but rather refers to a way of working in which makers are constantly trying ideas out, experimenting, and refining their creations and ideas over time [11]. The connection with bricolage, in particular, is clear in that it is about a continuous engagement with materials. In what ways do makers exhibit this kind of playfulness when using Bricoleur? While there are many ways that makers exhibit play and playfulness, we call attention to three that are most specific to Bricoleur.

Blended Making

In his master's thesis, Kreg Hanning coined the term "blended making" to refer to a process of where makers are creating in both the digital and physical world, and moving fluidly back and forth between the two [3]. This style of working back and forth between physical and digital materials is ubiquitous with Bricoleur. In virtually all workshops, participants were observed in a trajectory of creating where they would move fluidly back and forth between physical and digital materials for as long as the workshop would run. Sometimes this took the form where a team would capture and recapture a scene from the physical world into the digital world many times over trying to get it "exactly right." In other cases, a team would start by capturing images or sound of a single object, program that clip for a while, and then use what they had made to spark new ideas for what to add to their project. At that point, they would return to the physical world to capture more. As noted above in the projects section, some Bricoleur projects end up existing somewhere between a digital and physical creation in the end (for example, the hologram project in Figure 32), which was an interesting and unexpected outcome from the workshops.

Styles of Video Masking

Certainly one of Bricoleur's most unique aspects is the ability to draw the shape of each captured video when recording. Makers experimented a lot with the hand-drawn video masking and over the course of all the workshops, several "styles" of video masking emerged as common practices. In particular, three broad styles became apparent: framing, isolating, and texturing.



Figure 36 – A video clip demonstrating the framing style of masking. The round drawn shape frames the cow in its surrounding context.

Framing is perhaps the most straightforward way of using the video masking. In framing, the maker draws a shape as a way of providing a frame for the contents of a particular shot. The frame may be simple (e.g., a rectangle or circle) or more complex (e.g., a “splat” shape). In any case, the role of the mask is to primarily to define the shape of a traditional video “shot” which may contain objects, people, and surrounding context (Figure 36).

Isolating, on the other hand occurs when a maker attempts to use the video mask to capture a specific object in the physical world, without its surrounding context. For example, a maker may carefully trace the outline around a person’s hand in order to turn the hand into an object in the scene. The rest of the image around the hand is irrelevant (or at worst, problematic) for the maker’s project, so they utilize the mask to isolate the hand from its surroundings. In the example shown in Figure 27, the characters are masked using the isolation style.

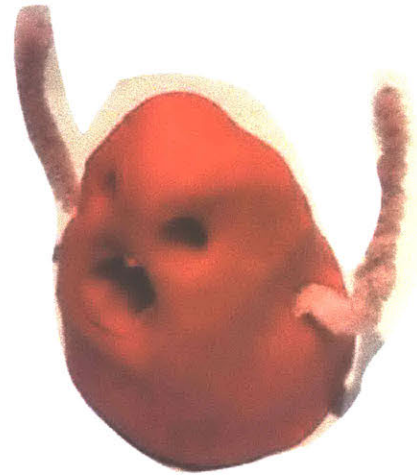


Figure 37 – An isolated character from the puppet show project shown in Figure 27.



Figure 38 – A textured teardrop shape from the eye project shown in Figure 34.

Finally, the third primary style of video masking involves using the video to texture the shape, as previously described above. In this style, the mask is drawn in such a way to represent an object in and of itself (e.g., a tree, a cloud, etc.). The maker then captures pixels into the shape only to give the shape texture. The pixels in the image are not the “subject” of the video clip, but rather work to help define the shape as its own subject. The eye example shown in Figure 34 utilizes texturing heavily (Figure 38).

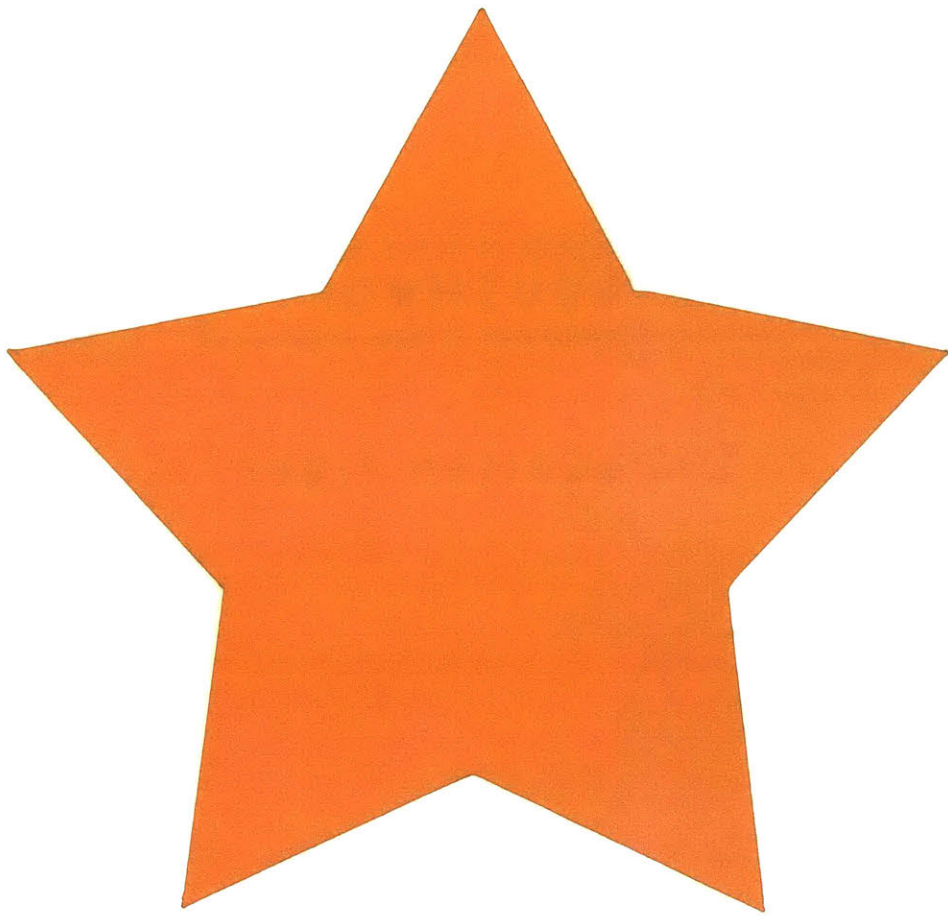
Tinkering with Programming

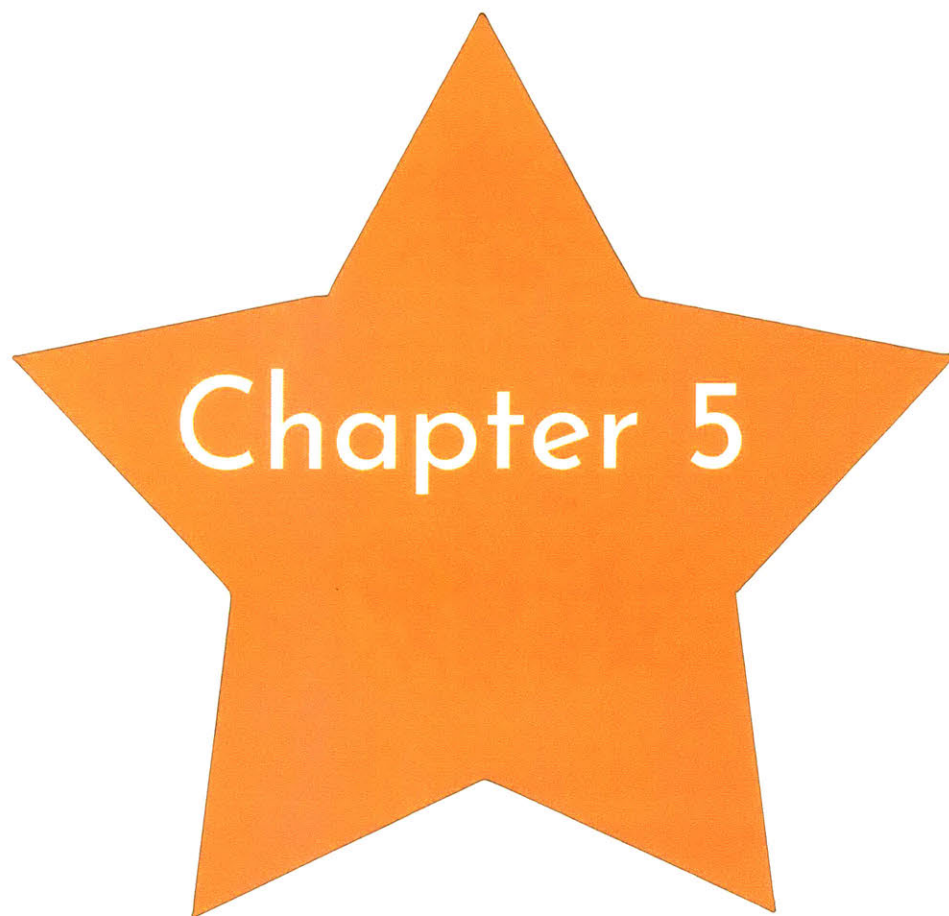
Using Scratch blocks as the programming interface in Bricoleur automatically brought with it a set of smart design choices that afford playfulness. For example, a block can be executed (even without dragging it out into the workspace) simply by tapping it, blocks can be “set aside” for later use by simply dragging them out of a script, and the lack of distinction between “editing” and “playback” mode.

In the case of Bricoleur specifically, the unique aspects of tinkering with code naturally came up in regard to how makers tinker with the video, audio, and interactivity blocks. In many ways, the style of making is similar to that of Scratch in that makers sometime experiment by tapping on different blocks, slowly building up scripts over time, and

rapidly iterating on their program designs. One major difference between Scratch and Bricoleur, however, is that each programmable asset has an “internal” representation of time (e.g., frames of a video, samples of a sound) in a way that standard Scratch sprites do not. In light of this, a lot of the tinkering with code in Bricoleur has to do with exploring notions of time, timing, and synchrony. “How do I get a video and sound to play at the same time?” was one of the most common questions that makers would ask during workshops. This afforded particular ways of engaging with code blocks and exploring different programming strategies, which will be elaborated upon in the next chapter.

Chapter 5





Chapter 5

Surveying

Outcomes

We have taken a deep look at the creation of our new landscape (the design work) as well as the explored the terrain (workshops). Now, we turn to the project of beginning to map the space by drawing connections and identifying relations that arise out of the use of Bricoleur. This mapping — drawing relations between points — is as much a cartographic project as it is an aesthetic one, in the sense that “aesthetics” are discussed in the introduction — i.e., as a sensibility that allows one to connect disparate concepts together. Here, the aim is not to declare exact outcomes for any specific workshop, let alone any specific maker, but rather to identify some ideas that makers have the opportunity to engage with when making projects with Bricoleur that are, perhaps, not as accessible with other tools or modes of creation.



Physical/Digital

There has been much research and writing on making experiences that involve both the physical and digital worlds. Similar to the “blended making” idea given in the previous chapter, these experiences tend to involve creating and connecting physical objects and digital objects, thus making artifacts that span the gap between the two worlds.

While making experiences with Bricoleur certainly can and do involve that type of physical/digital making, there exists another relation between the physical and digital

worlds which has a different quality. Rather than connecting physical materials with digital tools directly, the medium of video itself has a way of “holding” a representation of the physical world in its digital container. In talking with a group of collaborators from Italy, they noted that young makers tend to use the word *vero* when talking about the nature of videos in Bricoleur projects. The word has two translations in English, namely, “real” and “true.” Both recognize that the video maintains a relationship with the physical objects and space that was captured. In this sense, Bricoleur projects exist in relation to physical places.

This relation between project and place gives makers access to see their surroundings in a new light. From the maker’s physical place, objects, motions, colors, and more can be captured and, through programming, given new behaviors or become entirely new objects in a digital creation. The portability of the device further gives makers the freedom to move around, go outside, and capture images and sounds from all over their environment. In this sense, makers are empowered with the ability to capture and reprogram the world around them.

Body/Artifact

Bricoleur allows makers to engage their bodies in relation to their projects in numerous ways. We have already touched on many of these, but it is worth collecting them together into a single place to look at them in totality.

Makers get to use their bodies at all points in the process of making a project. The requirement of drawing the shape of each video clip immediately asks makers to bring a personal touch, quite literally, into their project. When capturing clips, they move their camera with their bodies to capture images, or move their bodies in front of the camera to perform. In many instances during workshops, makers would get up from chairs, and even leave the room or building to capture the clips that they desired. When programming, makers program with their fingers and pass the device back and forth to develop collaborative projects. Finally, when sharing projects with others, makers often perform their projects with their bodies through using tap gestures, tilting, or the compass blocks. Moreover, makers can move their bodies as they perform to create performances that are both physical and digital. Bodies also come up over and over again as subject matter in projects. Makers are often interested in capturing their face or faces of others around them, or using bodies to represent other objects or ideas in a project.

The affordances of video, audio, programming, and tablet hardware all combine to allow this kind of relation to emerge as strongly as it does in Bricoleur. It presents a context for work that is at once virtual and embodied, an unusual mixture in the outcomes of a creative tool.

I/We

Unlike many programming tools, Bricoleur gives makers a chance to engage with both themselves and with others throughout the making process. In the previous chapter, we

examined the various ways in which makers collaborate including taking on different roles while capturing clips, programming in both pair-programming and pass-and-program methodologies, and sharing their work through performance and from across workshops. Each of these types of collaborations are supported by the portability of the hardware, the nature of video and audio media, and the programming paradigm.

Moreover, the medium of video allows makers to capture images of themselves and others around them, thereby giving a maker a quick entry point to creating work about, with, and for oneself and the people in one's life. Bricoleur projects can be at once both personal and collaborative, giving makers new ways to connect with themselves and others.

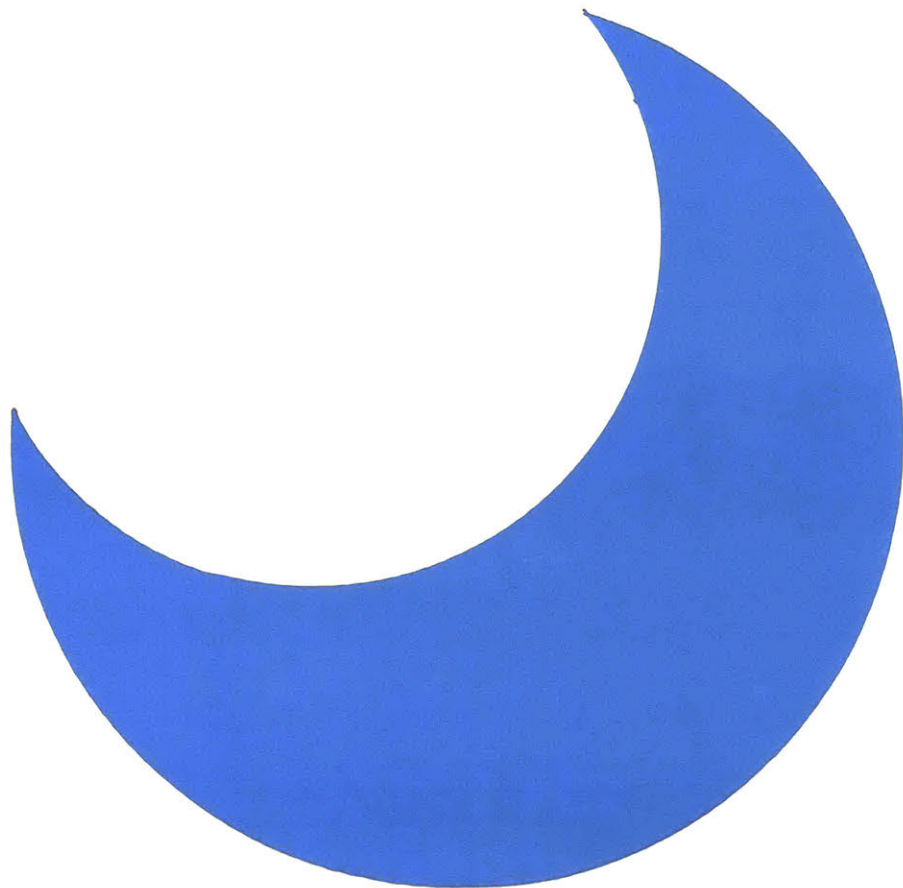
Computation/Time

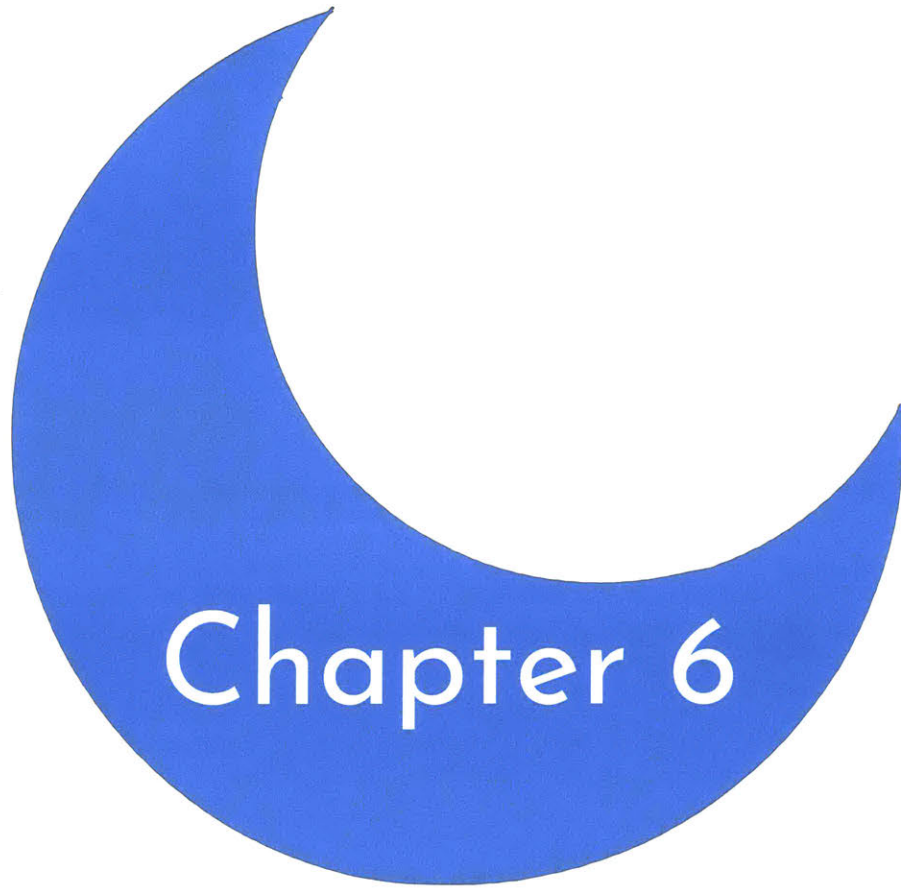
Bricoleur projects are created through programming time-based media — video and audio. In light of this, many of the most common desires for makers when programming their projects have to do with managing time without a standard timeline user interface. While this is also true for other media programming platforms — Scratch, for instance — the fact that video and audio have their own “internal” representations of time adds a new layer of consideration to the making process. Makers must engage with code in order to activate the time-based nature of their captured media. As a result, creating with Bricoleur sets up a relation between computation and time within a project. That is, thinking about time gives makers an entry point to thinking about programming, and in reverse, snapping blocks together to create scripts asks the maker to consider the temporal consequences of their programming on their media clips.

Throughout the workshops, several common tasks emerged that makers wanted to accomplish when it came to programming. To name a few, makers would often want to sequence clips (play one after another), loop a clip (play the same clip multiple times, sometimes infinitely), synchronize two clips (e.g., play a video clip and a sound simultaneously), and play a clip while adjusting its properties (e.g., play a clip while rotating it on the canvas). Each of these tasks naturally led to using and understanding certain programming constructs. Sequencing clips requires sequencing code blocks, repeating a clip involves using a loop construct, synchronization can be accomplished through event triggers or message passing, and playing a clip while adjusting its properties naturally leads to parallelization of scripts. Message passing, in particular, was a computational idea that many makers learned about through the process of creating projects since playing a video with a sound was a very common desire.

In other cases, makers would use the programming blocks to understand more about the nature of the media. For example, the `jump to frame [X]` block provides makers access to the underlying model of the media clip. Using the `set play rate to [P]%` allows one to consider the way in which a video is ultimately a sequence of still images played back at a particular speed. In any case, Bricoleur has shown that programming with time-based media affords a unique way to understand computational concepts and vice versa.

Chapter 6

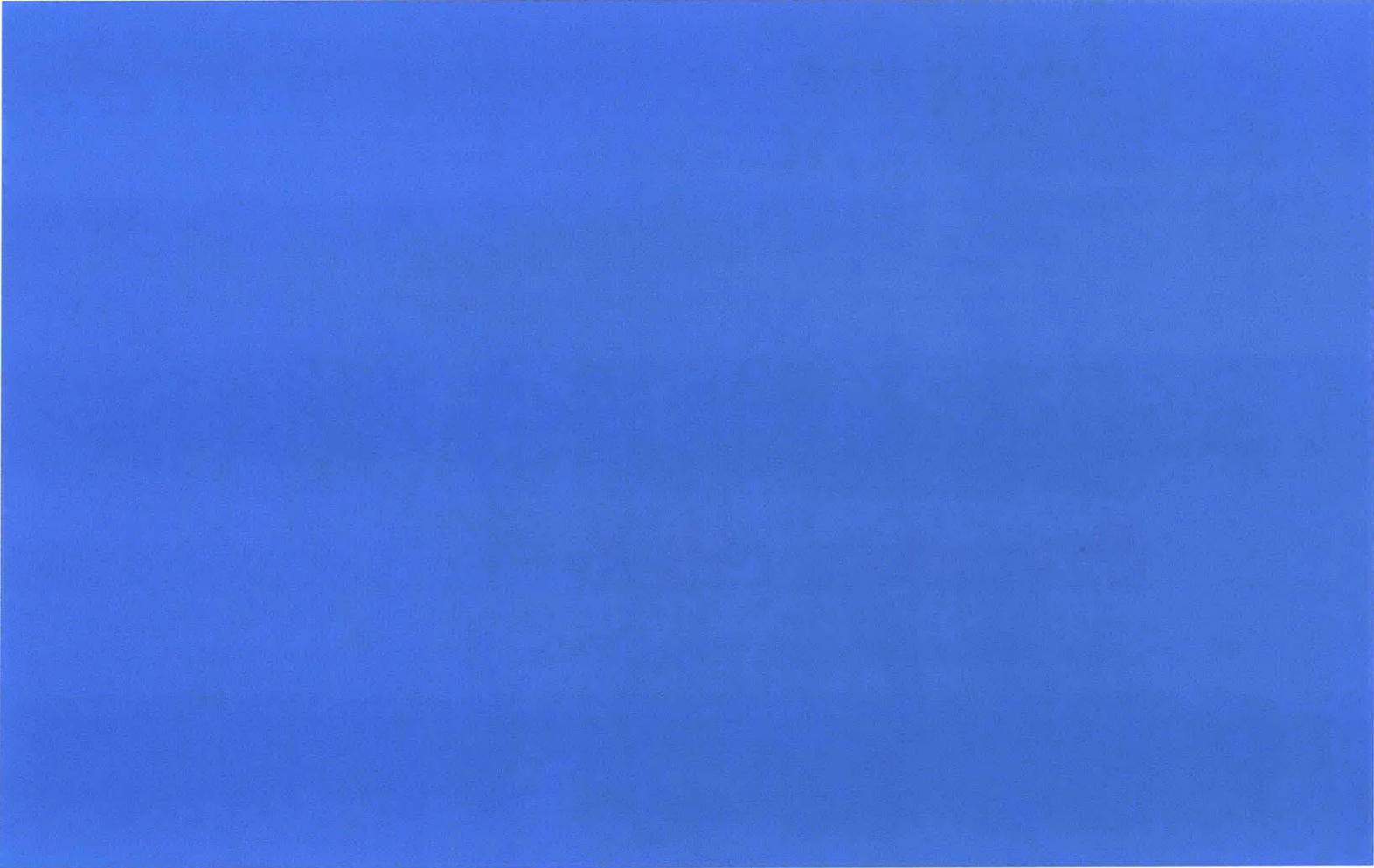




Coming Home

Reflections and Future Work

We now finish our journey by returning home to reflect on our explorations and mappings. Moreover, we use our reflections to think about new journeys we could take at a future point.



In this thesis, we have seen how the combination of time-based media, blocks-based programming, and mobile hardware can give rise to new ways to engage with materials and create expressive projects. Video and audio, in particular, afford new ways for makers to create projects related to the body and performance. Programming further enables performance-based projects and also give makers access to thinking about time in relation to computation. Mobility allows makers to capture media of the people and space they live in and also provides fluid ways to collaborate on programming and project making. Moreover, the amalgamation of all of these — time-based media, programming, and mobility — can provide a new context for creating, as Bricoleur shows.

In this chapter, we consider further directions for this project as well as other research ideas that this project can inform.

Bricoleur

As is true of any iterative design project, there are plenty more avenues to explore when it comes to the design of Bricoleur itself. The work in this thesis has attempted to pull out some of the most unique expressive potentials of Bricoleur including involving the body in making interactive projects. Future work could focus on developing further mechanisms for creating with and for the body. This could take the form of new types of blocks to support interactivity (gesture detection, sound sensing, accelerometer data, etc.) as well as other mechanisms and materials to support movement and performance. Other design directions could address optimizing the programming experience for mobile screens (elaborated upon in the next section), bringing Bricoleur to mobile phones, incorporating live video, and more.

One big question to consider for the future of Bricoleur is whether it will be publicly released. While this is one potential outcome for the project, it is also possible to view the project as simply a research experiment to inform development on other tools in the future. There is also the question of whether some aspects of Bricoleur could be brought into Scratch itself.

If Bricoleur were to be released publicly in some form, a natural question arises around if/how to allow makers to share their projects with each other — perhaps similarly to how the Scratch community allows project sharing and remixing. While there would undoubtedly be many upsides to this kind of sharing platform, there are a lot of complex considerations that would take time and effort to parse out. Privacy, safety, and moderation concerns have an added weight to them when working with video assets, and even more so when users are children. Scratch has built a strong culture where users know not to share identifying information on the site, but the nature of video is such that it is fluid and easy to capture one's face, location, and more. Building a sharing platform for Bricoleur that is rich and communal, while simultaneously safe, is likely a research project in and of itself.

Regardless of the future of the tool, there is much more work to do in exploring the space of projects that can be made as well as the way that makers' thinking evolves as they create in Bricoleur (for example, thinking about time, environment, etc.). This might mean connecting with a group of young makers to work on projects in an ongoing way over an extended period of time. Another potential direction is to connect with an artist or artists to see what types of projects they make and how that could inform future versions of Bricoleur.

Mobile Programming

Beyond Bricoleur as a specific tool, this project has elucidated some of the opportunities and challenges of programming on mobile devices. As the projects in this thesis show, there is a lot that one can express with just a little bit of code on a small screen. Furthermore, by giving makers the power and tools to program on mobile devices, they can capture, program, and therefore reimagine and redefine the environment around

them. When makers can take mobile devices into the city, the outdoors, a friend's house, or wherever they choose and program expressive projects in situ, new creative potentials arise. In light of this, there is future work to be done on designing creative programming tools specifically for mobile devices.

Certainly, a touch-friendly blocks-based programming environment is far more usable on a mobile device than text-based programming. Makers can program using only a single finger and without need of a keyboard. Nonetheless, there are inherent challenges in programming on small form devices — e.g., the limited space for code can hold some makers back from exploring longer and more complex scripts. This can get especially challenging when the project output itself takes up screen real estate, as is the case with Bricoleur's video canvas.

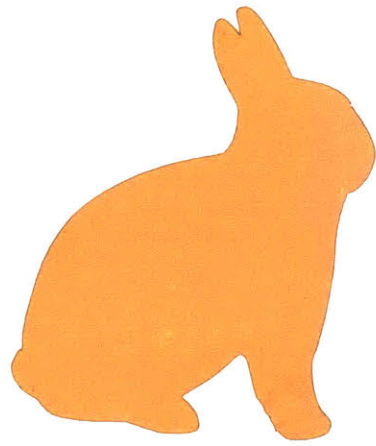
One way of addressing these challenges could be through designing new interfaces for blocks programming on small screens. Scratch offers a horizontal blocks grammar (similar to the horizontal blocks already used in ScratchJr) which may offer better usability. However, the horizontal blocks grammar lacks some constructs (e.g., branching statements, value “reporter” blocks, and more) which may preclude its usefulness for some projects. Perhaps future design work on the existing vertical blocks grammar could implement ways to collapse block stacks into smaller icons, thereby giving makers the ability to free up room in the user interface for scripts that they are currently programming.

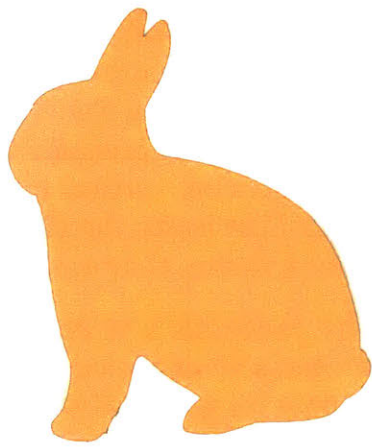
Beyond addressing the usability concerns, a more fundamental question about programming on mobile devices is: what sort of projects are mobile devices best suited for creating? In thinking about designing for yet smaller devices (i.e., phones), what sorts of making experiences would be powerful and generative? Perhaps there is an opportunity to build a creative programming tool focused specifically on sound and music, thereby eliminating the necessary screen real estate for image or video output.

Microworlds

Like any good research project, several unexpected outcomes arose during this project: the way that makers collaborate, the way they engage their bodies with the making process, and the way that makers think about computation and time simultaneously, to name a few. In light of this, Bricoleur can be considered a sort of microworld, in the sense that Papert used the word, for makers to engage with powerful ideas about time, performance, environment, and more.

This naturally leads to the question of what other related microworlds are out there, ready to be developed and explored? Perhaps Bricoleur is just one example of a microworld that exists in a much larger universe of microworlds. Finding, exploring, and mapping that universe is quite possibly the next big step.





Appendix A: Technical Notes

This appendix contains technical documentation and implementation details of Bricoleur. Hopefully, this will be useful to those trying to use Scratch Blocks and the Scratch Virtual Machine (Scratch VM) for a custom project, to those interested in real-time video rendering, and/or to those who are simply curious.

Overview

At a technical level, the Bricoleur editor consists of three major subsystems: the programming system, the renderer, and the application layer. Broadly, the programming system is responsible for the blocks-based programming environment, the renderer is responsible for drawing project output to the screen and sending audio to the speakers, and the application layer is responsible for coordinating between the other two systems. We'll begin by explaining each system in more detail.

The Programming System: Scratch Blocks & Scratch VM

Bricoleur uses Scratch Blocks and the Scratch VM to power the blocks-based programming environment. Scratch Blocks provides the UI which allows the user to manipulate programming blocks on screen while the Scratch VM is responsible for maintaining the underlying state of the system (e.g., properties of renderable entities) as well as executing the code provided by Scratch Blocks. The Scratch VM executes code in discrete time steps — in Bricoleur, the VM is “ticked” 30 times per second by the application layer. On each tick, all running code blocks and stacks are executed by the VM, updating the internal state of the system.

Bricoleur has two types of renderable entities that can be programmed: video clips and audio clips. Each clip is represented in the VM as a “target,” which is simply an entity with internal state that also has code blocks associated with it. Bricoleur extends the Scratch VM by defining two new types of targets (video and audio) that have custom properties appropriate for each. For example, a video target has properties such as total frames, play rate, position on screen, etc. On each tick of the VM, each target's running code blocks execute and update the state of the target's properties.

While the state of each target is kept in the VM, the actually renderable data (i.e., video frames and audio samples) are stored on disk and managed by the application layer. As a result, there were choices to make about which subsystem was responsible for which parts of asset management. Early prototypes of Bricoleur actually kept some state in the VM and other state in the application layer — for example, the state of whether a video was “playing” was kept in the application layer. However, this led to complications in keeping the VM in sync with the application layer and required two-way communication between the systems (e.g., the application needed to inform the VM when a video had finished playing). As a result, Bricoleur adopted the following principle.

State Principle

The state of all renderable entities is kept and maintained entirely within Scratch VM.

While this required some significant technical work on the VM (for example, the thread sequencer in the VM needed to be extended to manage video and audio playback), it paid off many times over in reducing complexity and coupling between the subsystems. This approach is highly recommended for anyone using Scratch VM in their own project.

The Renderer

The renderer is the code responsible for drawing image assets on the screen as well as sending audio to the sound hardware. In fact, the renderer itself is split into the video renderer and the audio renderer.

On each frame tick after the VM has run, the video renderer receives a buffer that contains instructions for rendering each on-screen video asset. Each entry in the buffer contains information like a pointer to the video asset, the frame number to draw, the transform matrix for positioning, and effects parameters. From there, the video renderer takes a standard GPU rendering approach by using the information in the buffer entries to fill GPU vertex and texture buffers with the necessary data, then sending those off to the GPU to be rendered to the screen. A simple vertex shader positions the video assets on screen and a fragment shader draws textures and applies effects. The current version of Bricoleur uses Apple's Metal graphics API, but any modern graphics API would suffice.

The approach of using a buffer to provide instructions to the video renderer is useful since it provides a very simple description that any rendering backend could use. For example, it would be straightforward to swap out the Metal renderer with OpenGL, a software renderer, or any other system.

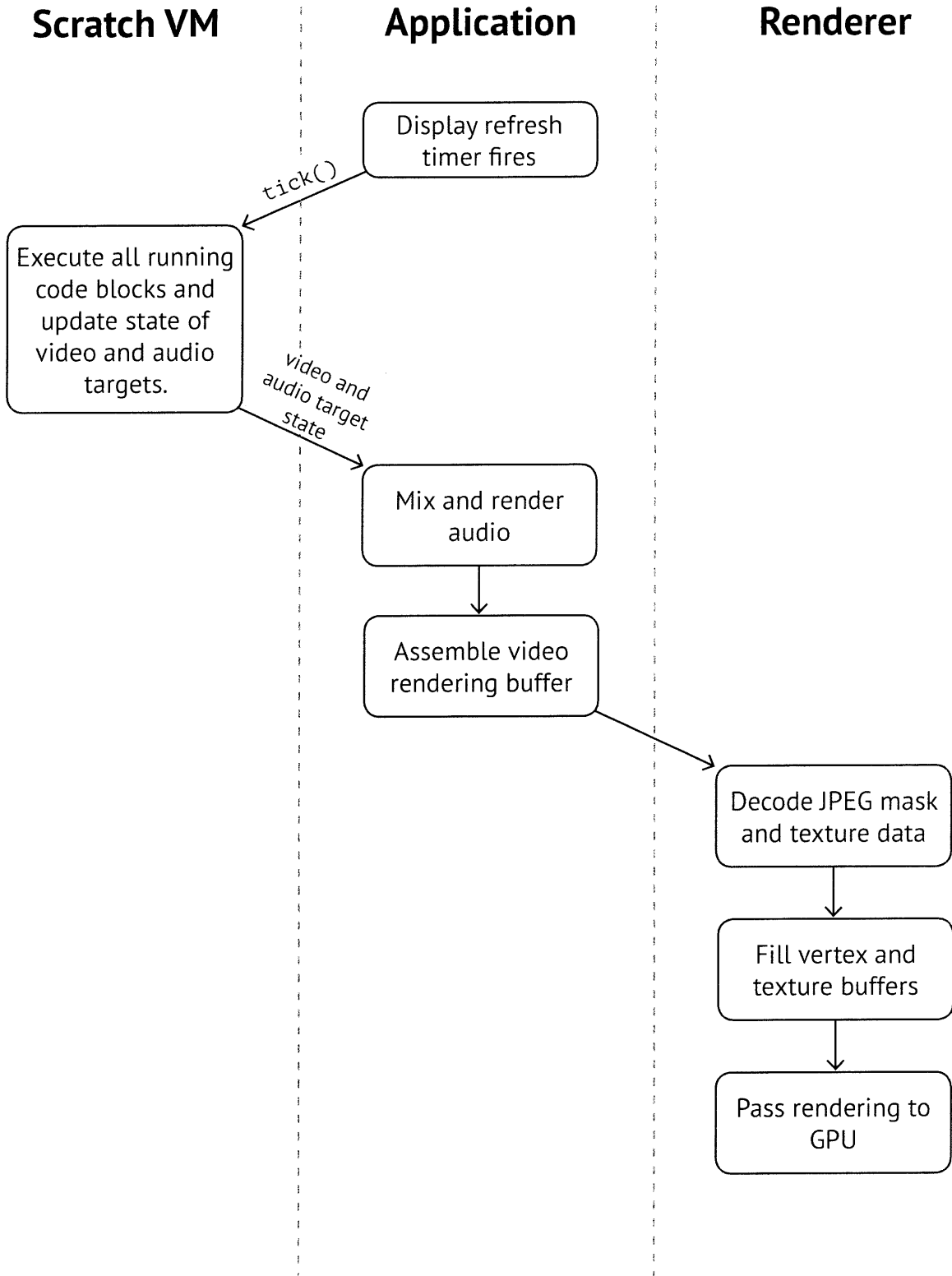
The audio rendering system is very simple. After the VM has ticked, the audio renderer receives the samples it needs to output to the speakers and writes them into a circular buffer. The operating system then reads samples out of that circular buffer on a background thread when it needs to output samples to the hardware.

The Application Layer

Finally, the application layer consists of the code that connects the programming system to the renderer, as well as the interface to the device hardware (gyro, compass, etc.) and the UI that the user interacts with (besides the programming blocks). The application layer also is responsible for setting up the timer that drives the VM and the renderer drawing code. This timer is itself driven by the display so that frame ticks are synced with the display refresh ("v-sync"). On each display refresh, the application layer calls into the VM to execute a tick, then gets the state of the targets from the VM, sends audio samples to the audio renderer, and builds the rendering buffer for the video renderer. When not managing the rendering loop, the application layer also manages user interactions (touch events, etc.).

The Rendering Loop: Bringing It All Together

In total, executing and rendering a frame in Bricoleur looks like:



Video and Audio Assets: A Fundamental Challenge

By design, Scratch Blocks and the Scratch VM work such that any change made to the code is executed by the VM on the next tick. Which is to say, not only is there no compilation step in the programming environment, but also that any change to the code must be reflected in the rendered output the next time the screen is drawn. Moreover, this means that the renderable output of any given frame is not known until the VM executes the code for that frame, just milliseconds before the frame needs to be drawn to the screen. This is much more akin to how a video game works than how most programming languages work. Still, in a video game, the content on screen usually doesn't change radically from frame to frame — the assets that were drawn last frame are likely to be the same ones that will be drawn this frame. With Scratch Blocks and Scratch VM, however, the user could tap a block at any time that requires an asset to be drawn to the screen that wasn't rendered the previous frame.

This presents a fundamental challenge when it comes to building a renderer to work with the Scratch VM: all renderable content in a project must be able to be randomly accessed, processed, and output to the screen or speakers within a single 33ms frame (and in fact, shorter than that since some time is taken up by the execution of the VM). This constraint necessarily informed the technical design of video and audio assets in Bricoleur.

Video

Tablet hardware devices typically provide programming interfaces for capturing video content through the operating system. However, the codecs available for video capture tend to use interframe compression schemes which minimize size and facilitate linear playback, but are highly non-optimal for randomly accessing frames. Bricoleur also has the added complication of capturing non-rectangular video assets, which are not handled at all by standard video formats. Since random frame access and non-rectangularity were absolutely essential requirements for Bricoleur, a custom video file format was developed.

As described in the thesis, video assets in Bricoleur consist only of images (by design), which made designing a custom file format considerably easier. In essence, the file format is a variant of Motion JPEG with a custom file header. Motion JPEG encodes a video stream as a simple concatenated sequence of JPEG images, one per frame. This allows for easy access of individual frames while still allowing each frame the benefits of image compression.

Bricoleur's file format uses the same JPEG compressed frame scheme, along with some additional metadata at the head of the file. JPEG was chosen as the compression method since it allows high compression ratio and fast decoding (essential in the rendering step). To capture non-rectangular assets, Bricoleur takes the approach of using image transparency to mask out pixels outside of the shape drawn by the user. However, standard JPEG does not permit transparency/alpha in images. As a result, the "shape" for each video asset is stored as a grayscale mask image, also encoded with JPEG. Since the mask is stored separately, the video renderer sends both the mask texture and the video

frame texture to the GPU when rendering an asset. The fragment shader then blends and discards fragments according to the value in the mask texture.

In order to facilitate random access, the header for each video file contains the byte offsets for each frame in the image sequence. This way, when an asset is loaded into memory, accessing a particular frame requires only a single lookup. The length of video asset capture was limited to 10 seconds, primarily to encourage makers to capture short clips, but served the dual purpose of providing hard boundaries on the size of captured video files. Therefore, there can only be up to 300 frame offsets stored in the file header (10 seconds @ 30 frames-per-second gives a maximum of 300 frames per video asset).

Table A1 describes the layout of the Bricoleur video file format.

Audio

Audio assets are about as simple as they could get. The audio renderer is set up to output at 48000 samples per second, so audio is captured at that same sampling rate. Since we currently only allow recording from the single-channel microphone, an audio asset is simply a linear PCM stream of 16 bit single-channel audio at 48000 samples per second. Audio assets are stored on disk as the raw sample data, since there is no need for a header containing metadata the same way that there is for video assets.

Project Format

A Bricoleur project is simply a folder on disk named with a unique identifier. The structure of that folder is as follows:

project_folder	(named with a UUID)
├ audio	(folder containing audio asset files)
├ video	(folder containing video asset files)
├ project.json	(Scratch VM file format - stores state for all targets in the project)
└ thumb.png	(thumbnail image for display in the UI)

Early prototypes of Bricoleur used a database to store project data, but this turned out to be cumbersome. Ultimately, using folders for projects had several benefits:

- No need to set up a database (i.e., the filesystem already IS a database)
- Projects can be moved around simply by moving folders around
- Projects can be serialized into a single file through a simple compression scheme like ZIP (and thus shared easily)
- Projects can be modified easily by hand if need be

In order to keep track of all projects in the system, Bricoleur uses a single flat text file consisting of one project UUID per line.

Table A1 - Bricoleur Video File Format Specification

Section	Offset	Data	Notes
Header	0x0	Magic Number	Value is 0x00F1DE0
	0x4	Total Frames	1-300
	0x8	Image Width	1-640
	0xC	Image Height	1-640
	0x10	Mask Offset	Offset in bytes from the beginning of the file to where the mask JPEG data starts
	0x14	Mask Length	Length in bytes of mask JPEG data
	0x18	Video Frames Offset	Offset in bytes from the beginning of the file to where the video frames JPEG data starts
	0x1C	Video Frames Length	Length in bytes of total video frames JPEG data
	0x20	Frame Offset 0	Offset in bytes from the beginning of the frames data to where frame 0 begins
		Frame Offset 1	Offset in bytes from the beginning of the frames data to where frame 1 begins
		...	
		Frame Offset n	Offset in bytes from the beginning of the frames data to where frame n begins
Mask	0x20 + (number of video frames * 4)	Start of Mask JPEG Data	Grayscale JPEG Format
		...	
		End of Mask JPEG Data	
Frame 0	0x20 + (number of video frames * 4) + length of mask data	Start of Frame 0 JPEG Data	Color JPEG Format
		...	
		End of Frame 0 JPEG Data	
Frame 1		Start of Frame 1 JPEG Data	
		...	
		End of Frame 1 JPEG Data	
⋮		⋮	
Frame n		Start of Frame n JPEG Data	
		...	
		End of Frame n JPEG Data	End of file

Performance

The biggest performance bottleneck by far in the whole system is the decoding of JPEG frames when rendering. One potential strategy for mitigating this in the future would be to use a image compression format that can be decoded natively on the GPU (e.g., DXT, ASTC, etc) instead of JPEG. Still, this would need testing to see if frames can be encoded fast enough during video capture into one of these formats.

Technical Design Takeaways

For those interested in using the Scratch VM for a custom project and/or designing and building related tools, we provide some overall takeaways from the process of making Bricoleur here:

- Use the Scratch VM to maintain all system state.
- Keep subsystems clean and decoupled. Provide simple interfaces between subsystems.
- Use simple file formats that can be inspected and modified by hand for all persistent data whenever possible. (This is especially useful in the early stages of a project when file formats may be changing rapidly).

Appendix B: Idea Spark Cards

The following table gives a full list of the idea spark cards used in Bricoleur workshops. Participants were invited to pick a random “structure” card and a random “strategy” card as a way to generate initial ideas for projects. For example, a participant might pick “a cycle...in total darkness” or “an experiment...with tension.” The combinations of cards are intended to provide “open constraints” – i.e., ways of constraining the possible project ideas while remaining open ended enough to be taken in many different directions depending on the interpretation of the person picking the cards.

Structures	Strategies
<ul style="list-style-type: none"> • A Poem • A Story • An Experiment • A Dance • A History • A Diagram • A Process • An (Un)still Life • A Cascade • A System • A Question • An Investigation • A Tangent • A Cycle 	<ul style="list-style-type: none"> • Extremely close • Only one sound • Monochromatic • Go Outside • Invert Everything • Make it Sing • Far Far Away • With Gravity • In Three Words • Always the Same/Never the Same • Split Screen • Multiple Perspectives • Light/Shadow • For Two or More People • Don't Move the Camera/Only Move the Camera • From High Above/From Far Below • In Tandem • In Harmony • With Tension • Very Very Slowly • On the Ground • Sunlight/Moonlight • In Total Darkness

Appendix C: Bricoleur Cards

Bricoleur cards were inspired by Scratch Coding Cards to provide makers quick code examples of common tasks in Bricoleur (e.g., “play a video” or “make a loop”).

The image displays seven Bricoleur cards, each containing a specific code snippet for a common task. The cards are arranged in three rows: the first row has three cards, the second row has three cards, and the third row has two cards.

- Play a Video:** A yellow 'when clicked' block followed by a red 'play from start to end' block.
- Play a Sound:** A yellow 'when clicked' block followed by a red 'play sound until done' block.
- Make a Loop:** A yellow 'when clicked' block followed by an orange 'forever' loop block containing a red 'play from start to end' block.
- Make a Boomerang:** A yellow 'when clicked' block followed by an orange 'forever' loop block containing two red blocks: 'play forward' and 'play in reverse'.
- Play with Sensors:** Two red blocks: 'when tilted right' and 'when pointed toward north'.
- Play with Time:** An orange 'wait 1 seconds' block, a red 'when I reach end' block, a red 'play from 1 to 2' block, and a red 'set play rate to 50%' block.
- Play with Effects:** Two red blocks: 'change color effect by 5' and 'set whirl effect to 200'.
- Sync Video & Sound:** Two parallel code paths. The 'Video' path has a yellow 'when I receive message1' block followed by a red 'play from start to end' block. The 'Sound' path has a yellow 'when I receive message1' block followed by a red 'play sound until done' block. Below these is the text 'Start them together!' and a yellow 'when clicked' block followed by a yellow 'broadcast message1' block.

References

- [1] Boyle, D. 2018. From Portapak to Camcorder: A Brief History of Guerrilla Television. *Journal of Film and Video*. 44, 1-2 (Oct. 2018), 1–14.
- [2] Hanhardt, J.G. (Ed.). 1986. *Video Culture*. Layton, Utah : G.M. Smith, Peregrine Smith Books, in association with Visual Studies Workshop Press.
- [3] Hanning, K. 2018. *Tinkering with ScratchBit: Explorations in Blended Making*. Master's thesis. Massachusetts Institute of Technology, Cambridge, MA.
- [4] Illich, I. 2009. *Tools for Conviviality*. Marion Boyars Publishers Ltd.
- [5] Jenkins, B. In the Bedroom/On the Road. *Millenium Film Journal*. 58, 146–153.
- [6] McCarty, A.N. 2005. *Toying with Obsolescence: Pixelvision Filmmakers and the Fisher Price PXL 2000 Camera*. Master's thesis. Massachusetts Institute of Technology, Cambridge, MA.
- [7] Papert, S. 1993. *Mindstorms*. New York : Basic Books.
- [8] Papert, S. 1993. *The Children's Machine*. New York : BasicBooks.
- [9] Piaget, J. et al. 1977. *The Essential Piaget*. New York : Basic Books.
- [10] Puckette, M. 2002. Max at Seventeen. *Computer Music Journal*. 26, 4 (2002), 31–43.
- [11] Resnick, M. 2017. *Lifelong Kindergarten*. Cambridge : MIT Press.
- [12] Resnick, M. and Rosenbaum, E. 2013. Designing for Tinkerability. *Design, Make, Play*. M. Honey and D. Kanter (Eds.). 163–181.
- [13] Resnick, M. et al. 2009. Scratch. *Communications of the ACM*. 52, 11 (Nov. 2009), 60–67.
- [14] Rusk, N. and the Scratch Team. 2016. *Scratch Coding Cards*. No Starch Press.
- [15] Silver, J. 2014. *Lens x Block: World as Construction Kit*. Ph.D. dissertation. Massachusetts Institute of Technology, Cambridge, MA.
- [16] Somerson, R. and Hermano, M. (Eds.). 2013. *The Art of Critical Making: Rhode Island School of Design on Creative Practice*. Wiley.
- [17] Vecchi, V. 2010. *Art and Creativity in Reggio Emilia*. London ; New York : Routledge.

