Bricoleur: A Tool for Tinkering with Programmable Video and Audio

Sean Hickey
Lifelong Kindergarten
MIT Media Lab
Cambridge, MA, USA
hisean@media.mit.edu

ABSTRACT

UPDATED—March 3, 2019. This paper presents *Bricoleur*, a new tool designed primarily for children to create rich dynamic audiovisual projects on mobile devices. Building off of technology developed for the Scratch programming language, Bricoleur allows users to capture video and audio media, then use visual programming blocks to quickly construct complex, interactive, and multilayered projects. The programmability of video and audio assets enables new creative possibilities that do not exist in traditional timeline-based video editing software nor in existing programming platforms. Drawing from Seymour Papert's assertion that all learning is an act of bricolage — i.e., a process of constructing knowledge while dialoguing back and forth with creative materials — we describe the design of an initial prototype of Bricoleur, which emphasizes *tinkerability* as a primary design goal.

CCS CONCEPTS

• **Human-centered computing** → **Interaction paradigms**; • **Social and professional topics** → **Children**; **Adolescents**; • **Applied computing** → **Media arts**; *Sound and music computing*; *Education*;

CHI'19 Extended Abstracts, May 4–9, 2019, Glasgow, Scotland UK
© 2019 Copyright held by the owner/author(s).
ACM ISBN 978-1-4503-5971-9/19/05.
https://doi.org/10.1145/3290607.3312919

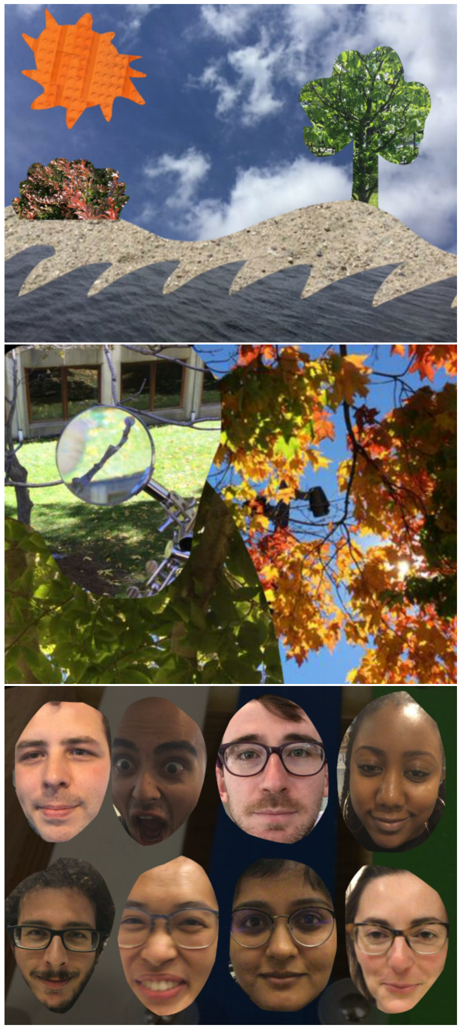
Figure 1: Example Projects. Top: Motion landscape. Middle: Story about autumn leaves. Bottom: Interactive musical instrument called "Xylo-face". When each face is tapped, the face opens their mouth synced with the sound of a xylophone note.

KEYWORDS

programmable video; programmable audio; programmable media; bricolage; tinkering; collage; learning; creativity; children

INTRODUCTION

Seymour Papert defined *bricolage* as “a style of organizing work that can be described as negotiational rather than planned in advance” [7]. 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. Moreover, as their work develops, so do the bricoleur’s ideas that drive the work forward. It is exactly this relationship between concrete materials and cognitive processes that lies at the root of Papert’s theory of constructionist learning. In particular, he asserted that learning is always a process of constructing knowledge in one’s head, and that constructing concrete artifacts in the world is one of the best ways to construct that knowledge. Moreover, Papert posited that the computer — due to its programmability — is a tool uniquely suited to bricolage, especially for children. When children 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 [6].

In this paper, we describe *Bricoleur*, a new tool designed primarily for children to create rich interactive projects on mobile devices by capturing and programming video and audio assets in a bricolage style of making. This project leverages the affordances of mobile hardware in combination with the blocks-based programming paradigm which powers the Scratch programming language to generate a new space of creative possibilities. Through hand-drawn video masking, users can quickly capture non-rectangular video assets and layer them on a common canvas. By capturing, layering, and programming a collection of assets, makers can create a motion landscape collage, a story about the changing autumn leaves, or an interactive musical instrument that can be played by tapping on each video asset, to name a few examples (Figure 1). [Note: throughout the paper we use the term "maker" in reference to the user of the tool to emphasize the creative process of using the tool]. Finally, we connect the notion of bricolage to that of *tinkering* and use Resnick and Rosenbaum’s tinkerability criteria to inform the design of the tool [8].

RELATED WORK

Bricoleur follows in the trajectory of tools like Scratch that enable children to create media-rich projects through blocks-based programming [9]. Scratch, however, does not contain any functionality for working with video assets and has minimal programmability of audio, so Bricoleur opens up new spaces of creative possibility and allows makers to play with notion of sequence and time in a way not possible through Scratch.

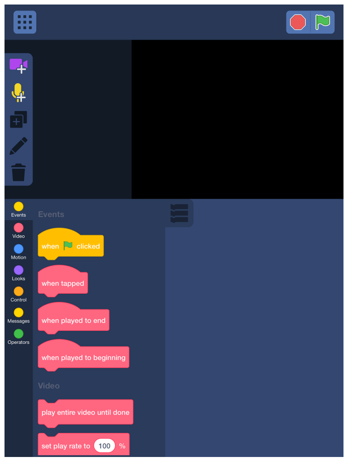

Figure 2: A blank project in Bricoleur

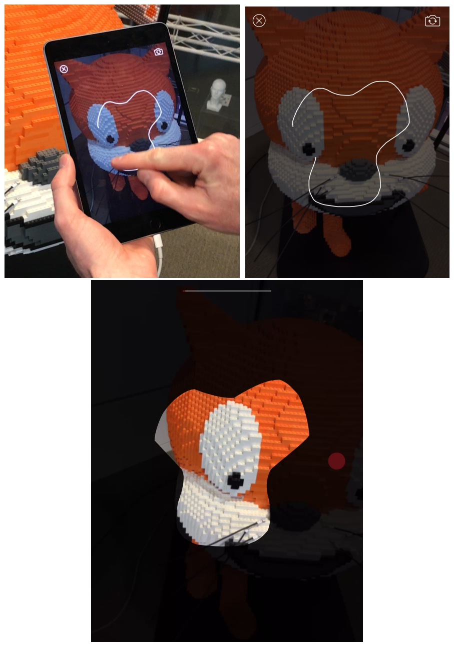

Figure 3: Hand-drawing a video mask to define the recording area

Other tools have explored ways for users to create media rich projects on mobile devices. MadPad used the audio sampler as inspiration for a virtual video sampling tool. Users could capture up to twelve video clips with sound, which were arranged in a grid, and then play them back by tapping on them [5]. Tools like LACES demonstrated the power of manipulating and compositing video media directly on tablet devices to construct video content in situ [3]. As the interactive musical instrument example shows (Figure 1), the flexibility provided by the programming blocks in Bricoleur allow for users to create their own MadPad-esque interactive sampler project within Bricoleur itself, in addition to traditional linear video content.

It is important to mention other media programming tools that enable the creation of interactive and dynamic work. Processing is one such example which allows users to manipulate media through programming [2]. Max is another powerful media programming environment and even has direct support for working with video streams [1]. While all of these media tools are designed with creativity in mind, only Scratch has the expressed goal of supporting learning. Bricoleur aims to combine the power of media programming tools with the approachability and blocks-based interface of Scratch to empower children to learn through playing with video, audio, and code.

CONCEPT & WORKFLOW

The current implementation of Bricoleur runs on tablet hardware and presents the maker with an editor interface consisting of a canvas at the top half of the screen and a blocks-based programming workspace on the bottom half (Figure 2). Tablets provide a rich context for the tool since the maker is able to utilize the onboard cameras and microphone to capture media directly into their project. Makers can also take the tablet into the world around them and capture images and sounds of their environment as well as use the front-facing camera to capture images of themselves. Makers begin creating a project by capturing either a video or audio asset.

When creating video, the maker is first presented with an interface that prompts them to draw the shape of the video they want to capture. By tracing a shape, the maker defines the area of pixels on the screen that will be captured during recording (Figure 3). After capture, the video asset appears in the canvas area of the editor. Using the programming blocks, the maker implements actions and behaviors of the asset (Figure 4). Examples include playing the video frames in sequence, jumping to a particular frame, changing the playback rate, and more (Figure 5). The maker can also program the asset to perform actions like moving across the canvas, rotating, and scaling. Programming control flow blocks (e.g., conditional "if" statements and loops) allow the user to create scripts with complex flow.

When capturing audio, the maker is presented with an interface that allows them to record an audio clip and see the visual representation of the waveform. After recording, the maker can create markers at locations in the audio clip which can later be used in the programming blocks (Figure 6).

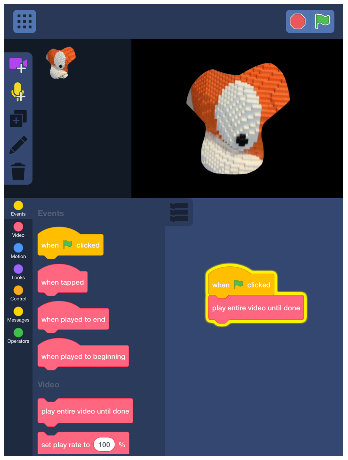

Figure 4: Video clip appears in editor after capture and can be programmed with blocks

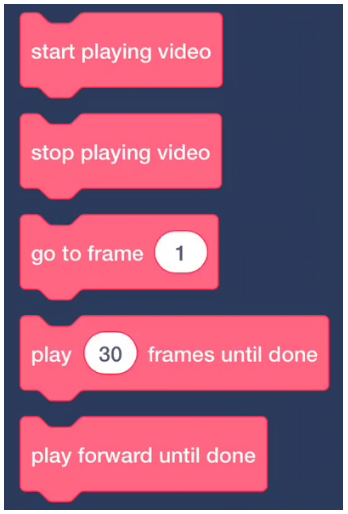

Figure 5: Example programming blocks for a video asset

Returning to the editor, the user can program their audio asset using blocks to perform actions like playing the entire clip from start to end or playing from one marker to another (Figure 7).

For both video and audio assets, blocks can be snapped together to create scripts, and each asset can have multiple scripts associated with it. Each asset can be programmed independently of the others since each script belongs to a particular asset (though there are blocks that assist in communicating between assets for advanced projects). By attaching a "hat block" (event block) to the top of a script, the maker can program an asset to respond to a variety of events including when the project starts (i.e., "when green flag is pressed"), when the video asset is tapped, or when it receives a message from another asset. By capturing, layering, and programming a collection of video and audio assets, makers can create a wide variety of interactive audiovisual projects, including the examples described in the introduction and shown in Figure 1.

DESIGN FRAMEWORK & CONSIDERATIONS

The notion of working in a bricolage style is closely related to the idea of "tinkering" as used by the maker community. Resnick and Rosenbaum provide a framework for designing tools that promote tinkering and working in bricolage methods [8]. Their framework provides three criteria that are useful in evaluating tools like Bricoleur. Namely, these are: (1) *immediate feedback*, (2) *fluid experimentation*, and (3) *open exploration*. We use these criteria to guide the development of the tool over time. Here, we highlight some examples of how each of these criteria have come into play during the development of the initial prototype.

Immediate feedback refers to the idea that the results of any change to a project should be reflected as quickly as possible. Building off of the work that Scratch has done in this area, code blocks execute immediately when they are tapped and any script can be modified while the program is running. Which is to say that there is no rendering step, unlike many audiovisual authoring tools. When code executes, the results are rendered in realtime immediately to the canvas. For this reason, there is no distinction between editing mode and playback mode.

Fluid experimentation refers to the way in which the tool facilitates an iterative making process. Moreover, it means that it should be easy for a maker to get started creating a project and that they should be able to grow their project in complexity over time. To facilitate getting started easily, video and audio capture is done directly within the tool and each asset can be activated through a single code block (e.g., in the case of audio, the first block presented to the maker is "play from start to end" which is the most common use case of an audio asset). The hand-drawn video masking also allows for users to quickly create video artifacts that would be difficult or impossible to create in other tools. To grow a project in complexity, the maker can continue to add and program more video and audio assets or further develop their existing programming scripts. As mentioned earlier, various blocks are

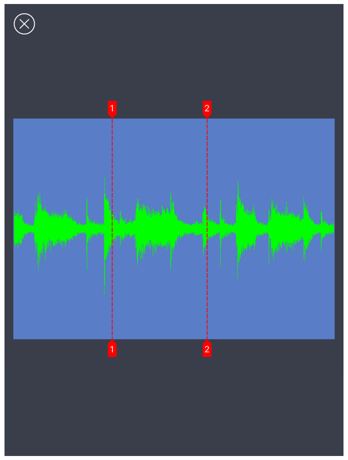

Figure 6: A captured audio asset. Red lines indicate user specified markers.

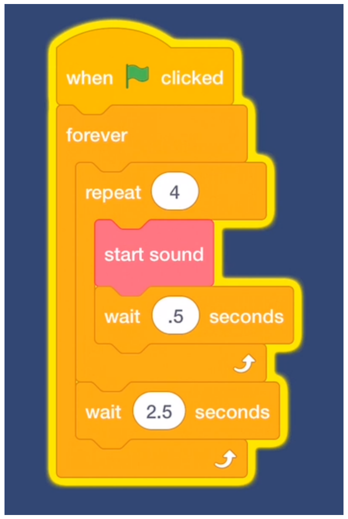

Figure 7: A block stack for an audio asset that loops and sequences the sound.

provided for users to program more complex actions as their projects develop (e.g., sending messages between assets).

Open exploration refers to the idea that there should be a wide space of possibilities for makers, who will naturally come to the tool with a diverse set of interests. As shown in the examples in the introduction, Bricoleur allows for a variety of project types (art, stories, music, interactive works, etc.). The choice of video as a computational material also facilitates open exploration in that it is a medium uniquely positioned to combine with other materials. Video can capture the motion of dance, the color of paint, the semantics of text, and the acoustics of sound, to name a few. Historically, many of the earliest video artists worked primarily in other media — performance, music, dance, etc. [4]. Using video as a primary medium, makers can capture their ideas and interests through the camera.

As development on the tool continues, these criteria will play an important role in guiding design decisions.

EARLY RESULTS: LEARNING & POWERFUL IDEAS

In thinking about initial validation of the tool, we return to Papert's notion that working with computers in a bricolage/tinkering style can give makers access to what he called "powerful ideas." In his work with the LOGO programming language, he asserted that programming the turtle allows access to powerful mathematical ideas not typically offered to children [6]. Early playtesting of Bricoleur suggests that the blocks-based programming approach to manipulating media provides a unique entry point for makers to engage with powerful ideas about time, synchrony, and asynchrony. Common tasks when making a project (e.g., putting two video clips in sequence, or having a sound and video start playing at the same time) require the maker to consider ideas about timing, clip length, play rate, and more when snapping programming blocks together. Furthermore, the various notions of time within the app (i.e., seconds, frames, and markers) encourage the maker to think about time in different scales and units, as well as the relationship between those scales and units, appropriate to the context of their project. As testing continues, we are also curious to see how using the tool may impact makers' notions of media creation broadly.

FUTURE WORK

There are several directions for further development of Bricoleur. To support ease of getting started, the on-boarding experience will be redesigned to allow makers to create projects as quickly as possible. In addition, there is a rich design question to explore around which blocks provide the most expressive potential for makers while keeping the palette of blocks relatively small, as well as which blocks afford makers most direct access to the types of powerful ideas described in the previous section. This may mean refining current block definitions, removing some existing blocks, as well as inventing new blocks for users to manipulate audio and video media. Relatedly, it would be possible to create blocks that would allow users to interact with sensors on the tablet, e.g., the accelerometer and compass, which would enable projects to react to device motion and orientation. Finally, there is an outstanding question of how best to allow users to capture and share their projects with others. User playtesting with kids in the spring of 2019 will help guide future iterations of the tool.

Ultimately, the goal is to push the boundaries of programmable media for children which will open up new spaces and contexts for researchers to investigate how children can learn in playful ways with new and evolving technologies.

ACKNOWLEDGEMENTS

Thanks to the Lifelong Kindergarten group for their inspiration, ideas, playtesting, and feedback on early versions of this project.

REFERENCES

[1] 2019. Cycling ’74. https://cycling74.com/
[2] 2019. Processing. https://processing.org/
[3] Dustin E R Freeman, Stephanie Santosa, Fanny Chevalier, Ravin Balakrishnan, and Karan Singh. 2014. LACES. In *the 32nd annual ACM conference*. ACM Press, New York, New York, USA, 1207–1216.
[4] John G Hanhardt (Ed.). 1986. *Video Culture*. Layton, Utah : G.M. Smith, Peregrine Smith Books, in association with Visual Studies Workshop Press.
[5] Nick Kruge and Ge Wang. 2011. MadPad. *Collected Work: Proceedings of the International Conference on New Interfaces for Musical Expression. Published by: Oslo, Norway: Universitetet i Oslo/University of Oslo, 2011. Pages: 185-190. (AN: 2011-28094).* (2011).
[6] Seymour Papert. 1993. *Mindstorms*. New York : Basic Books.
[7] Seymour Papert. 1993. *The Children’s Machine*. New York : BasicBooks.
[8] Mitchel Resnick and Eric Rosenbaum. 2013. Designing for Tinkerability. In *Design, Make, Play*, Margaret Honey and David Kanter (Eds.). 163–181.
[9] Mitchel Resnick, Brian Silverman, Yasmin Kafai, John Maloney, Andrés Monroy-Hernández, Natalie Rusk, Evelyn Eastmond, Karen Brennan, Amon Millner, Eric Rosenbaum, and Jay Silver. 2009. Scratch. *Commun. ACM* 52, 11 (Nov. 2009), 60–67.