Cartoonimator: A Low-cost, Paper-based Animation Kit for Computational Thinking

Krithik Ranjan
krithik.ranjan@colorado.edu
ATLAS Institute, University of Colorado Boulder
Boulder, Colorado, USA

Peter Gyory
peter.gyory@colorado.edu
ATLAS Institute, University of Colorado Boulder
Boulder, Colorado, USA

Michael L Rivera
mrivera@colorado.edu
ATLAS Institute, University of Colorado Boulder
Boulder, Colorado, USA

Ellen Yi-Luen Do
ellen.do@colorado.edu
ATLAS Institute, University of Colorado Boulder
Boulder, Colorado, USA

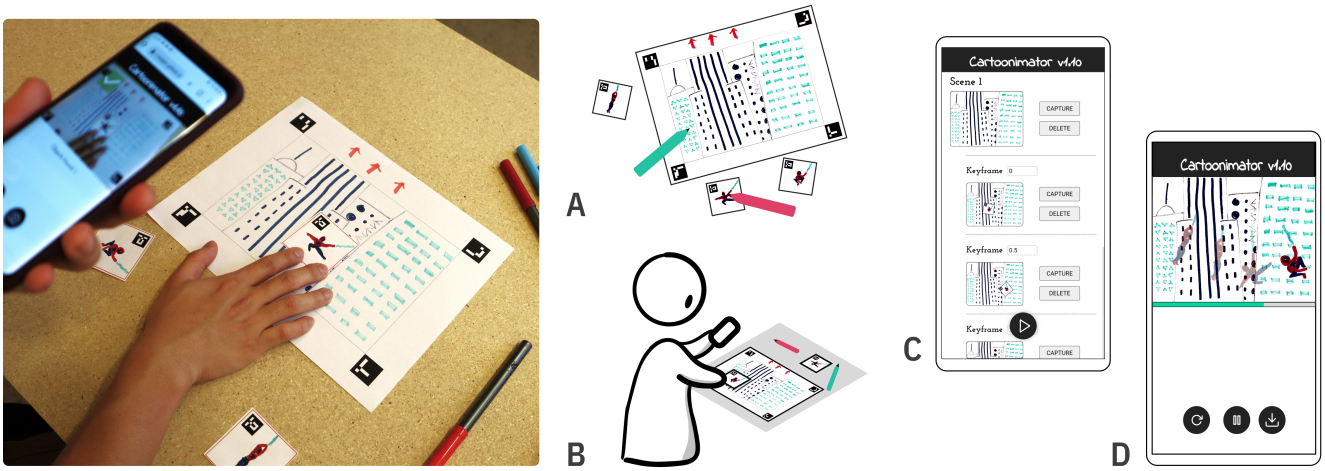

Figure 1: Creating an animation with Cartoonimator: (A) Drawing the scene background and characters; (B) Capturing the scene and keyframes, shown in the app in (C); (D) Playing the video.

ABSTRACT

Computational thinking has been identified as an important skill for children to learn in the 21st century, and many innovative kits and tools have been developed to integrate it into children's learning. Yet, most solutions require the use of devices like computers or other expensive hardware, thus being inaccessible to low-income schools and communities. We present Cartoonimator, a low-cost, paper-based computational kit for children to create animations and engage with computational thinking. Cartoonimator requires only paper and a smartphone to use, offering an affordable learning experience. Children can draw the scenes and characters for their animation on the paper, which is printed with computer vision markers. We developed the mobile web app to provide an interface to capture keyframes and compile them into animations. In this paper, we describe the implementation and workflow of Cartoonimator, its deployment with children at a local STEAM event, and a planned evaluation for the kit.

IDC '23, June 19–23, 2023, Chicago, IL, USA

ACM ISBN 979-8-4007-0131-3/23/06.
https://doi.org/10.1145/3585088.3593886

CCS CONCEPTS

• **Human-centered computing** → **Collaborative and social computing systems and tools**; • **Social and professional topics** → **Computational thinking**; **K-12 education**.

KEYWORDS

tangible programming, paper computing, computational thinking, animation

ACM Reference Format:
Krithik Ranjan, Peter Gyory, Michael L Rivera, and Ellen Yi-Luen Do. 2023. Cartoonimator: A Low-cost, Paper-based Animation Kit for Computational Thinking. In *Interaction Design and Children (IDC '23), June 19–23, 2023, Chicago, IL, USA.* ACM, New York, NY, USA, 5 pages. https://doi.org/10.1145/3585088.3593886

1 INTRODUCTION

Computational Thinking [25] is viewed as a fundamental skill that children should learn. Computational concepts like algorithmic thinking, deconstruction, abstraction, and debugging, not only help children work well with computers but also "understand the world in new ways" [27]. Studies show that children as young as kindergarteners can grasp these computational concepts [22] and exposure to them can improve their self-efficacy towards STEM fields in the future [6]. To help children begin thinking computationally from an early age, programming has become an important subject in K-12 education [26]. A variety of approaches to programming education range from graphical coding environments to computational toys and activity kits [27].

Graphical programming languages (e.g. Scratch [19]) offer a low barrier to entry for children through a drag-and-drop programming interface by removing the need to understand complex code syntax. However, these programming environments are designed for a single-user workflow, and are therefore less effective and engaging in settings where not every student has their own computer [12, 21]. Tangible programming approaches attempt to overcome this through shared interaction with physical objects that represent programming concepts [10]. Unfortunately, these tools often involve embedded electronics, additional computers, and high-end image processing techniques, which make them expensive and unreliable for a classroom environment [9].

The cost and resources needed for both graphical and tangible programming tools make them inaccessible to economically-disadvantaged schools and communities. As an approach towards solving this problem, paper programming has emerged as a subset of tangible programming which relies on everyday materials like paper or cardboard, and mobile-based computer vision algorithms to offer an affordable and effective learning experience. Groups of students in classrooms can work on these paper programming activities with a shared smartphone, which further increases the accessibility [21].

In this work, we present Cartoonimator — a low-cost computational kit that enables children to create animations using paper. Our kit builds on the paper programming paradigm of using computer vision on smartphones to use paper as a medium for computing. Cartoonimator enables children to explore computer animations while engaging in computational thinking concepts (e.g. sequences and parallelism),practices (e.g. being incremental and iterative, testing and debugging) and perspectives (e.g. expressing), as defined in the computational thinking framework by Brennan and Resnick [4]. The Cartoonimator kit consists of (1) paper enhanced with computer vision (CV) markers on which children can draw and paint parts of their animation, and (2) a simple smartphone app that they can use to capture frames and view their animation. Cartoonimator features a technique called *keyframing* in which animators specify starting and ending points (*keyframes*) of a smooth transition and the software interpolates between them to create a complete animation. The following sections describe the related work, the implementation and workflow of the system, its deployment with children at a local STEAM festival, and a planned extensive evaluation of Cartoonimator.

2 BACKGROUND

Over the decades, various approaches have been used to promote computational thinking among children, one of the earliest being the LOGO programming language that enables learners to create graphics by giving commands to move a turtle on their screen [17]. In recent years, Scratch [19] and ScratchJr [7] have been two of the popular block-based graphical programming languages for children to create games and interactive animations on the web or using apps. Although Graphical programming interfaces provide a low technical barrier for learners, they are limited to individual on-screen interaction. In contrast, tangible programming tools enable multiple students to collaborate using a physical computational interface (e.g. Tern [12]). These can be electronic robotic toys like Cubetto [24] that children can program using tangible blocks, or non-electronic passive learning tools like board games (Robot Turtles [23]) or storybooks (Hello Ruby [14]).

To make these tangible programming tools more accessible, researchers have turned to everyday materials like paper as the interface and mobile devices as the computational component. For example, the Kart-ON project [21] offers a paper-based programming environment for creative coding applications using scannable programming cards for the p5.js framework [15]. Roberto [11] takes a storybook approach in which readers can create their own animated stories by adhering code stickers to the book and scanning them. With Draw2Code [13], children can create gesture-controlled AR animations for a single drawing by arranging and scanning paper coding blocks.

Cartoonimator differs from these paper programming tools by shifting focus away from traditional block-based programming — instead of describing the animation with coding blocks, in Cartoonimator the child positions their character drawing directly on top of their animation background to capture keyframes and build the animation. This engages their spatial knowledge domain in children's learning, which has been found to rarely be leveraged in the design of programming tools [22]. Moreover, Cartoonimator enables children to draw their own scenes and characters, and create longer and more complex animations than other paper-based animation kits (e.g., Roberto [11] and Draw2Code [13]). Cartoonimator also supports creating multiple independently animated characters, which engages children with the computational concept of parallelism.

3 CARTOONIMATOR OVERVIEW

Cartoonimator is a computational kit for children to create animations from their drawings. We chose drawing as a medium for computing in Cartoonimator, an activity that children already engage in and derive meaning from [5]. Moreover, creating animations with Cartoonimator encourages a child to see computation as a means of self-expression to tell stories, which can stimulate creative play and communication [18], and computational thinking [4].

We developed Cartoonimator as a paper-based kit that uses shared smartphones instead of individual computers. Further, we limited the smartphone interaction so that it is easier to share phones it in a classroom or for a child to borrow one from their parents. In line with Resnick's design principles for creative thinking [20], we designed the kit to be easy to interact with, while providing support to create complex animations.

Cartoonimator is centered on translating the *keyframing* workflow of traditional 2D computer animation software (e.g. Macromedia Flash [2] and Adobe Animate [1]) in which animators create digital characters or *sprites* and position them on top of the animation background (*scene*) to specify different *keyframes* and their time occurrence in the animation (*timestamp*). Similar to these tools, Cartoonimator enables children to create complete animations using *keyframing*, while learning about animation concepts such as sprite, scene, keyframe, and timestamp. As a child creates animations with Cartoonimator, we expect them to engage with the computational concepts of sequences and parallelism, which are core to the keyframing animation process. While creating animations with the kit, children apply computational practices and acquire the expressive computational perspective as defined by Brennan and Resnick in their framework. [4].

3.1 System Elements

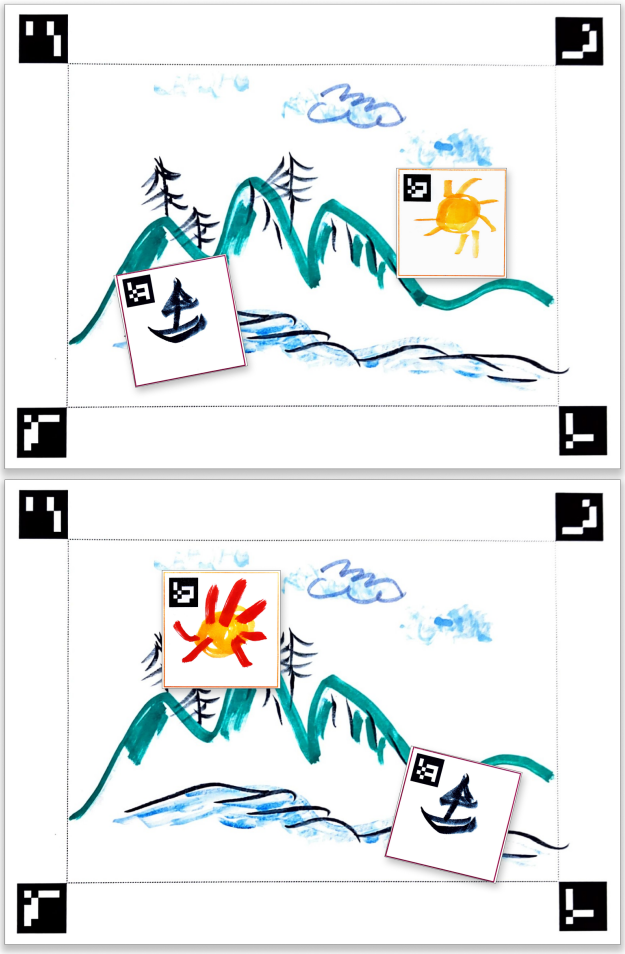

Figure 2: Placing multiple object cards on the scene over two keyframes.

The Cartoonimator system is made up of (1) paper with printed CV markers, and (2) a smartphone app. The paper component of the kit can be further divided into (1a) scene sheets on which children can draw the background for their animation (Fig. 1A), and (1b) object card templates to draw sprites (Fig. 1A). We use the ArUco computer vision (CV) markers [8] to recognize and process the drawings. Each scene sheet has a blank drawing region surrounded by four ArUco markers (one at each corner) which enable a child to capture the scene from any angle – the app uses the markers to flatten and crop to the drawing using a CV technique called *homography* (1C). The object cards are small square paper sheets that have one ArUco marker at the top-left corner and a colored border. Using CV algorithms, Cartoonimator subtracts the background and the marker from these object cards to place them on top of the scene. A child can add multiple sprites to the animation using different object cards, or use the same object card to draw different forms of the same sprite (Fig. 2).

The Cartoonimator application is a mobile website that enables the children to capture the scenes and the keyframes, assign timestamps, and view the complete animation (1C,D). When designing Cartoonimator, we sought to make the application's interface simple to promote use by children and support sharing in a classroom. The application can be opened on most mobile browsers (tested on Chrome, Edge, and Safari), and does not require any additional installation. The app utilizes OpenCV [3] and ArUco [8, 16] libraries in JavaScript to perform the ArUco marker recognition and construction and rendering of the video.

3.2 Workflow

Fig. 1 describes the workflow of using Cartoonimator. To create an animation with Cartoonimator, the child first draws a background for their scene on one of the set drawing sheets and one or more character sprites on the object cards (Fig. 1A). Once all the drawings are ready, they can begin capturing the scenes with the app (Fig. 1B). First, the child adds a scene to their animation. Each scene is made up of a background and keyframes on that background. The keyframes are used to specify the position and orientation of characters at different times during the scene and therefore have an associated timestamp. For each keyframe, they place the object cards on top of the scene sheet to specify their position, orientation, and timestamp, which will change over different keyframes. After capturing a few keyframes, they can play their created animation and download it to the device. To make a more complex animation, they can add more scenes and keyframes.

4 CARTOONIMATOR IN THE WILD AT STEAM FEST

We deployed Cartoonimator with children at a local STEAM Fest where 20+ children aged between 4 and 13 tried out the kit and no data was formally collected. Children were given a brief overview of how to use Cartoonimator and then set free to explore. Attendees only spent a few minutes at each booth, so in addition to the blank scene sheets and object cards, we had pre-printed scenes and characters for children to work with. Many children were excited to make an animation with these printed cartoon characters familiar

Figure 3: Cartoonimator at STEAM Fest: (A) A child setting up animation with multiple objects, (B) Viewing the created animation, (C) A sprite spanning two object cards, (D) One child helping their sibling use Cartoonimator.

to them, while some drew their own scenes and characters, and a few modified the pre-printed characters.

Most of the children were able to independently create full animations (at least 5 keyframes long), but some needed guidance with using the app and performing all the steps in order to create the animation. Some children were with their siblings or friends, and after we taught one, they were easily able to teach the other and make an animation together (Fig. 3D). One collaboration strategy we saw was to divide their roles between 'picture-taker' and 'character-mover'. This made it easier for them to interact with the phone app as both children were not touching the screen at the same time. Some children also came back to play with Cartoonimator (one even returned three times) — they remembered how to use the kit and returned with ideas for more complicated animations. For example, one wanted to create a bigger character than the object cards permitted, so they inventively put two cards together to draw and move the character (Fig. 3D).

We observed children indirectly learning computational concepts of sequences and parallelism (when they used multiple characters like in Fig.3). Further, they created their animation incrementally and often went back to recapture the keyframe or reset the timestamp to adjust the animation, thus engaging in the computational practices of being incremental and iterative, and testing and debugging.

The main obstacle to working with Cartoonimator we found from this deployment was the concept of timestamps. It was hard for the children to see how the timestamp number for each keyframe determines the keyframe's position in time in the animation. Some of them understood the numbers as a way to make the animation go faster or slower and changed the timestamps to adjust the speed

of the characters moving on the scene. Others, however, needed guidance on what numbers they could fill in for the timestamps to obtain their intended animation. For future updates to the system, we will devise more intuitive ways to introduce keyframe times to children, such as an interactive timeline.

5 CONCLUSION AND FUTURE WORK

In this paper, we present our preliminary work on Cartoonimator, a paper-based animation kit for computational thinking. By developing this kit to need only some printed paper and a smartphone, we aim to stay within the economic constraints faced by many. Creating with Cartoonimator exposes young animators to computer animation concepts, as well as computational thinking concepts and practices.

Our experience with kids trying out Cartoonimator at the STEAM Fest has revealed promise in terms of engagement and usability. Moving forward, we plan to conduct a formal evaluation of the kit aimed toward answering the following research questions — (1) *Does Cartoonimator teach children about the target animation concepts of sets, scenes, sprites, and keyframes?* (2) *Do children develop computational thinking concepts of sequences and parallelism, and computational practices of being incremental and iterative, testing and debugging, and reusing and remixing when interacting with Cartoonimator?* We will use the framework put forward by Brennan and Resnick [4] and specifically the techniques of project analysis and artifact-based interviews to evaluate the kit along these questions. We are excited to see how Cartoonimator improves kids' computational learning in an affordable and engaging way, and we hope this kit can be a step toward making computing education more accessible to kids in low-income schools and communities.

ACKNOWLEDGMENTS

This research is sponsored in part by the U.S. National Science Foundation through grant IIS-2040489.

REFERENCES

[1] Adobe.com. n.d.. Adobe Animate. Retrieved December 15, 2022 from https://www.adobe.com/products/animate.html
[2] Adobe.com. n.d.. Macromedia - Flash. Retrieved December 15, 2022 from https://www.adobe.com/support/documentation/en/flash/fl8/releasenotes.html
[3] Gary Bradski. 2000. The openCV library. *Dr. Dobb's Journal: Software Tools for the Professional Programmer* 25, 11 (2000), 120–123.
[4] Karen Brennan and Mitchel Resnick. 2012. New frameworks for studying and assessing the development of computational thinking. In *Proceedings of the 2012 annual meeting of the American educational research association, Vancouver, Canada*, Vol. 1. 25.
[5] Margaret L Brooks. 2017. *Drawing to learn.* Springer.
[6] Yu-Hui Ching, Yu-Chang Hsu, and Sally Baldwin. 2018. Developing computational thinking with educational technologies for young learners. *TechTrends* 62, 6 (2018), 563–573.
[7] Louise P Flannery, Brian Silverman, Elizabeth R Kazakoff, Marina Umaschi Bers, Paula Bontá, and Mitchel Resnick. 2013. Designing ScratchJr: Support for early childhood learning through computer programming. In *Proceedings of the 12th international conference on interaction design and children*. 1–10.
[8] Sergio Garrido-Jurado, Rafael Muñoz-Salinas, Francisco José Madrid-Cuevas, and Manuel Jesús Marín-Jiménez. 2014. Automatic generation and detection of highly reliable fiducial markers under occlusion. *Pattern Recognition* 47, 6 (2014), 2280–2292.
[9] Sidhant Goyal, Rohan S Vijay, Charu Monga, and Pratul Kalita. 2016. Code bits: an inexpensive tangible computational thinking toolkit for K-12 curriculum. In *Proceedings of the TEI'16: Tenth International Conference on Tangible, Embedded, and Embodied Interaction*. 441–447.
[10] Michael Horn and Marina Bers. 2019. Tangible computing. *The Cambridge handbook of computing education research* 1 (2019), 663–678.
[11] Michael S Horn, Sarah AlSulaiman, and Jaime Koh. 2013. Translating Roberto to Omar: computational literacy, stickerbooks, and cultural forms. In *Proceedings of the 12th International Conference on Interaction Design and Children*. 120–127.
[12] Michael S Horn and Robert JK Jacob. 2007. Designing tangible programming languages for classroom use. In *Proceedings of the 1st international conference on Tangible and embedded interaction*. 159–162.
[13] Hyejin Im and Chris Rogers. 2021. Draw2Code: Low-Cost Tangible Programming for Creating AR Animations. In *Interaction Design and Children*. 427–432.
[14] Linda Liukas. n.d.. Hello Ruby. Retrieved December 15, 2022 from https://www.helloruby.com/
[15] Lauren Lee McCarthy. n.d.. p5.js. Retrieved December 15, 2022 from https://p5js.org/
[16] Juan Mellado. 2011. js-aruco | GitHub. Retrieved March 10, 2023 from https://github.com/jcmellado/js-aruco
[17] Seymour A Papert. 2020. *Mindstorms: Children, computers, and powerful ideas*. Basic books.
[18] Louise Phillips. 2000. Storytelling: The seeds of children's creativity. *Australasian journal of early childhood* 25, 3 (2000), 1–5.
[19] Mitchel Resnick, John Maloney, Andrés Monroy-Hernández, Natalie Rusk, Evelyn Eastmond, Karen Brennan, Amon Millner, Eric Rosenbaum, Jay Silver, Brian Silverman, et al. 2009. Scratch: programming for all. *Commun. ACM* 52, 11 (2009), 60–67.
[20] Mitchel Resnick, Brad Myers, Kumiyo Nakakoji, Ben Shneiderman, Randy Pausch, Ted Selker, and Mike Eisenberg. 2005. Design principles for tools to support creative thinking. (2005).
[21] Alpay Sabuncuoglu and T Metin Sezgin. 2022. Kart-ON: An Extensible Paper Programming Strategy for Affordable Early Programming Education. *Proceedings of the ACM on Human-Computer Interaction* 6, EICS (2022), 1–18.
[22] Amanda Strawhacker and Marina Umaschi Bers. 2019. What they learn when they learn coding: investigating cognitive domains and computer programming knowledge in young children. *Educational Technology Research and Development* 67, 3 (2019), 541–575.
[23] Inc. ThinkFun. n.d.. Robot Turtles: A Coding Board Game for Little Programmers. Retrieved December 15, 2022 from https://www.thinkfun.com/products/robot-turtles/
[24] Primo Toys. n.d.. Cubetto: A Toy Robot Teaching Kids Code & Computer Programming. Retrieved December 15, 2022 from https://www.primotoys.com/
[25] Jeannette M Wing. 2006. Computational thinking. *Commun. ACM* 49, 3 (2006), 33–35.
[26] Jeannette M Wing. 2016. Computational Thinking, 10 years later. Retrieved March 3, 2023 from https://www.microsoft.com/en-us/research/blog/computational-thinking-10-years-later/
[27] Junnan Yu and Ricarose Roque. 2019. A review of computational toys and kits for young children. , 17–36 pages.