

Software Tools for Elementary Math Education: Animated Mathematical Proofs

by

Timothy M. Mwangi

S.B. Electrical Engineering and Computer Science

Massachusetts Institute of Technology, 2008

Submitted to the Department of Electrical Engineering and Computer Science
in partial fulfillment of the requirements for the degree of Master of Engineering in Electrical
Engineering and Computer Science at the

ARCHIVES

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

September 2013

© Massachusetts Institute of Technology 2013. All rights reserved.

Author

.....
Department of Electrical Engineering and Computer Science
August 29, 2013

Certified by

.....
Dr. Kimberle Koile
Research Scientist, MIT CECI
Thesis Supervisor
August 29, 2013

Accepted by

.....
Prof. Albert R. Meyer
Chairman, Masters of Engineering Thesis Committee

This page intentionally left blank

Software Tools for Elementary Math Education: Animated Mathematical Proofs

By

Timothy M. Mwangi

Submitted to the Department of Electrical Engineering and Computer Science on August 30, 2013 in Partial Fulfillment of the Requirements for the Degree of Master of Engineering in Electrical Engineering and Computer Science

ABSTRACT

The National Council of Teachers of Mathematics [6] has identified the learning of proofs as a critical goal for students from pre-kindergarten through grade 12 (p. 56). A proof for elementary students is not the highly structured mathematical argument seen in high school algebra classes. It is, however, a rational mathematical argument created by students using the appropriate vocabulary for their level of understanding. To aid students in learning to create mathematical proofs software that enables them to create simple animations is invaluable. This thesis looks at the characteristics, design, testing and evaluation of such software. An initial design is presented and the feedback gained from testing its implementation in a class setting is discussed along with the changes that were required to improve the software in light of the feedback. A comparison is then made between the final implementation of the software and other similar programs. The results indicate that the software enables students to create, share and discuss mathematical proofs in the form of simple animations.

Thesis Supervisor: Dr. Kimberle Koile

Title: Research Scientist, MIT CECI

This page intentionally left blank

Acknowledgements

I would like to express my gratitude to my supervisor, Dr. Kimberle Koile, for her mentorship, guidance and leadership. This project would not have been possible without her. It was an honor to work with her, and I believe that we owe much of the success of this project to her.

I would also like to thank Andee Rubin at TERC, especially for helping me understand the nuances of developing software for teaching children mathematics. She too played a critical role in the success of this project, and the feedback and analysis that I received from her made this project better in many ways.

The technical assistance that I received from Steve Chapman was invaluable. I am greatly indebted to him for always being available to answer the questions I had. He was selfless and sacrificed his personal time to assist me.

This page intentionally left blank

Contents

1	Introduction	8
2	Software Infrastructure	13
3	Design Objectives	16
3.1	Simplicity.....	16
3.2	Structure.....	16
4	Software Design	17
4.1	Extensions to CLP.....	17
4.2	Designs.....	20
4.2.1	Design 1, Class 1.....	20
4.2.2	Design 2, Class 1.....	21
	<u>Improvements</u>	21
4.2.3	Design 3, Class 1.....	23
	<u>Improvements</u>	23
4.2.4	Design 4, Class 2.....	24
	<u>Improvements</u>	24
4.2.5	Final Design, To Be Tested In Fall 2013	25
	<u>Improvements</u>	25
5	Student Work	27
5.1	Class 1	27
5.2	Class 2	36
6	Related Work	41
7	Future Work	43
7.1	Audio.....	43
7.2	Better Visual Cues: Flashing Buttons	43
7.3	Greater Leeway: Allow Mistakes, Cut a Single Cube Out of an Array.....	43
7.4	Even Smaller Histories	44
7.5	Create Animations Retroactively.....	44
8	Conclusion	46
9	References	47
	Appendix	48

1 Introduction

The National Council of Teachers of Mathematics [6] has identified the learning of proofs as a critical goal for students from pre-kindergarten through grade 12 (p. 56). What does a proof look like for elementary students? It is not the highly structured proof seen in high school algebra classes. It is, however, a logical argument about mathematical relationships, but created by students using vocabulary and representations that are appropriate for the students' level of understanding. There is evidence that elementary students are capable of creating proofs about mathematical relationships using representations of mathematical quantities and that doing so deepens their understanding of mathematical content and prepares them for a later transition to algebra [9]. In learning about arithmetic operations, for example, instead of just focusing on facts such as $2 \times 10 = 20$, presenting students with problems such as $2 \times 10 = 20$ and $3 \times 10 = 30$ and asking them about a generalization for $N \times 10$ in terms of arrays, helps students begin to see general patterns about operations on numbers. Once an abstraction from specific numbers to more general groups of numbers is made, a deeper understanding of the mathematical operation results.

As an example of a proof, consider the question of showing that reordering the factors in a multiplication equation does not change the answer, i.e., showing that $m \times n = n \times m$. Elementary students may be asked to show this proof by choosing any two numbers and representing the multiplication using an array. A student may use cubes to build a physical representation of an array and rotate the cubes to show the transition from beginning state to end state. He has no physical artifact, however, that enables him or his teacher to see evidence of his work. He also may draw a starting state and an ending state and describe orally or in writing the transition from one state to the other.

This thesis presents a new way for elementary students to demonstrate their proofs: by creating an animation that shows the manipulation of mathematical representations. With such a software tool, students are freed from having to create static “before and after” pictures, and they have an artifact of their work that can be easily “replayed” and shared with students and the teacher. Sharing of the proof is key, as being able to convince oneself and others of a claim is what distinguishes a proof from reasoning.

1.1 Examples

Example 1

Consider again the example of showing that $m \times n = n \times m$. Shown in Figure 1-1 is a sample of student work on paper from a third grade class.

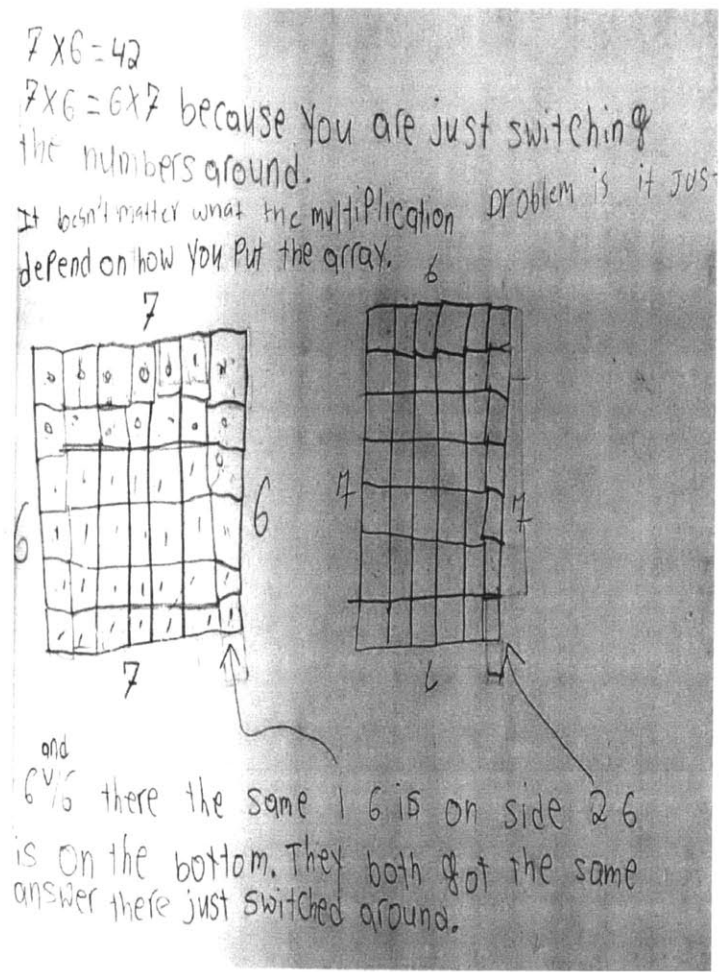


Figure 1-1. A student's answer given on paper

The student work shown in the figure above captures many of the differences between an answer on paper and an animation. On paper, students will usually draw an initial state and a final state. They will then use arrows to indicate movement and words to explain what the "movement" represents. This approach requires students to create twice as many objects as they need in a proof. Sometimes pen and paper may suffice but it is a less expressive medium that requires students to do more work and limits the ideas that can be communicated. On the other hand, an animation allows a student to communicate ideas that he is unable to put into words. Creating an animation is also easier and more effective. Using the example in the picture above, the student could simply have created a 7×6 array and rotated it to show that $7 \times 6 = 6 \times 7$. By sharing the animation, his peers can watch the animation and actually see what he is describing. This process bypasses the need to convert the idea of rotation into words and then have the other students and the teacher visualize what those words mean.

Shown in Figure 1-2 is an example of the beginning state of one such animation that could be created using the software tool developed in this thesis. The student could create an array using

an Array Tool on the top command bar, then record rotating the array using animation control buttons. She can then send the animation to the teacher, and the teacher can project and play the animation for the class. The process of creating and manipulating mathematical representations and creating animations is described in this thesis.

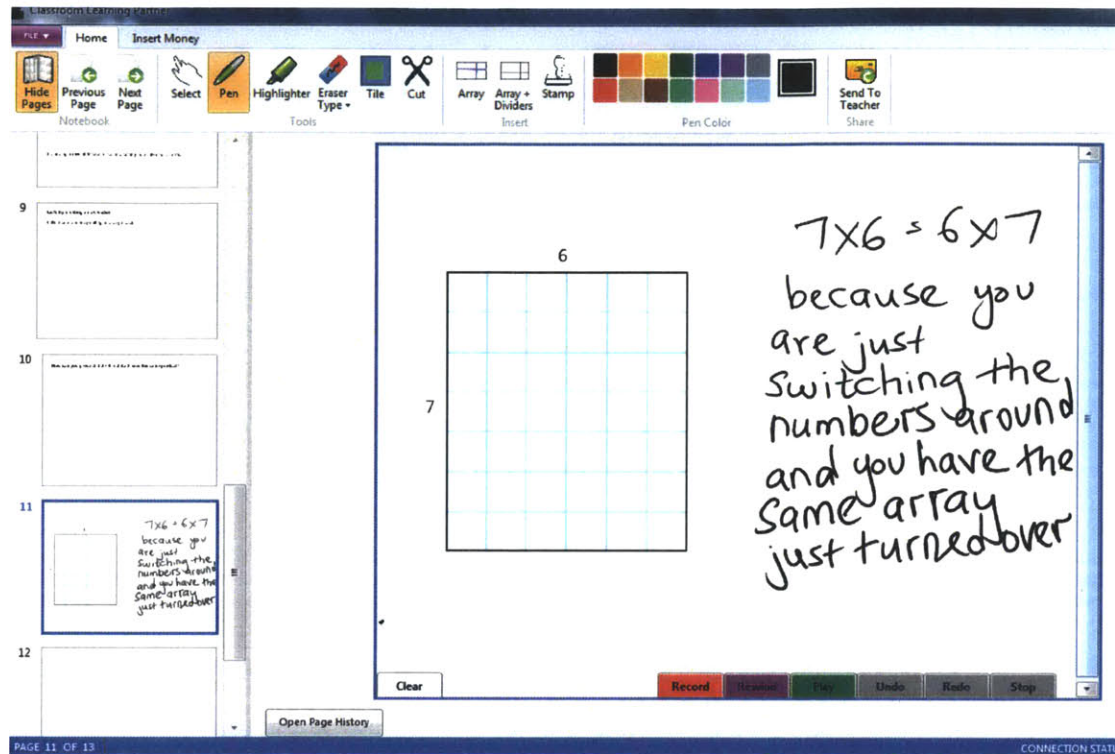


Figure 1-2. Example of starting state for animation for showing $7 \times 6 = 6 \times 7$

Example 2

Consider the proof for what happens when multiplying two numbers and one of the numbers is increased by one. Shown in Figure 1-3 is student work from the same third grade class. Note that the student has done a good job of explaining what happens, but has had to create before and after pictures to attempt to show the addition of a row or column.

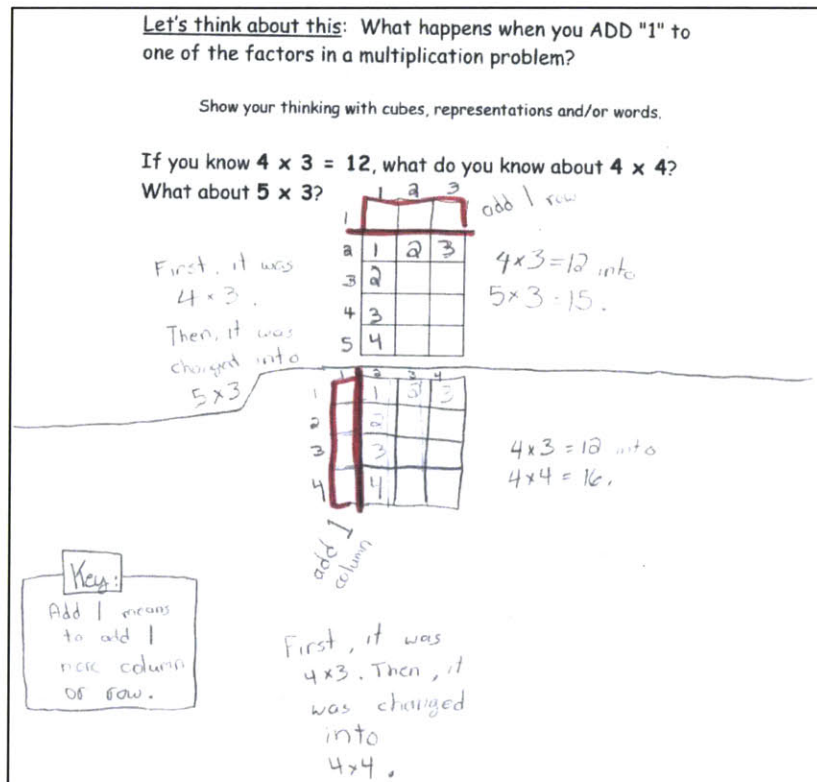


Figure 1-3. Student mathematical proof work on paper

Using the animation software created for this thesis, the student could create an animation that shows the initial array, and the acts of creating a new row (or column) and moving that row (or column) to align with the original array. The student could add text to further explain what the animation is showing. Shown below in Figure 1-4 is an example of this same student's work using the animation tool to show what happens when subtracting one from a factor in a multiplication problem. (This work is described in more detail in Section 5.)

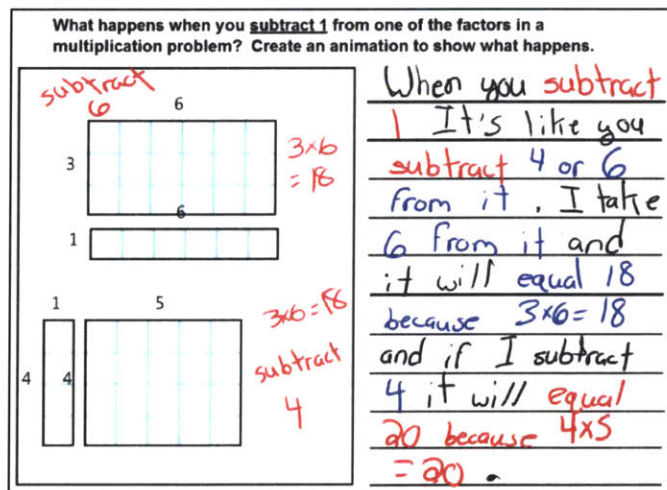


Figure 1-4. Ending state of a student's mathematical proof work using our animation software

1.2 Outline

Chapter 2 describes the underlying software that provides the infrastructure for the animation software described in the examples above. Chapter 3 presents the central design objectives of the animation software while Chapter 4 presents the initial design and the changes that were made during in-class testing. Chapter 5 looks at students work in in-class testing and Chapter 6 looks at related software products. Chapter 7 addresses future improvements that can be made to our software and Chapter 8 notes the impact that the animation software has on learning. Finally, the Appendix gives specific details about the software classes, interfaces and methods in the initial design.

2 Software Infrastructure

The animation tool described in this thesis is an extension to an existing classroom interaction system called Classroom Learning Partner (CLP) [4, 7]. CLP runs on tablet computers that communicate with each other via a wireless network. The teacher has a tablet, each student has a tablet, and a tablet is connected to a projector. The teacher's tablet runs a different version of the software and is able to receive submissions from the student tablets. The software enables students (and teachers) to write with "digital ink" and create mathematical objects such as geometric shapes, arrays, and tiles, which are 1×1 arrays that users "snap" together to create larger arrays.

Before a typical class in which CLP is used, the teacher (or education researcher) authors questions in the form of a notebook, creating "pages", and a copy of the notebook is transmitted to each student tablet at the beginning of class. During the class, students open their copy of the notebook and proceed to answer the questions. The students make use of the mathematical objects that the software allows them to create as they answer the questions. The student UI is shown in Figure 2-1.

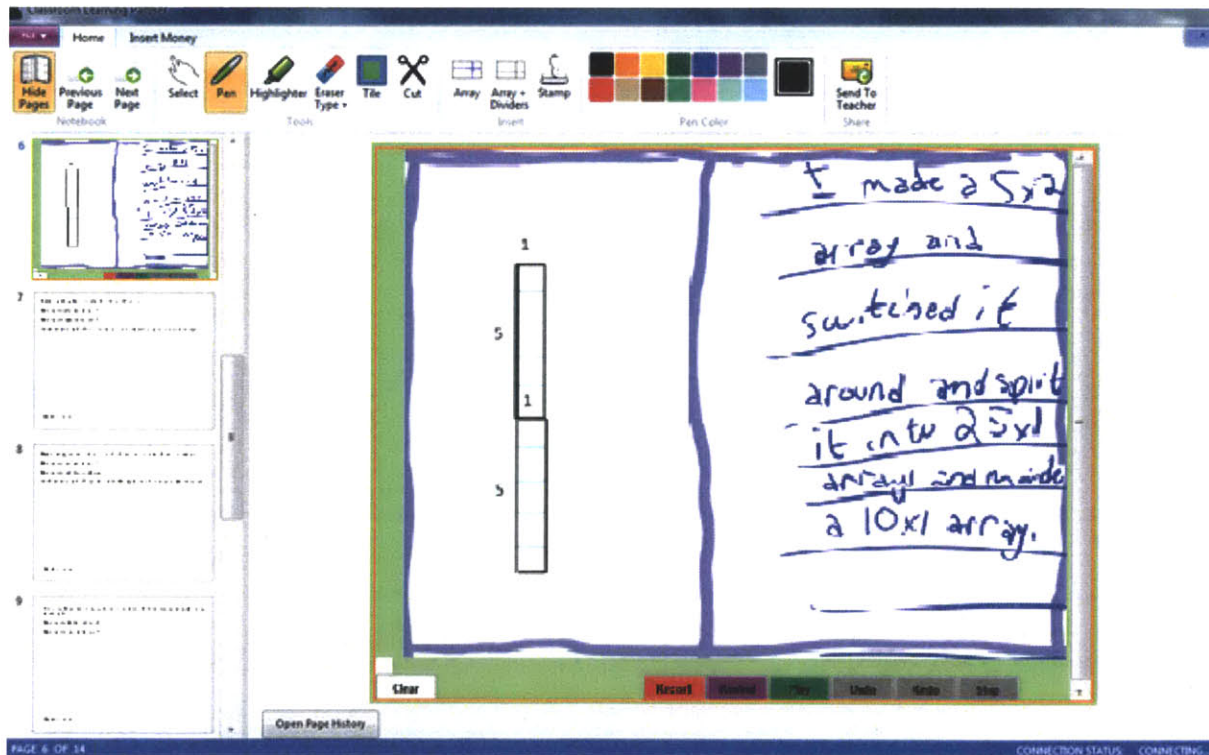


Figure 2-1: Student tablet user interface; notebook pages are shown in the left panel, students work in the main display window by using the tablet pen to draw and to tap on commands on the top tool bar

When a student has completed a question, she can submit her answer to the teacher, who can view the work on her tablet, as shown in Figure 2-2.

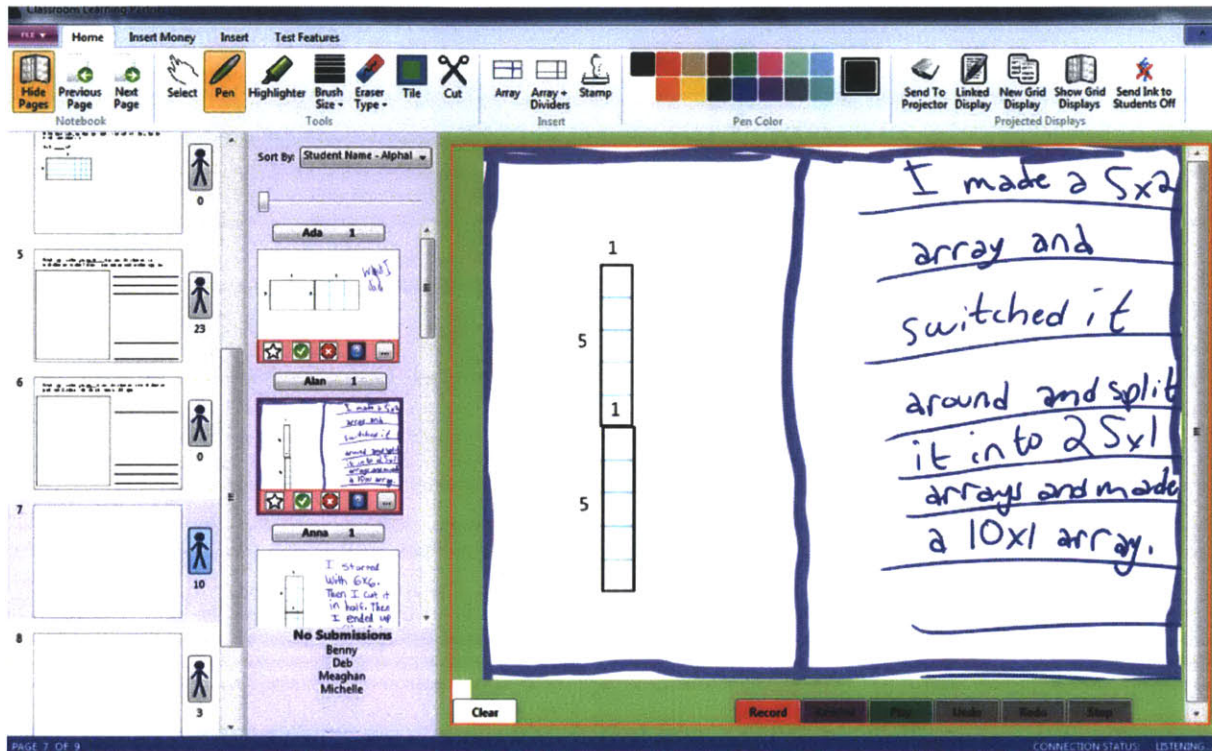


Figure 2-2: Teacher tablet user interface; student submissions for a particular page are displayed in a panel between the notebook page display and the main window

When the teacher has received an adequate number of responses for a particular question, she can pick some of the submissions she received for that question, arrange them in what we call a grid display, and send them wirelessly to the projector machine, which then projects the display to the class (See Figure 2-3). CLP uses AI techniques to sort answers to the questions so that students who made similar mistakes or used the same approach to get an answer are grouped together. These techniques help the teacher pick a representative set of submissions to discuss with the class.

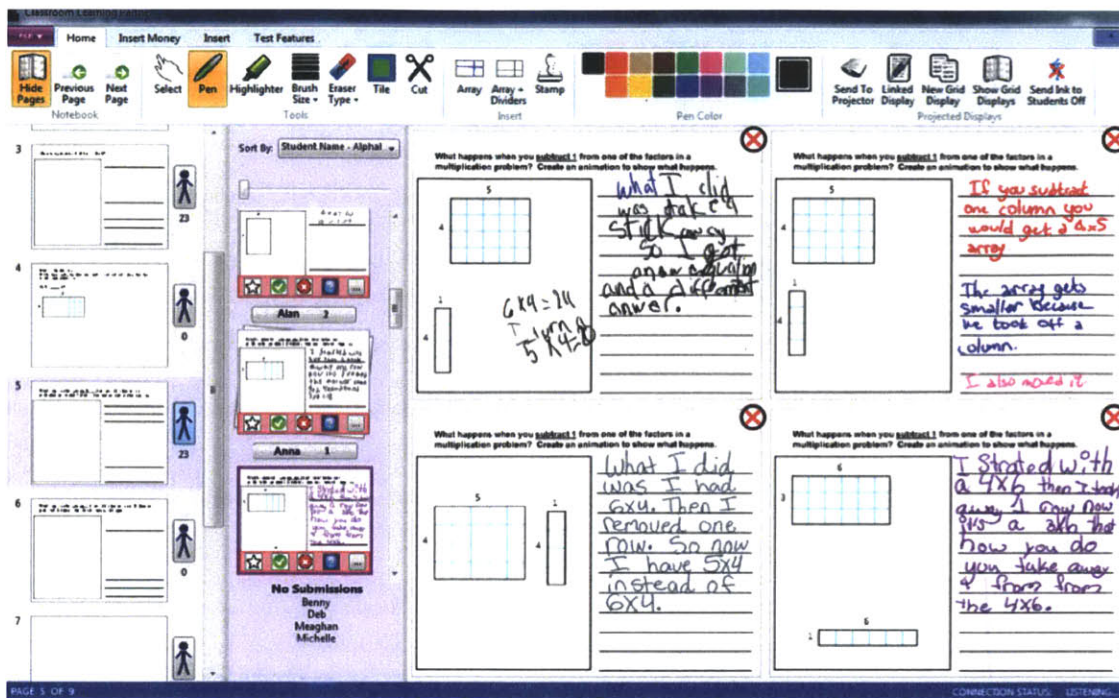


Figure 2-3: A grid display of students work on the teacher's machine; the four chosen examples will be displayed by the projector; the red X's enable the teacher to remove an example from the display

CLP is organized around the idea of a notebook page and page objects that are placed on the page. It was implemented in the Windows Presentation Foundation (WPF) framework. The animation tool described in this thesis is an extension of CLP that relies on a new kind of notebook page that allows students to create animations. The new page has all the functionality previously available on a page and also allows students to create and record animations. The new page includes the necessary animation controls. In addition, the set of available mathematical objects was augmented to support animation actions such as cutting page objects into smaller pieces. The architecture already included a history of user actions performed on a page, and new methods were added to play this history backwards and forwards in order to create, play and rewind animations.

Although there are many mathematical objects that can be created on a page, most of them have a common API which the design of the animation software exploits. This page object abstraction was useful because it allowed us to extend the software to include all the necessary methods and then treat the objects in the same way. This abstraction reduced the task of creating an animation to recording the insertion, deletion, resizing and movement of objects along a timeline.

3 Design Objectives

There were two main design objectives for the animation tool: simplicity and structure.

3.1 Simplicity

A tool which is supposed to make a task easier might initially make it more difficult because time and effort must be invested to master the use of the tool. In designing the animation software, we sought to minimize or even eliminate this initial cost of time and effort. There is already more material in the elementary school curriculum than teachers can hope to cover. Anything that further decreased the time available for students' learning would likely be met with skepticism. Furthermore, the software is supposed to make math more accessible to students, not less. We sought to avoid having any student unable to use the software and therefore unable to understand the ideas conveyed through its use. As the initial design evolved during testing in a classroom setting, extraneous function was removed for simplicity.

3.2 Structure

We intentionally built well-defined structure into the tools that students would use to create their animations. This structure was meant to simplify the process for the students without restricting them too much. It also made the software design simpler and easier to implement without any significant sacrifices to our pedagogical goals. In the initial design, for example, students would have been able to draw any closed shape, cut it into smaller pieces and move those pieces around to make an animation. The sweep line algorithm required for this task is relatively complicated and computationally intensive. The majority of the shapes used in the arithmetic curriculum with which we were working are regular, so they can be divided into smaller pieces using less sophisticated methods. We were thus able to restrict the software to using regular shapes without compromising the mathematics learning goals. Later, special emphasis was placed on arrays and other objects in the software that were used more often in student animations created for arithmetic proofs.

4 Software Design

We used a form of Agile software development in which we started with an initial design, observed what students did with it in a classroom setting and then fixed bugs and implemented an updated design before returning to the classroom the next day to repeat the cycle. In educational technology development, this iterative approach is often referred to as design-based research [2, 3, 8]. Our initial design involved two extensions to CLP: a new math tool and the animation software. Our classroom observation of students using the initial and subsequent new versions of the software took place in two Boston-area classrooms, three days in a classroom with 23 third grade students (Class 1) and two days in a classroom with five fourth and fifth grade special needs students (Class 2). Below is a description of the CLP extensions and each version of the software.

4.1 Extensions to CLP

There were two extensions that we needed for the animation software, a cutting tool (Scissors) and a new type of CLP page (Animation page).

Scissors

The Scissors command enables students to split the mathematical representations already available in CLP. It enables students to cut mathematical representations—an array, square, etc.—into two objects at a specific point, as shown below in Figure 4-1.

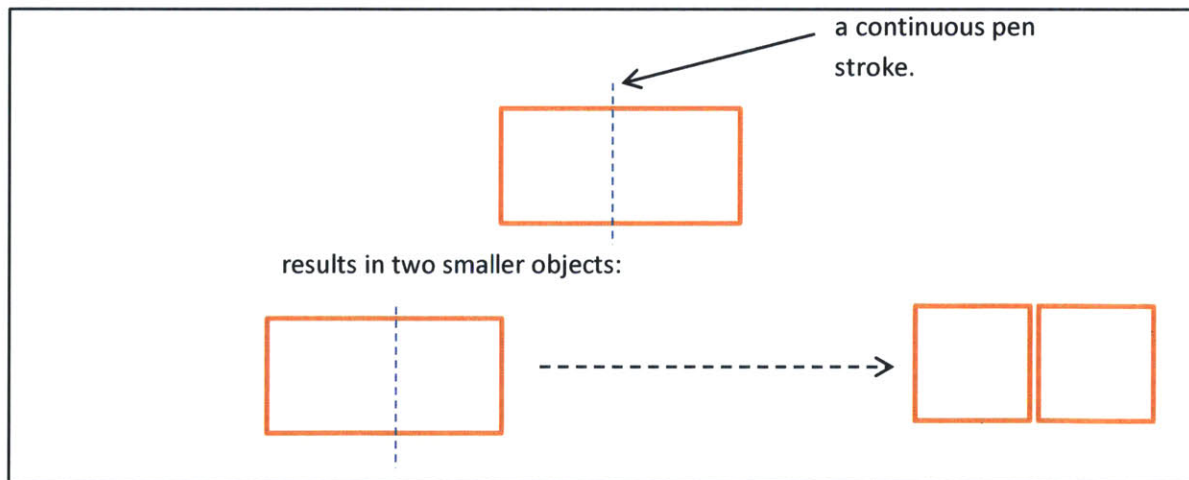


Figure 4-1: The new cutting math tool; the cut can be vertical or horizontal

Such a cutting tool is necessary for the kinds of mathematical proofs that are prevalent at the elementary grade level, e.g., showing mathematical equivalencies such as $4 \times 6 = 8 \times 3$. For such a proof, a student could create a 4×6 array, cut it into two 4×3 arrays, then move one of the 4×3 arrays below the other one to show an 8×3 array. Shown in Figures 4-2a and 4-2b are the beginning steps in such a proof.

To use the Scissors, students tap on the ✂ icon in the top command bar and use the tablet pen to draw a continuous stroke across the object where they wanted to split it, as shown below. While the line is being drawn, the cursor is a scissors icon.

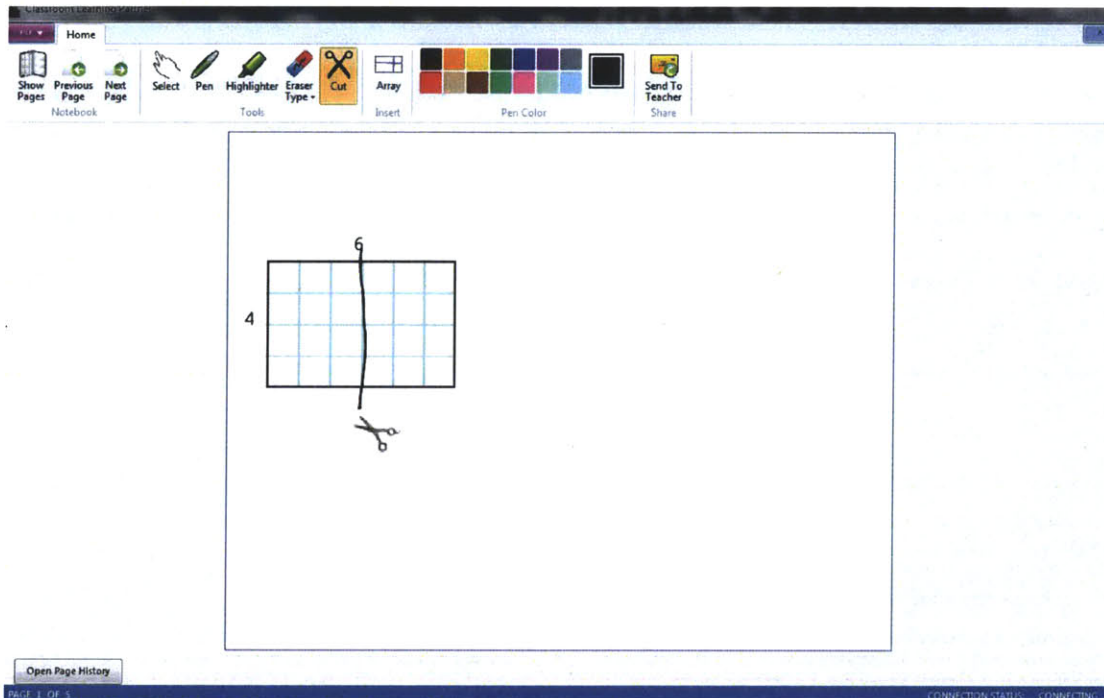


Figure 4-2a: Using the Scissors cutting tool on a 4x6 array; students tap on the Scissors icon then draw a line where they want to cut

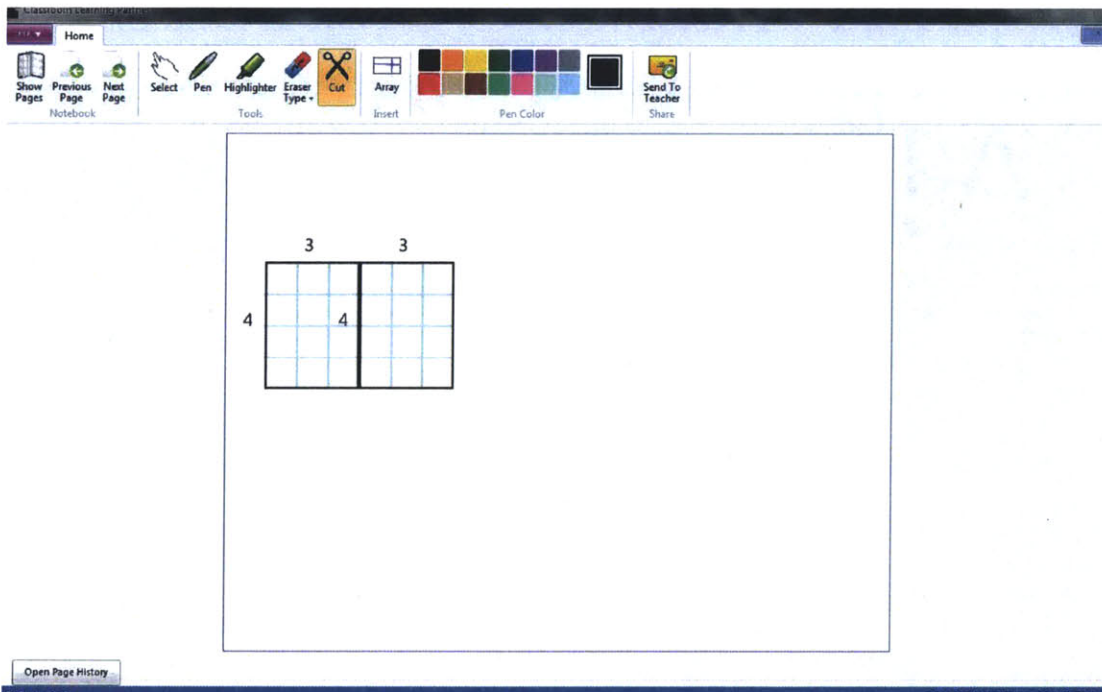


Figure 4-2b: Result of cutting the array shown in Figure 4-2a; two 4x3 arrays are created

Animation Page

A new kind of page was necessary because we needed a place to conveniently locate controls for creating and playing animations. We did not want to have the controls available for every page in a notebook, as they might prove distracting when not relevant to a particular lesson, so we opted to not put the controls on the top command bar. Instead the controls are visible on the Animation Page itself and placed at the bottom of the page, out of students' way. In order to have the functionality that was available on other CLP pages on the Animation Page, the Animation Page was implemented as a subclass of the regular CLP page. A CLP object model showing the page extension is shown below in Figure 4-3. The data object model uses the standard notation found in *Program Development in Java: Abstraction, Specification, and Object-oriented Design* [5].

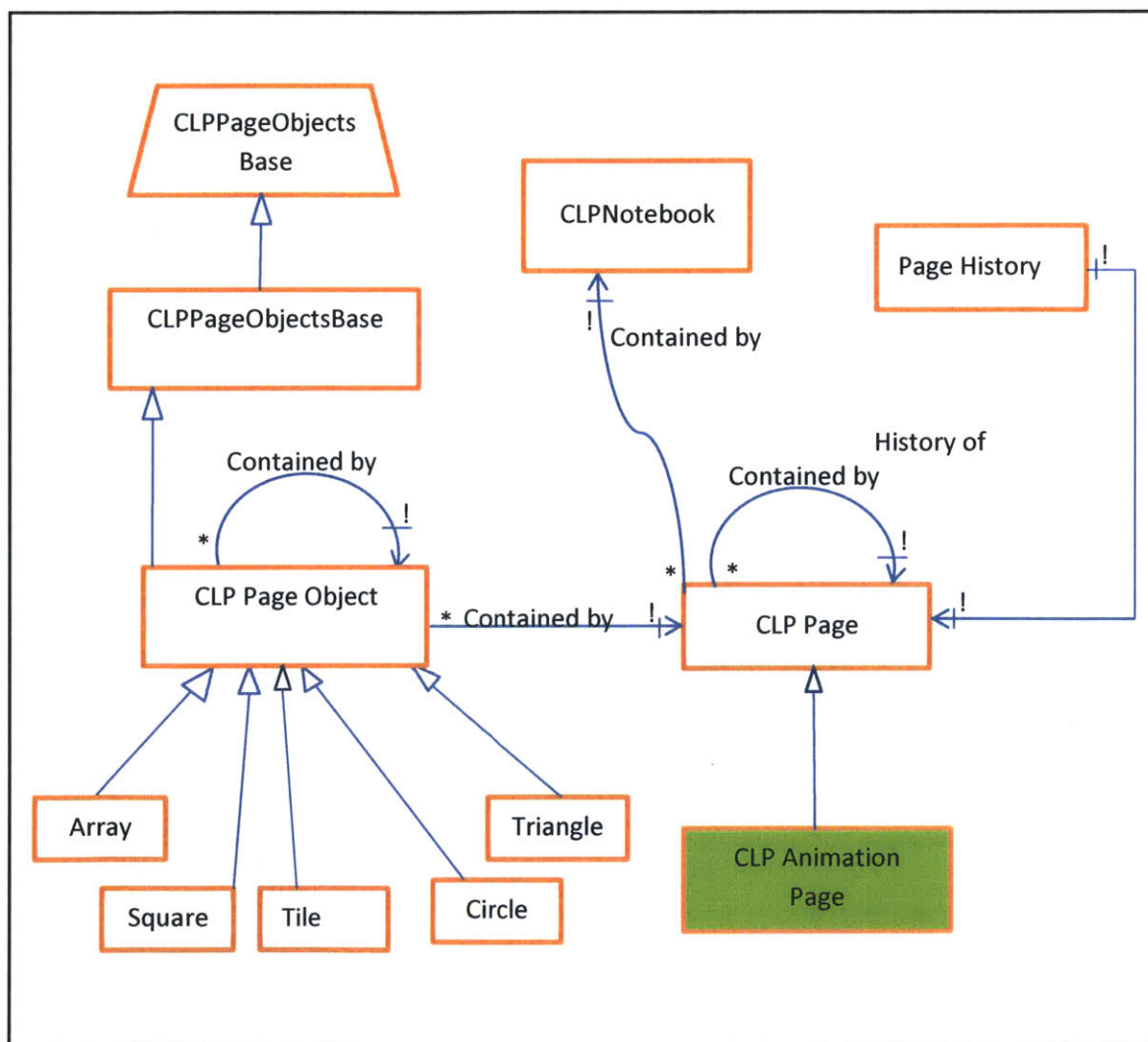


Figure 4-3: A data object model showing the extensions to the CLP software; see the Appendix for a more detailed explanation of the classes in this diagram.

4.2 Designs

This section describes five designs: three designs tested in Class 1, one design tested in Class 2, and a final design that will be tested in the Fall of 2013.

4.2.1 Design 1, Class 1

We sought to have a UI that was intuitive for students to use. To accomplish this goal we decided to pattern our UI after that of a general media player. This design has the benefit of familiarity as students are likely to have seen it before. More importantly, it captured the functionality that we needed. The UI is shown in Figure 4-2 below. It had a Record button that was used to indicate when actions would begin to be recorded to create an animation. It had Rewind, Play and Pause buttons for the animation, and it also had an Insert button which could be used to edit an animation at a specific location after the animation had been rewound, forwarded, and/or played to the location. When an animation was paused, students could edit it and have all their edits performed as a single action during playback. The Stop button ended whatever action was taking place and rewound the animation to the beginning; the objective was to mimic the behavior of a general media player's stop button.

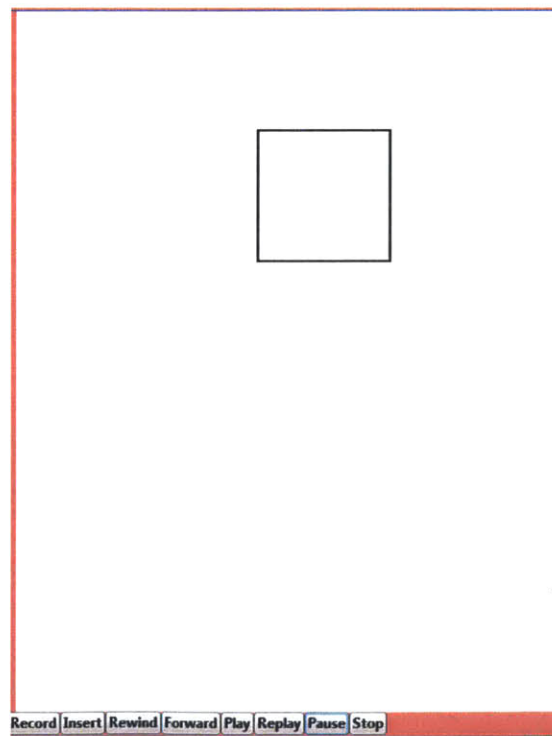


Figure 4-2: The first UI for the animation software (shown in portrait mode)

This design was used in Class 1 for an hour. We noticed that students were often unable to tell if an Animation page already contained an animation, and they tended to lose track of their position in the animation they were currently authoring. They sometimes seemed to get confused by the many buttons, and they did not use all the buttons. In particular, they never used the insert

button. We noticed some confusion about the difference between the Pause and Stop button. We decided to leave both buttons in for another day, then revisit the decision to include both.

Sometimes a student would tap several buttons in quick succession. This resulted in concurrent access to the history of an animation by multiple threads. The effect was that actions in the animation would become disordered. If a happened before b when the animation was created, b would now happen before a .

We also noted that the teacher's machine would sometimes slow down when too many animations were submitted.

4.2.2 Design 2, Class 1

Improvements

To improve Design 1 and address the problems we observed when it was used by students, we made the following changes:

Animation present indicator

We added an indicator—a small box on the bottom left of the animation page, above the clear button—that turned black when an animation was present on the animation page.

Progress bar

We added a progress bar that spanned the bottom of the page, indicating a user's position in the animation. The progress bar also enabled a user to tell how much of an animation had already played during playback.

Fewer buttons

We removed the Forward and Replay buttons to simplify the UI. The functionality of these buttons was either unnecessary or could be achieved by a combination of the remaining buttons.

Lock Mechanism

We created a global lock on the history that represents an animation to prevent concurrent access by different threads.

Smaller Histories

The software design that we used to create animations keeps a history of events that could be redone on a page to replay the animation. The time lapse between the events controls how fast the animation is replayed. Although individual animations tended to be relatively short, moving objects around when creating an animation produced large histories. This situation occurred because, as an object was moved from one point to another its position was sampled (at a pre-defined rate), and whenever a sampling produced a new position, this position was recorded as an item in the history. To get an accurate path of the object's movement, a higher sampling rate was required, which led to large histories. The large histories proved problematic when students submitted their animations to the teacher's machine; sometimes the teacher's machine slowed down and in extreme cases froze. To obviate this problem, we needed to reduce the size of the

history that resulted from storing an object's path as a list of history items. Since each history item was actually a class instance, we were able to get a significant improvement by modifying the class so that several contiguous new positions of the same object within the history were stored in a single class instance that contained a list of the points.

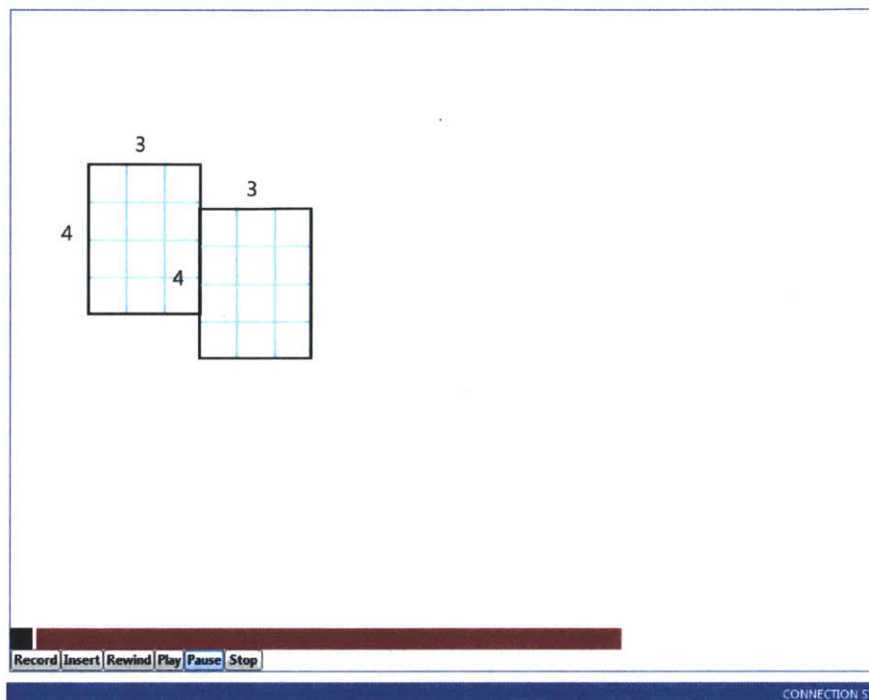


Figure 4-3: The second UI of the animation software

A picture of the UI for the second design is shown above in Figure 4-3. This design was used in Class 1 for an hour. The indicator was easy to see, and students had no trouble understanding its meaning. Students had no trouble understanding the progress bar.

The smaller histories in this design prevented the teacher's machine from slowing down when the teacher received submissions from the students and the lock mechanism prevented concurrent access to the history representing an animation. This resulted in a consistent replay of all the animations.

We noticed that none of the students used the Insert button and they rarely paused their animations during replay. The students sometimes decided that they wanted to start over by deleting everything in their current animation, but we had not provided the equivalent of a "delete all", so the deletion process sometimes proved time-consuming.

We had instructed students to pause their animations when they wrote text so that the writing (and erasing) of ink was excluded from the animation—only the final text would be included in the animation. The students rarely remembered to pause their animations and when the teacher replayed some of them for the class, significant portions of time would be spent watching the student's ink appear and disappear (when erased).

We also noticed that students sometimes had trouble distinguishing between the buttons.

4.2.3 Design 3, Class 1

Improvements

To improve Design 2 and address the problems we observed when it was used by students, we made the following changes:

Fewer buttons

We removed the Insert and Pause buttons to simplify the UI. The students did not seem to use these buttons. We changed the functionality of Stop to simply stop whatever action was taking place and to leave the animation at the same location rather than stopping and rewinding. The automatic rewind seemed confusing to students, plus they could explicitly use the Rewind button if they wanted to.

Clear Button

We added a button that erased everything and allowed students to start over easily.

More Color

We made the UI features more colorful (see Figure 4-4 below) to make them more visible (and friendly) to elementary students: The buttons are colored, the animation indicator is blue, and the animation progress bar is yellow.

Ignore the ink

We made the contiguous text that students wrote appear on the screen at once during playback.

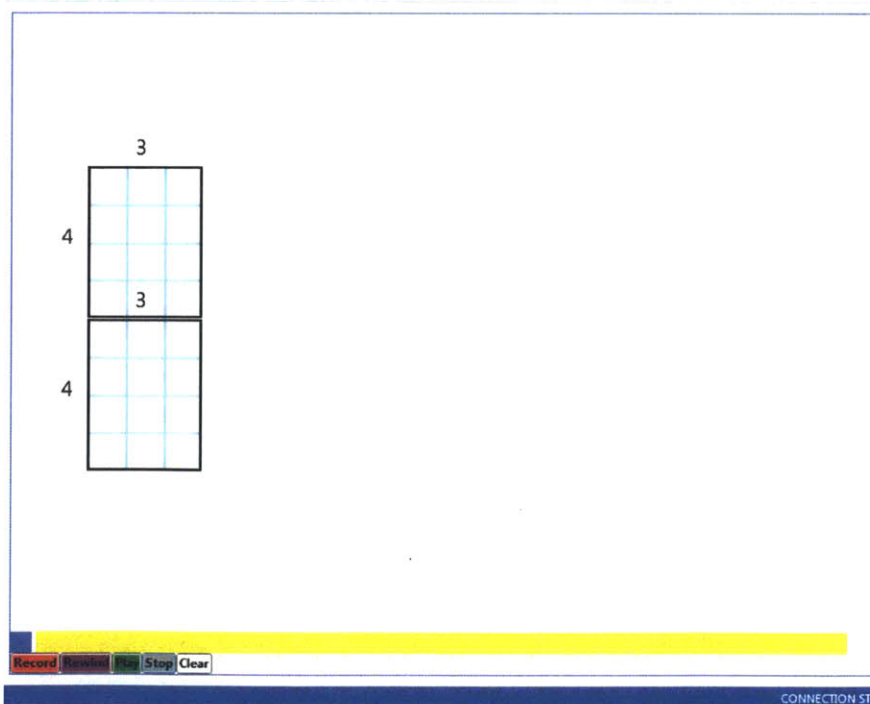


Figure 4-4: The third UI of the animation software

A picture of the UI for the third design is shown above in Figure 4-4. This design was used in Class 1 for an hour. The more colorful buttons in the user interface made the students remain more engaged when interacting with the software.

We noticed that the students sometimes had trouble clicking on the small buttons. They also tended to use the Clear button too frequently, sometimes even by accident.

4.2.4 Design 4, Class 2

Improvements

To improve Design 3 and address the problems we observed when it was used by students, we made the buttons larger so that students would be able to click on them easily. We also moved the Clear button off to the side so that students would not click on it by accident.

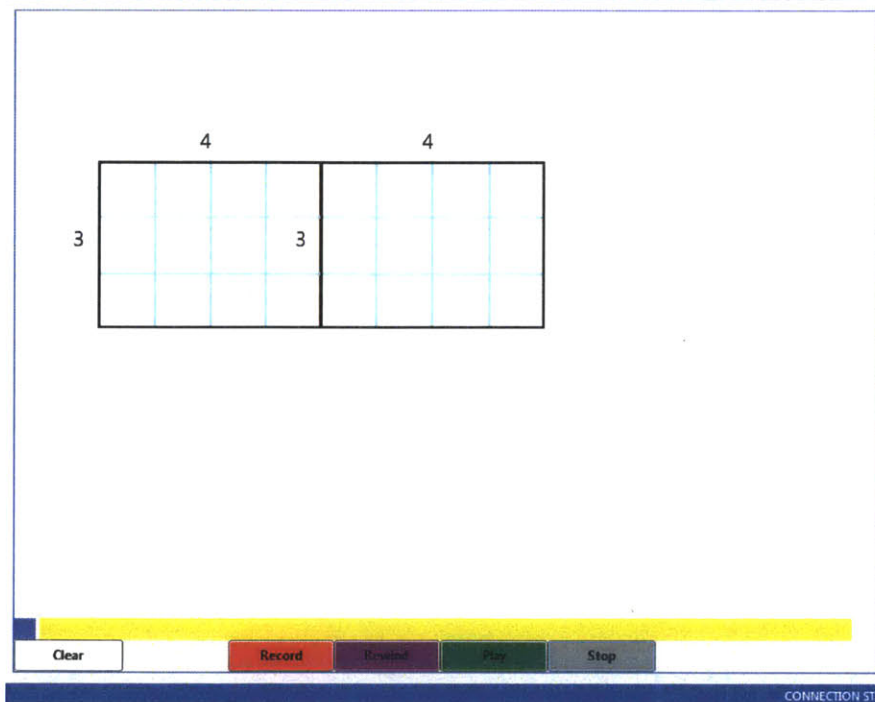


Figure 4-5: The fourth UI of the animation software

A picture of the UI for the fourth design is shown above in Figure 4-5. This design was used in Class 2 for two days, one hour per day. The students had no problems using the larger buttons and they no longer hit the Clear button by accident.

We noticed that the teacher sometimes wanted to vary the speed at which an animation played back. For more complex animations, the teacher would want to play them back slowly so that students could see exactly what was taking place but for simpler animations, she wanted to play them back faster. The teacher and students would sometimes want to forward/rewind an animation to a specific spot and sometimes the students lost track of what they were doing with an animation (rewinding, editing, replaying, etc.)

In the process of authoring a notebook that students would use in a class, the education researcher sometimes found that she wanted to convert a regular page to an animation page because it occurred to the teacher and to her that students might want to use an animation in their answers.

4.2.5 Final Design, To Be Tested In Fall 2013

Improvements

To improve Design 4 and address the problems we observed when it was used by students, we made the following changes:

Variable playback speed

Initially there was an attempt to decide on an optimal average speed, but it was better to create a user control, implemented as a slider, that allowed the teacher and students to change in real time the speed at which an animation was playing. The slider is located between the Clear button and the animation control buttons.

Location slider

We added a location slider that played the history items one at a time and allowed a user to play an animation by moving the slider back and forth, until he got to a specific location. The slider is also useful because it allows a user to forward an animation past uninteresting or repetitious sections. In the current design it is overlaid on the yellow animation progress bar.

Recording cue

We also made the border of the animation area flash red when students were recording an animation. This modification provided an extra cue that helped students to keep track of what they were currently doing.

Converting a regular page to an animation page

We added a button that converts a page into an animation page, leaving anything already added to the page untouched.

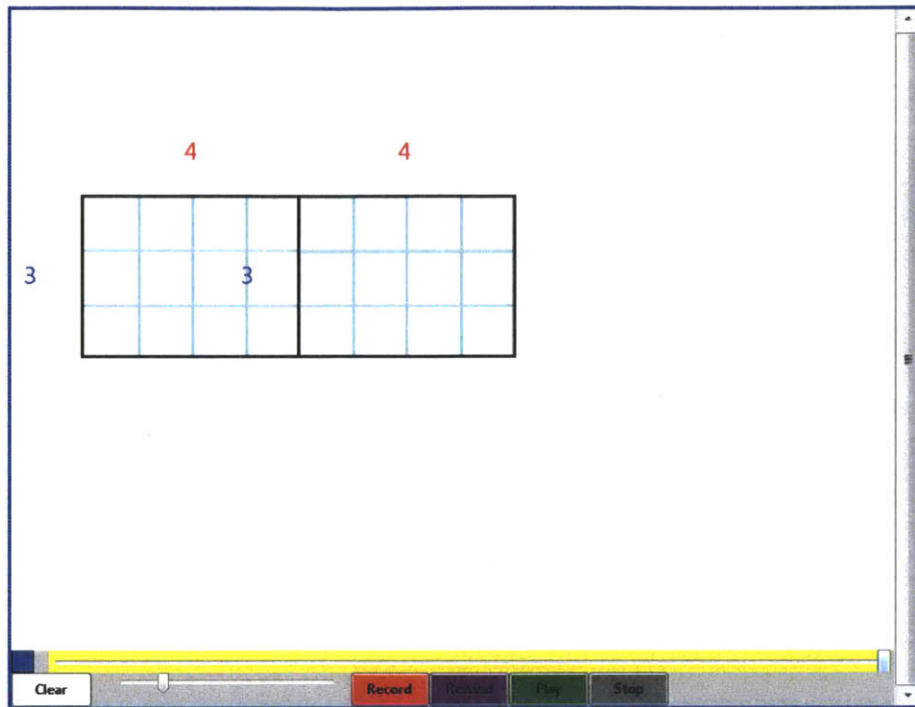


Figure 4-5: The final UI of the animation; location slider is shown all the way to the right on the progress bar, users drag it back and forth to move forwards and backwards through an animation; speed control slider is between the Clear button and other control buttons

A picture of the UI for the final design is shown above in Figure 4-5. This design will be tested in the Fall of 2013.

5 Student Work

This section presents examples of student work created using the animation software and discusses the work from a math education point of view.¹

5.1 Class 1

The first class consisted of 25 students in third grade, and we spent an hour per day with them for four days. The students and teacher had worked with the math education researchers whose work has influenced the development of our tools [9], so they were very familiar with the process of creating a mathematical proof about arithmetic operations.

On the first day, we introduced the students to the software and the available math tools. The students did not create animations that day, but rather practiced using the tablet pen to write and draw and to create, cut, move, and resize arrays. On subsequent days, the students practiced array creation and cutting and created animations. The student work discussed below is from the fourth day during which the students created animations for two proof problems.

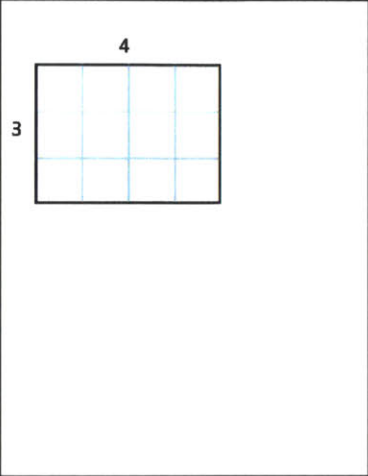
Problem 1: *How can you prove that $3 \times 4 = 6 \times 2$?*

In order to solve this problem, students had to create either a 3×4 array or a 6×2 array, cut it in half and re-arrange the halves to create the other array, recording the change from one shape to the other in an animation. The proof of the equality of the two expressions relies on the fact that there is still the same number of units in the representation even though the shape has changed.

Several student responses are shown below. One student's response, shown in Figures 5-1a through 5-1c is typical. She began with a 3×4 array, cut it into two 3×2 arrays and moved one of the smaller arrays under the other small array to create a 6×2 array. While she wrote her explanation after creating the animation, it is not officially part of the animation and stays on the page while the animation is playing.

¹ This section is excerpted from the Year 1 annual report for the NSF-funded project *Technology to Support Mathematical Argumentation*, Kimberle Koile and Andee Rubin, Principal Investigators.

How can you prove that $3 \times 4 = 6 \times 2$?

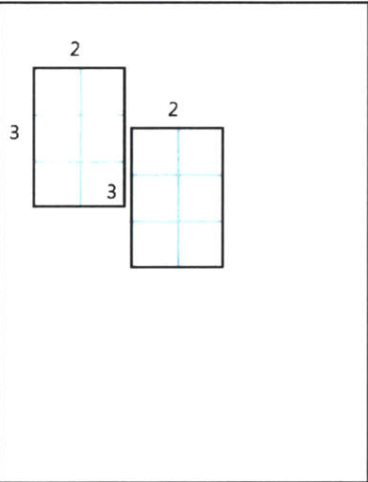


I started with a 3×4 array then I made it into a 6×2 array.

I cut the 3×4 array into a $3 \times 2 + 2 \times 3 \times 2$ array then moved one 3×2 array under the other one.

Figure 5-1a: An early state of a student's animation

How can you prove that $3 \times 4 = 6 \times 2$?



I started with a 3×4 array then I made it into a 6×2 array.

I cut the 3×4 array into a $3 \times 2 + 2 \times 3 \times 2$ array then moved one 3×2 array under the other one.

Figure 5-1b: An interim state of the student's animation

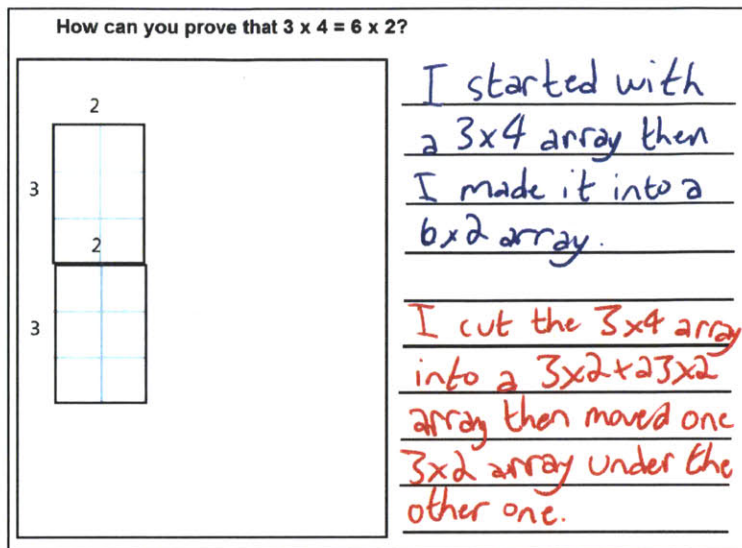


Figure 5-1c: The ending state of the student's animation

Some students wanted to show the transformation in both directions, i.e., from 3×4 to 6×2 and from 6×2 to 3×4 . One example of this approach is shown in Figures 5-2a through 5-2c. Before starting her recording, this student created a 3×4 array, a 2×6 array and two 3×2 arrays. Her final screen shows just two arrangements: a 6×2 made up of two 3×2 arrays and a 3×4 made up of two 3×2 arrays. Her written explanation is a little hard to follow, partly because of her use of some vocabulary that had been developed in her classroom (e.g., "stick" to mean an array that is much longer than wide) but her oral explanation in class was quite clear.

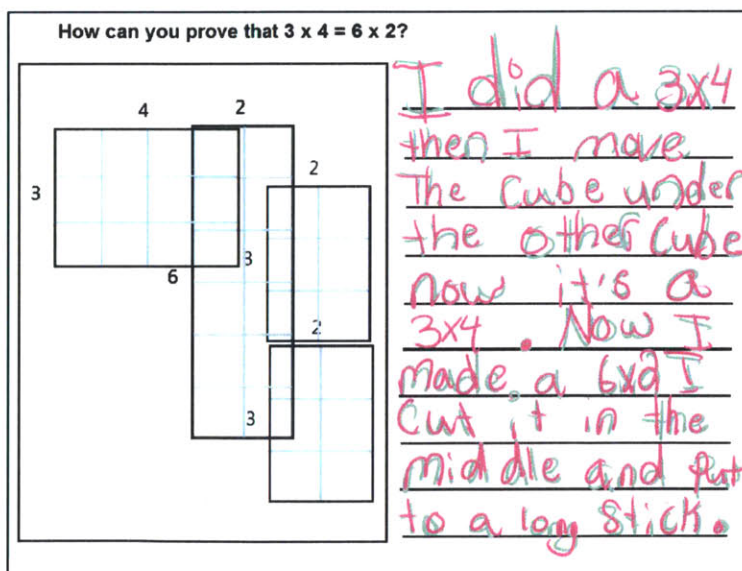


Figure 5-2a: An early state of a student's animation

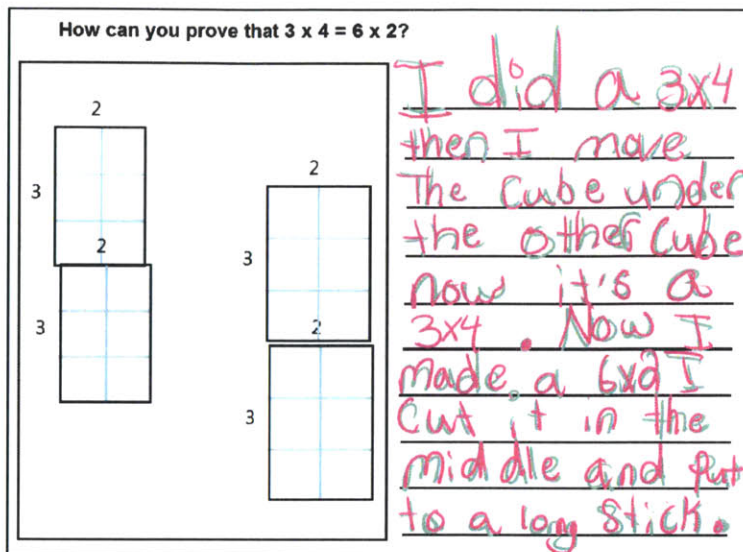


Figure 5-2b: An interim state of the student's animation

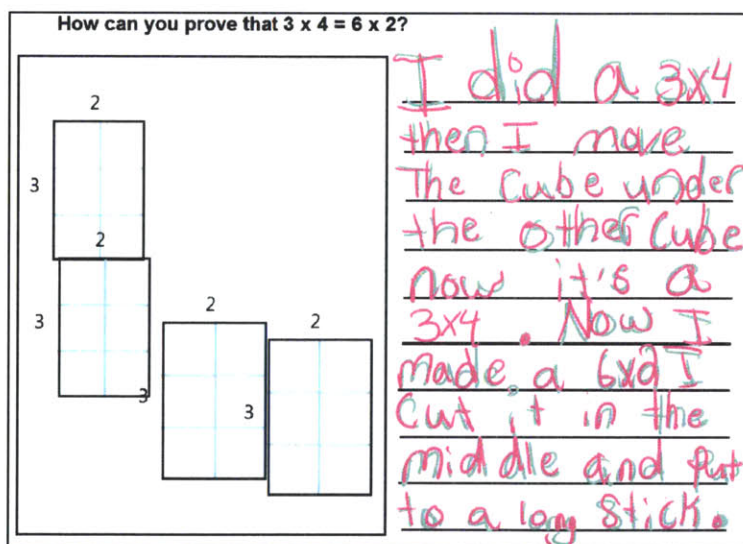


Figure 5-2c: The ending state of the student's animation

While few students actually wrote down a statement that “the quantity had not changed, just the arrangement of the arrays,” the teacher led the class discussion in that direction and at least one student included the following in his work, “The arrays are different sizes but have equal numbers.” (By “size,” he means “shape.”)

All but two or three students were successful in creating animations to prove this equivalence. Several students went on to make up their own, similar problems, such as showing that $5 \times 2 = 10 \times 1$ or $6 \times 6 = 12 \times 3$.

Problem 2: What happens when you subtract 1 from one of the factors in a multiplication problem? Create an animation to show what happens.

While this problem was stated as a general question, we decided after talking with the teacher to give the students a particular multiplication problem to begin with; we used the problem 6×4 . We planned to move to the more general problem statement if we had time, but we did not.

The key insight we hoped students would have in working on this problem is that subtracting 1 from one of the factors can be shown by subtracting a row or a column from the array representing the original multiplication problem. Some students initially thought that they could show “subtracting 1 from a factor” by subtracting 1 from the product. In fact, our software made this impossible, since using the Scissors on an array automatically cut off some number of rows or columns. However, we observed at least one student trying—unsuccessfully—to cut off a single unit from an array. Her eventual solution to the problem is shown below in Figures 5-3a through 5-3c, which show the beginning, middle and end of the animation she created. She cut off one column from the 6×4 array and moved it to the side. Her written explanation demonstrates her struggles.

What happens when you subtract 1 from one of the factors in a multiplication problem? Create an animation to show what happens.

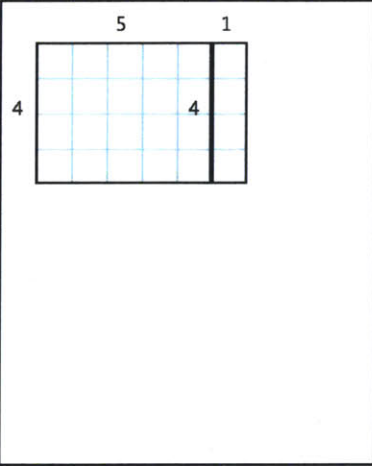
6

4

I notice that
if you take
away 1 your
answer would
be 1 more
less and if
you take aw-
ay 4 your
answer would
4 more less.

Figure 5-3a: An early state of a student's animation

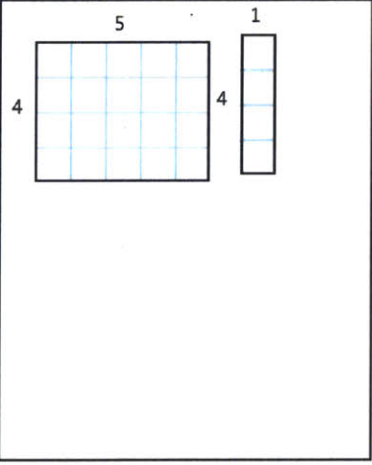
What happens when you subtract 1 from one of the factors in a multiplication problem? Create an animation to show what happens.



I notice that
 If you take
 away 1 your
 answer would
 be 4 more
 less and if
 you take aw-
 ay 4 your
 answer would
 4 more less.

Figure 5-3b: An interim state of the student's animation

What happens when you subtract 1 from one of the factors in a multiplication problem? Create an animation to show what happens.



I notice that
 If you take
 away 1 your
 answer would
 be 4 more
 less and if
 you take aw-
 ay 4 your
 answer would
 4 more less.

Figure 5-3c: The ending state of the student's animation

While many of the students created animations similar to the one above, cutting off and moving either a row or a column, several students realized that they could subtract one from either factor, ending up with two different array manipulations. One student, for example, created two 6×4 arrays, cut a row off one and a column off the other and wrote an explanation for the two actions. His animation is illustrated below in Figures 5-4a through 5-4c.

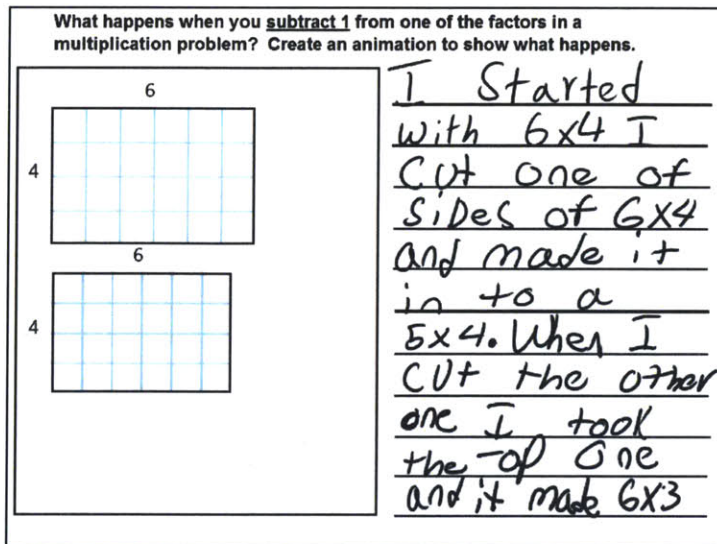


Figure 5-4a: An early state of a student's animation

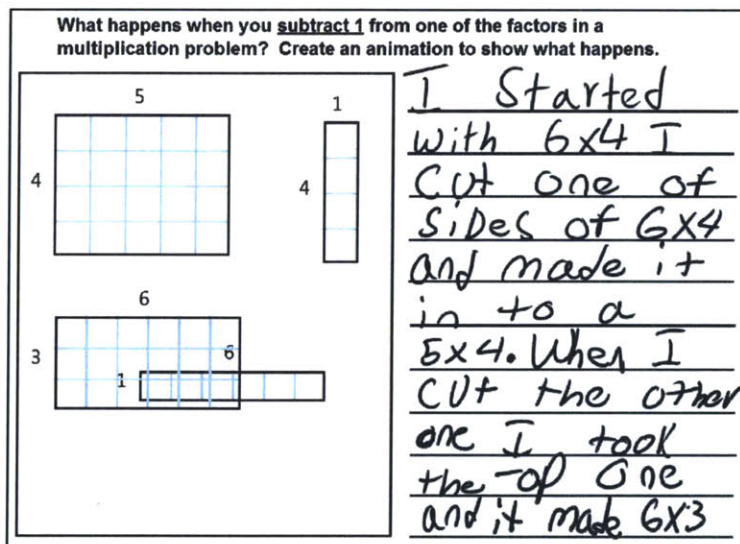


Figure 5-4b: An interim state of the student's animation

What happens when you subtract 1 from one of the factors in a multiplication problem? Create an animation to show what happens.

I Started
with 6×4 I
cut one of
sides of 6×4
and made it
in to a
 5×4 . When I
cut the other
one I took
the ~~of~~ one
and it made 6×3

Figure 5-4c: The ending state of the student's animation

Another student, whose work was shown in Figure 1-4 and is shown in more detail in Figures 5-5a through 5-5c, created two 6×4 arrays, cut a row from one and a column from the other. He also noted what the final products were and how each differed from the original answer of 24. (His picture is actually mislabeled, but his text is correct.)

What happens when you subtract 1 from one of the factors in a multiplication problem? Create an animation to show what happens.

When you subtract
1 It's like you
subtract 4 or 6
from it. I take
6 from it and
it will equal 18
because $3 \times 6 = 18$
and if I subtract
4 it will equal
20 because 4×5
 $= 20$.

Figure 5-5a: An early state of a student's animation

What happens when you subtract 1 from one of the factors in a multiplication problem? Create an animation to show what happens.

When you subtract
 1 It's like you
subtract 4 or 6
 from it, I take
 6 from it and
 it will equal 18
 because $3 \times 6 = 18$
 and if I subtract
 4 it will equal
 20 because 4×5
 = 20.

Figure 5-5b: An interim state of the student's animation

What happens when you subtract 1 from one of the factors in a multiplication problem? Create an animation to show what happens.

When you subtract
 1 It's like you
subtract 4 or 6
 from it, I take
 6 from it and
 it will equal 18
 because $3 \times 6 = 18$
 and if I subtract
 4 it will equal
 20 because 4×5
 = 20.

Figure 5-5c: The ending state of the student's animation

All in all, the students were able to use the software tools quite successfully. They were adept at creating, cutting, and manipulating arrays and demonstrated their understanding of mathematical proofs by successfully using the animation tool to create proofs about arithmetic operations.

5.2 Class 2

The second class was a special education class with five fourth and fifth grade children, with whom we spent one hour per day for two days. We visited the class because we wanted to know if the new cutting and animation software tools were accessible to struggling students: Could the students understand the concept of cutting arrays? Could they easily move the new small arrays on the screen? Could they understand the concept of creating an animation that explained their solution to a problem?

We had worked with these students on several previous occasions so the students were familiar with CLP and needed no introduction to using the pen or creating arrays. Unlike the students in Class 1, these students had not been introduced to the idea of proof, so on the first day, we introduced the new software features—the cutting tool and animation—in the context of math problems familiar to the students, namely division with remainders. Then on the second day we introduced the idea of proof and had the students create animations for proofs similar to some of the proofs in Class 1.

Practice with cutting, introduction to animation

The first day, students worked division with remainder problems, three problems that involved cutting and moving parts of arrays without animation and one that involved creating an animation. The animation problem was: *Ms. Lockwood's class is putting flowers in vases. They have 13 flowers, and they want to put 3 flowers in each vase. How many vases do they need?*

Below in Figures 5-6a and 5-6b are screen shots showing how one student created an animation to show his solution for this problem. The students were given a 1×13 array to start the problem. In this example, the student tapped on Record, then cut the array to represent putting four flowers in each vase, moved the smaller arrays that represented vases together, and determined that there would be one flower left over. He then sent his notebook page to the teacher, who was able to view and play the animation on her machine and on the projector machine.

Ms. Lockwood's class is putting flowers in vases. They have 13 flowers, and they want to put 3 flowers in each vase.

How many vases do they need? _____

How many flowers are left over? _____

Create an array and split it to show how many vases they need.

3
1 | Vase

3 3 1
1 | Vase 1 | Vase 1

3
1 | Vase

Number sentence: _____

Figure 5-6a: An early state of a student's animation: cutting and moving arrays

Ms. Lockwood's class is putting flowers in vases. They have 13 flowers, and they want to put 3 flowers in each vase.

How many vases do they need? 4

How many flowers are left over? 1

Create an array and split it to show how many vases they need.

3
1 | Vase

1 | Vase

1 | Vase

1 | Vase

1 |

4 vases

1 flower left

Number sentence: $13 \div 3 = 4 \text{ r}1$

Figure 5-6b: The ending state of this student's animation

The students had no trouble using the new features. All the students but one were extremely engaged in using the animation tool and enjoyed seeing their work replayed for the class. One student created an animation for one problem, then decided for subsequent problems that having to tap on the animation control buttons slowed him down, so he didn't create more animations. Instead he cut and moved arrays around and sent his final arrangement to the teacher. So animations may not appeal to all students, especially those who are impatient to be finished; we need further testing to know whether this one student's experience can be generalized. (The teacher noted that this particular student had impulse control issues and often rushed ahead during assignments, generally understanding the math but often making careless mistakes in his haste.)

Creating animations, proofs

The second day students continued to practice making animations for three division with remainder problems, then created animations for three proof problems. For the first two of the proof problems, the students were given both starting and ending states, unlike the Class 1 students, who were given only starting states. The reason for this change was the following: The Class 2 students were special education students who benefited from working with concrete examples, and providing the ending state enabled them to more easily visualize the end of their proof. We intended for the students to turn the start state of an array into the end state. We observed that some students also turned the end state into the start state, showing their proof in both directions.

The proof problems were all what can generally be called a “double and half” proof: If you double one factor in a multiplication problem and half the other, then you get the same answer. The third proof problem, which gave the students only the starting array, was the following: *Show that $3 \times 4 = 6 \times 2$.*

Below in Figures 5-7a through 5-7d are screen shots showing one student’s work on this problem. He began his animation by cutting the 3×4 array into two 3×2 s. He then created a 6×2 , cut that into two 3×2 s, moved these arrays next to each other to show their similarity to the original array (now cut into two 3×2 s), then realigned the arrays into the original 6×2 configuration.

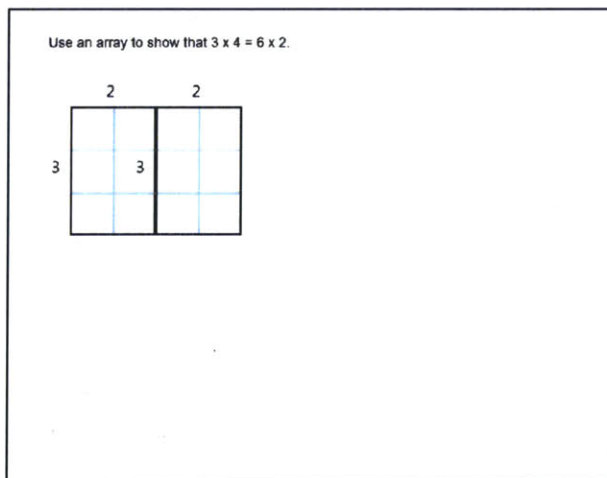


Figure 5-7a: Starting state of this student’s animation, cutting the original 3×4 array

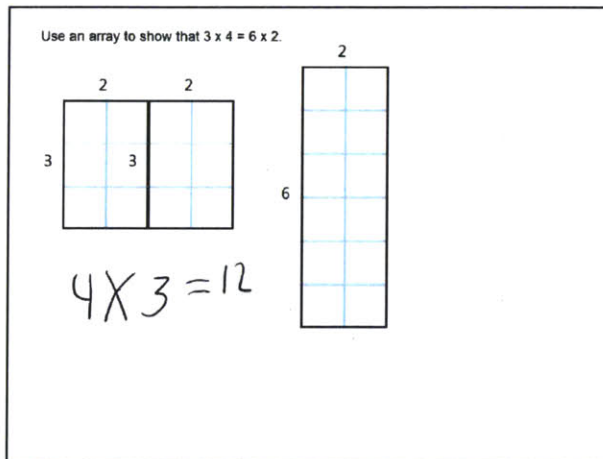


Figure 5-7b: Interim state, creating a second array to be used to match the first one

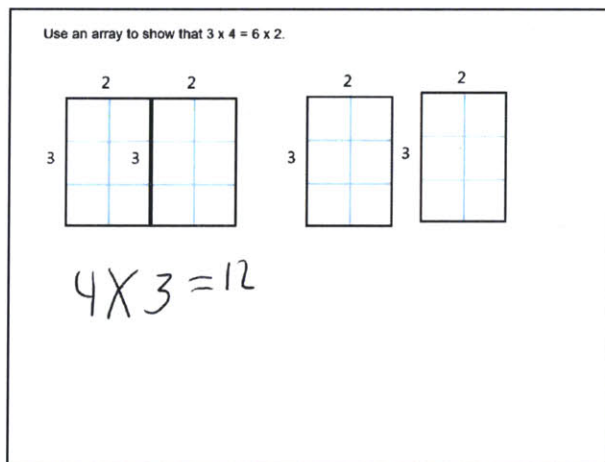


Figure 5-7c: Interim state, aligning the new 3x2 arrays to match the initial ones

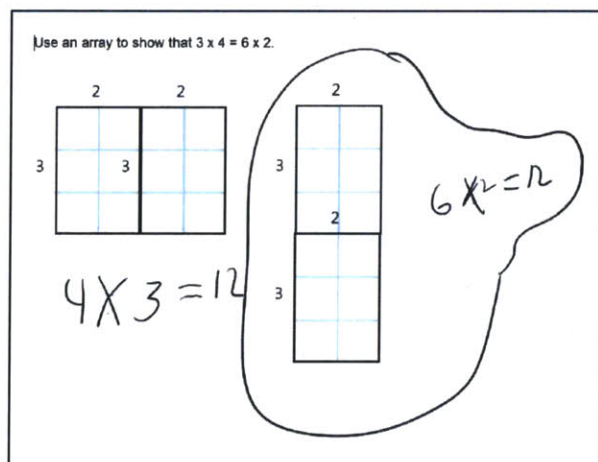


Figure 5-7d: The ending state of this student's animation

This student was very proud of his answer saying, “I showed it both ways!” The other students were equally successful, though the two fourth grade students needed some assistance from the teacher in figuring out the ending state. One of these students, who had attentional issues, became distracted and moved the arrays around randomly when he lost focus. Prompting from the teacher brought him back on task. He was able to clear his animation and start again.

Overall, we found that the answers to our original questions were yes: These special needs students easily used the cutting and animation tools, and they understood the concept of creating an animation as a solution to a problem.

6 Related Work

There are a few products on the market that enable creation of animations for teaching and learning, but their focus differs greatly from ours. They include: Screenchomp™, ReplayNote™, Educreations™, Explain Everything™, Doceri™ and FluxTime Studio™ (educreations.com, explaineverything.com, doceri.com, and fluxtime.com, respectively). Instead of comparing each of these programs to our animation software, it is more informative to look at the total set of relevant pedagogical features that they have as a group and compare that set of features with our software.

These programs allow a user to record audio and accompanying text as it is written on a screen and then play it back. A typical use is to create an audio recording of a lesson or lecture, enabling students to also see writing as it appears on a “board”. Some of the programs also enable users to import images and video. Many of the programs are excellent teaching tools especially when a teacher is creating a lesson that will be watched by students online. They differ, however, in some very important ways from our software.

Just creating an audio recording of a lesson or lecture where you can also see writing as it appears on a board is not enough functionality to allow students to create mathematical animations. Students need to have a comprehensive set of relevant mathematical objects, e.g., arrays, tiles etc.; enough so that any question that one would find in an elementary math curriculum can be represented in terms of one of the objects. You also need to be able to manipulate these objects, e.g., by cutting them up, snapping them together to form a single object, rotating them, moving them etc., and to record these actions as part of an animation. Unlike our CLP animation software, none of the other programs have this functionality.

Almost all of these other programs are geared towards an adult user group (teachers), and their interfaces are ill-suited for elementary grade students. The programs are complex enough that elementary grade students would need time-consuming lessons to learn to use them effectively. One of the other programs, FluxTime Studio, is actually geared towards children and has a simplified UI. This particular program also allows users to insert predefined objects, but it lacks mathematical objects because it is designed to be an art program. It also does not allow users to cut objects into smaller pieces or snap existing objects together to form larger objects.

Another major difference between the other programs and our CLP animation software is that they seem to be designed to substitute for classroom teaching. They do not seek to augment the classroom experience but rather to replace it with something comparable. As such, if the target audience is in the same room with the teacher, the programs become superfluous.

What really separates our CLP animation software from these other programs is the capability for students to author mathematical objects—arrays, shapes, tiles etc.—and manipulate them by

cutting them up and moving them around to create an animation that illustrates a mathematical idea. Almost any elementary math curriculum will have problems that ask a student to imagine taking common objects, e.g., a bag of oranges, and manipulate them, e.g., by moving and grouping them, to explore mathematical relationships. Software has not been available, however, that enables students to put representations of oranges on a page and actually manipulate the oranges to create a dynamic representation of a mathematical concept.² Combined with the ability to record and replay the entire exercise, this functionality provides a powerful learning tool for students to use in class. The animations created in this way are submitted to the teacher in real time, and the teacher can pick some of them to replay to the class in order to illustrate something important or address a common mistake. (See Section 5 for examples of student animations.)

² CLP includes functionality that enables students to draw and make copies of representations using what we call stamps. The resulting representations resemble hand-drawn pictures, though they are actually composed of page objects that can be manipulated. We have not used this tool to date in mathematical proof problems, but [9] includes examples of students creating proofs containing hand-drawn representations for which the stamp tool would be applicable. See [4, 7] for examples of student work with stamps in multiplication and division problems.

7 Future Work

This section describes improvements to the current software that would make for a better learning experience. Although one cannot be sure how practical or useful the suggestions are without actually implementing them and testing them in a classroom setting, our classroom experience to date leads us to think that these suggestions are worth trying.

7.1 Audio

Currently our software does not have audio recording. The animations that elementary grade students make are usually simple enough for the teacher to understand even without an explanation. Furthermore, students can write text and annotate their animations for greater clarity. A few sentences written at the end of an animation are usually sufficient. We were hesitant to include audio in the initial design because we were unsure how much additional attention it would require for students to give a verbal explanation of their animation as they created the animation. It was also unclear what, if anything, would be gained by including audio recording. Having students summarize their explanation with text at the end of an animation kept the task relatively simple and had the additional benefit of making students think through what they had done in the process of writing the summary. If a teacher decided to show a particular animation to the class after looking through the submissions she had received, she would usually ask the student a few clarifying questions and sometimes even ask him or her to talk the class through the animation. This option seemed to work well, and from the classroom trials, there did not seem to be much that was lost from not having audio recording. It might be interesting from a pedagogical perspective, however, for a teacher to have the option of listening to the explanation a student gave as she made her animation. This capability may be particularly helpful if the teacher is reviewing work for an assignment she gave to the class, and the students are not available to describe or answer questions about their animations.

7.2 Better Visual Cues: Flashing Buttons

Currently the border of the animation area that is normally blue when the animation is playing back (see Figure 7-1) flashes red when a student is recording. This visual cue allows a student to keep track of what she is currently doing. Given the easy distractibility of younger children, it might be helpful if the button associated with the current state of the program flashes until a different task is initiated. The Record button would begin to flash, for example, when a student pressed Record and would continue to do so until the student tapped a different button, e.g., Rewind, and then that button would continue to flash until a different button was tapped. Such visual cues are useful because they help students keep track of what they are currently doing.

7.3 Greater Leeway: Allow Mistakes, Cut a Single Cube Out of an Array

One of the things that we noticed during classroom trials was that the software sometimes prevented students from making a pedagogically interesting mistake. A good example was when students were asked to create an animation showing what happens when we divide 19 by 5. The purpose was to show that we get 3, remainder 4. One student who was still learning about

remainders decided to “stretch” the array of length 4 so that it was the same length as the three arrays of length 5. She then gave her answer as $19 \div 5 = 4$. She noticed, though, that the array of length 4 was wider when she stretched it to be the same length as the three arrays of length 5. She wondered why they were not the same width. This situation occurred because our software was maintaining a specific aspect ratio for each array. For younger students less familiar with the concept of arrays, it might be better to enforce a rule that all cubes of all arrays on a page remain the same size, but to allow older more knowledgeable students to resize an array however they wish (no aspect ratio). In an effort to understand how two arrays that are the same length and width but have a different number of entries are different, the student learns an important lesson: Cutting a large object into two still gives two objects half the size of the original not two objects the same size as the original. This kind of flexibility in the software mimics real life more closely and makes the software a more useful tool for learning.

We also noticed that some students working on a proof about adding or subtracting one from a factor in a multiplication problem tried to cut single cubes out of multidimensional arrays and found that they could not—the software only enabled them to cut off an entire row or column. For younger students, this restriction is useful, especially if they are still learning exactly what an array is. Allowing older students to make such mistakes, though, leads to useful learning opportunities and enables the teacher to gain insight into what her students do and do not understand. Perhaps there could be different levels of flexibility that a teacher can specify that are suitable for students of particular ages and knowledge, and the software’s flexibility could be increased when used by older, more knowledgeable children.

7.4 Even Smaller Histories

Although we were able to achieve some reduction in the history size for animations by collecting contiguous position history items into a single class instance, it is possible for us to reduce the size even further. If we have a series of points (XY coordinates) representing the position of an object on the screen (sampled every Δt seconds), we might not need all the points to reconstruct the path followed by the object. If we have the object moving in a straight line at a constant speed, just knowing the starting position, the ending position and the time taken to move between the two points makes it possible to reconstruct the path followed by the object. For more complicated paths, we can store a representative subset of the points on the path and use a suitable method for interpolation between the points in the subset. This design would allow us to reduce the history size even further.

7.5 Create Animations Retroactively

Since users sometimes forgot to press Record before they began creating an animation, it would be nice if animations could be created retroactively. Most animations are relatively short, and it is possible to always keep a record of the latest actions performed by a user, perhaps the last 20

or so, especially since CLP already keeps a history of user interactions. By keeping such a history of actions, users who forgot to press Record before they created their animation could simply press Rewind and have this history of actions undone. Once the page was in the state at which they would have initially pressed Record to start creating their animation, they could press a different button and the history of items stored from that point onwards could be converted into the animation that they had initially intended to create.

8 Conclusion

The idea that it is possible to extract the regularities found in elementary grade mathematics and use those regularities to create a software program that enables students to create animations in order to communicate mathematical proofs is extremely powerful. Such a software program provides a good example of a new and innovative way to improve teaching and learning of mathematical proof in elementary grades.

The feedback we got from teachers who used the software in their classrooms was very positive. They genuinely thought that their students had benefitted from the software and learnt more than they otherwise could have. The students found it extremely easy to learn how to use the software and were usually able to figure out how to do what they wanted with minimal instruction. They also found the experience very engaging.

There are improvements that can be made to enhance the animation software but it is already very useful in its current state. We intend that future editions will be used more widely and hope that more students will reap the benefits.

9 References

- [1] Bloch, J. *Effective Java*. Upper Saddle River, NJ: Addison-Wesley, 2008. Print.
- [2] Collins, A. Toward a design science of education. In E. Scanlon & T. O'Shea (Eds.) *New directions in education technology*. Berlin, Germany: Springer, 15-22, 1992.
- [3] Design-Based Research Collective. Design-Based Research: an emerging paradigm for educational inquiry. *Educational Researcher*. 32 (1) 5-8, 2003.
- [4] Koile, K. and Rubin, A. Machine Interpretation of Students' Hand-drawn Mathematical Representations. *Workshop on the Impact of Pen and Touch Technology on Education (WIPTTE) 2013*.
- [5] Liskov, B. and Guttag, J. *Program Development in Java: Abstraction, Specification, and Object-oriented Design*. Boston: Addison-Wesley, 2000. Print.
- [6] National Council of Teachers of Mathematics, NCTM (2000). *Principles and Standards for School Mathematics*, Reston, VA.
- [7] Rubin, A., Storeygard, J, and Koile, K. Supporting Special Needs Students in Drawing Mathematical Representations. *Workshop on the Impact of Pen and Touch Technology on Education (WIPTTE) 2013*.
- [8] Roschelle, J. and Jackiw, N. Technology Design as Educational Research: Interweaving Imagination, Inquiry, and Impact. *Handbook of Research Design in Mathematics and Science Education*. Mahwah, New Jersey: Lawrence Erlbaum Associates. 777-797, 2000.
- [9] Russell, S. J., Schifter, D., and Bastable, V. *Connecting Arithmetic to Algebra: Strategies for Building Algebraic Thinking in the Elementary Grades*. Portsmouth, NH: Heinemann, 2011. Print.

Appendix

Appendix A provides a more detailed view of the functions of specific classes and UI elements that are present in the design of the animation software. It begins by specifying the actions caused by the buttons that were added to the menu bar and proceeds to look at the classes and interfaces that contain the code which is executed to perform these actions.

Ribbon Buttons

- 1) Proof
 1. Changes current page into an Animation Page.
 2. Instantiates CLPAnimationPage Class that backs Animation Page.
- 2) Scissors
 1. Changes cursor appearance.
 2. Pen/Mouse cuts arrays/objects instead of writing on them.

Software Design

New Classes

- (1) CLPAnimationPage Class
 - 1) Extends CLP Page.
 - 2) Fields
 1. Page History Past
 2. Page History Future
 3. Current action (record/edit, rewind, play, forward, replay, pause, done, submit)
 - 3) Methods
 1. Play
 - i. Disables editing of Animation Page (this includes disabling of the Page History Past and Page History Future).
 - ii. Calls parent CLP Page replay with Page History Past but no Page History Future.
 - iii. Afterwards, enables editing of Animation Page (if editing was previously enabled).
 2. Replay
 - i. Disables editing of Animation Page (this includes disabling of the Page History Past and Page History Future).
 - ii. Calls parent CLP Page replay with Page History Past and Future.
 - iii. Afterwards, enables editing of Animation Page (if editing was previously enabled).
 3. Record/Edit
 - i. Instantiates Page History Past and Future (mementos) if not already present.
 - ii. Enables editing of Animation Page (if disabled).

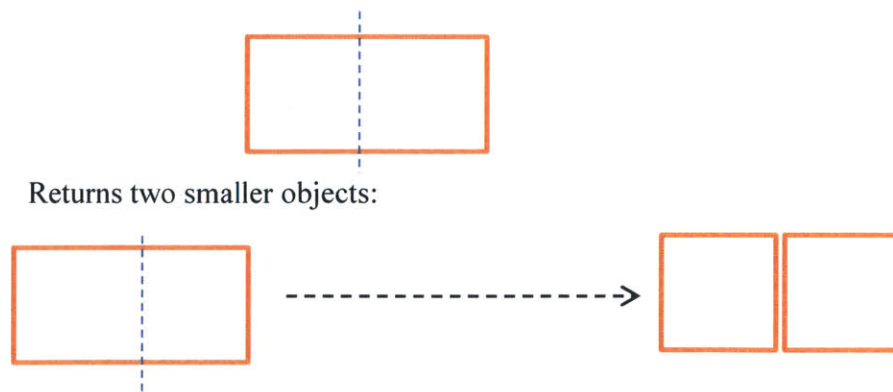
- iii. Overrides parent Page record method so that Future History items are not deleted when a student rewinds and then records.
- 4. Rewind
 - i. Uses parent CLP Page undo method.
- 5. Forward
 - i. Uses parent CLP Page redo method.
- 6. Pause
 - i. Marks items stored in history so that there is no wait period between them during play/replay/rewind.
- 7. Done
 - i. Disables editing of Animation Page Object.
- 8. Submit
 - i. Calls parent CLP Page submit.

New fields in existing Classes

- 1) boolean CutEnabled (CLPPPage.cs, CLPPPageViewModel.cs): Set to true in scissors mode

New Methods in existing Classes

- 1. CutObjects (CLPPPage.cs)
 - i. `public List<ObservableCollection<ICLPPPageObject>> CutObjects(double ave, double topY, double botY)`
 - ii. Called when user is in scissors mode and pen/mouse makes a continuous stroke across an object in Proof/Objects Area.
 - iii. Returns a list with two smaller objects for object across which user made a continuous stroke:



- iv. Delegates function of getting two smaller objects to the methods:
 - a) `SplitAtX: ObservableCollection<ICLPPPageObject> SplitAtX(double ave)`
 If this method is called in a CLPPPageObject that begins at x1 and ends at x2 on the x-axis, it returns two new shapes such that:
 - 1. The first shape begins at x1 and ends at ave.
 - 2. The second shape begins at ave and ends at x2.
 - b) `SplitAtY: ObservableCollection<ICLPPPageObject> SplitAtY(double ave)`

If this method is called in a CLPPageObject that begins at y1 and ends at y2 on the y-axis, it returns two new shapes such that:

3. The first shape begins at y1 and ends at ave.
4. The second shape begins at ave and ends at y2.

2. ReInkShapes (CLPPage.cs)

- i. `public void ReInkShapes(InkStrokes PageInkStrokes, Observable Collection<ICLPPageObject> AffectedPageObjects)`
- ii. This method removes Inkstrokes that were over page objects that have been replaced with new objects and reinserts them over the new page objects.