AN INTERACTIVE HIGH SCHOOL LAB FOR EXPLORING COGNITIVE LOAD THEORY

BY

DREW MOSE MAHRT

A Thesis Submitted to The Honors College

In Partial Fulfillment of the Bachelors degree

With Honors in

Computer Science

THE UNIVERSITY OF ARIZONA

December, 2011

Approved by:

____

Dr. Richard Snodgrass

Department of Computer Science

Abstract

Every day we continue to learn new information. Sometimes it might be through a formal classroom setting, and other times it might be through daily life experiences. Over the years, many theories and practices have been created to enhance and optimize the information that we learn. When we learn information, the ultimate goal is that we are able to retain this new knowledge in the hope that it can be used at a later time. Being exposed to massive amounts of information with no way to process or remember it defeats the goal of learning. Over the years, instructors have applied teaching techniques that use combinations of visual, audio, and spatial information in the hopes that students will learn information more easily as well as a higher retention rate. Cognitive Load Theory explores how these different combinations of techniques can either inhibit or enhance the learning process. This isn't just applicable in the classroom though; it can also be applied to the field of Computer Science. For Example, when designing the user interface for a program, the programmer should take into account how the different interface elements interact with each other (such as location, size and color). The goal of this thesis is to design a lab to give students in-depth knowledge of Cognitive Load Theory through reading and interactive examples, as well as how it applies to the information they learn. The goal of this lab is to effectively teach high school students about Cognitive Load Theory and its applications in Computer Science.

Table of Contents

1. Introduction

Over the years, many theories and practices have been created to enhance and optimize the information that we learn. When we learn information, the ultimate goal is that we are able to retain this new knowledge in the hope that it can be used at a later time. Being exposed to massive amounts of information with no way to process or remember it defeats the goal of learning. Over the years, instructors have applied teaching techniques that use combinations of visual, audio, and spatial information in the hopes that students will learn information more easily as well as a higher retention rate. Cognitive Load Theory explores how these different combinations of techniques can either inhibit or enhance the learning process. The following section details exactly what Cognitive Load Theory is, and how the labs are presented to students through the LoCuS software.

A) What is LoCuS?

First, it is important to understand the method by which the lab is presented to the students. The lab itself consists of an XML file supplemented by various graphics and data files. These files are processed by software called LoCuS. This stands for Laboratory for Computer Science. Basically, LoCuS provides the framework for the lab. All labs must follow a general guideline supplied by the software authors. This forces a standard layout across all labs, allowing for faster development as well as easier familiarity for students using multiple labs. More information on LoCuS can be found at http://www.cs.arizona.edu/projects/focal/ergalics/fieldguide/

B) What is Cognitive Load Theory?

Important definitions:

- Task: A specific goal for a user to accomplish, such as memorizing a sequence of colors for the game Simon.
- Modality: Different methods of conveying information to the user, such as images, sound, text, and spatial organization.
- Presentation: The way of displaying a specific modality. For instance, in the game Simon, one presentation can be the text displayed.
- Complementary combination: Presentations that work towards improving the learning process, such as showing the word “blue” which is also colored in blue text.
- Conflicting combination: Presentations that are aimed at hindering the learning process, such as showing the word “blue” which is colored in red text.
- Mental effort value: A number, produced through a short survey, representing how hard the user felt they had to work towards accomplishing the given task.
- Performance: A measure of how well the user accomplished the given task. For example, performance in the game Simon is measured by how many words the user can correctly reproduce in the sequence.
- Cognitive Load Value: A quantitative value which combines the effort put towards learning information (mental effort) with how well the information was actually learned (performance). In fact, the following equation combines these two values to measure cognitive [3]. The two z-values represent standardized scores for performance and mental effort.

$$E = \frac{z_{Performance} - z_{Mental\ Effort}}{\sqrt{2}}$$

Cognitive Load Theory predicts that any particular combination of modalities of given information for an individual will yield stable mental effort values. The theory also predicts that complementary combinations of presentations will give lower mental effort values and higher performance scores. However, conflicting combinations of presentations will give higher mental effort values and lower performance scores [3].

Figure 1 provides a graphical representation of how various types of presentations affect mental effort and performance, and how those affect cognitive load. For instance, the + arrow from the Complementary Presentations bubble to Performance means that as the number of complementary presentations increases, the performance increases. The colors of the arrows are to distinguish what node the source arrows are connected to.

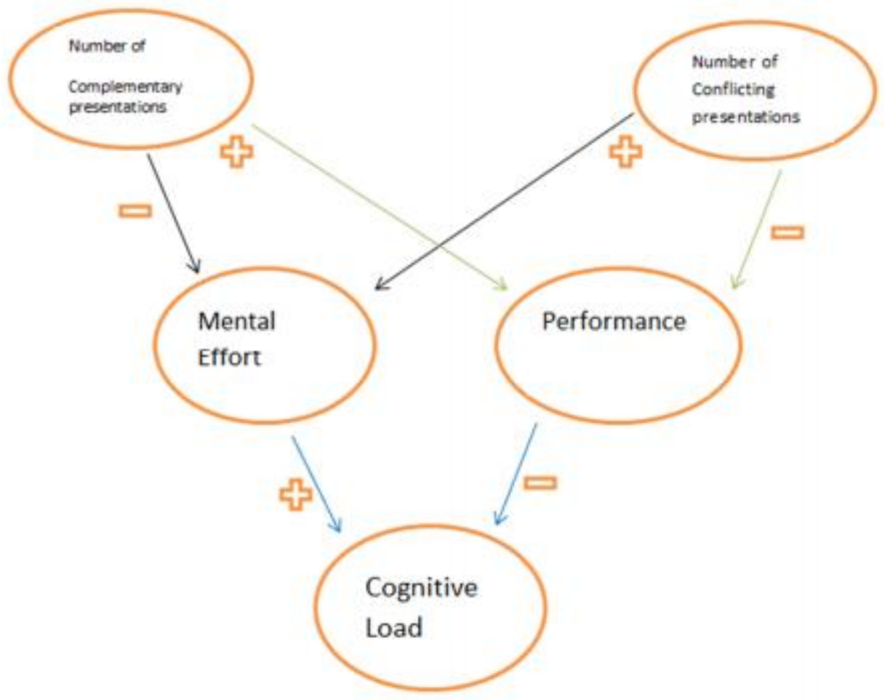

Number of Complementary presentations
Number of Conflicting presentations
Mental Effort
Performance
Cognitive Load
+
−
+
−
+
−

Figure 1 - This figure shows how an increasing or decreasing amount of a certain value affects the value the arrow is pointing to.

2. Simon Apparatus

In the following section, we will show how the Simon apparatus was planned and created, discuss problems in the development of the program, as well as show how the game is played.

A) Planning

After defining exactly what Cognitive Load Theory was, I needed to think of an effective and engaging way to show the students an example of the theory. I needed to take into account the restrictions present in a computer laboratory setting, which ruled out sound as a type of input. Eventually I came to the decision to create a modified version of the Milton Bradley game, also named Simon. In the original version, the physical game is a circle of buttons, colored green, red, yellow, and blue. The game presents the player with a sequence of colors by lighting the buttons while playing a unique tone for each button. The player then repeats the sequence by pressing the buttons in the same sequence. Each time the player correctly repeats the sequence, the game increases the sequence by adding another color. This continues until the player loses.

My version of the game follows a similar concept of repeating a given, constantly increasing sequence. However, I had to eliminate the sound. I adapted the game by having Simon present the sequence to the player in the form of black text. Since I was trying to show how Cognitive Load Theory could be applied to the game, two additional game modes were added. A complementary input mode was achieved by changing the font color to match the word it represented. Similarly, a conflicting input mode was achieved by changing the font color to a different color than the word represented. For example, the text "Red" could be displayed in a green font.

Execution

Simon was created using a section of the LoCuS framework called an Apparatus. An Apparatus is an abstract class which provides the basic tools to create an interactive program. It follows the Model View Controller (MVC) design pattern [1]. I was provided with the basics of an apparatus window and the model behind it. My apparatus, Simon, extended the Apparatus class. This allowed me to access all of the built-in apparatus functions, which made the writing of Simon itself much easier. Additionally, using the Apparatus class made the insertion of Simon into my lab a simple few lines of code. From there, I wrote 652 lines of code to create Simon.

The view consisted of three main screens. First was the main menu, which allows the player to select plain black text, complementary colors, or conflicting colors. The second screen displayed the sequence of colors to the player. The third screen shows a picture of a square containing the four colors, and requires the player to use the mouse to press each color in the same sequence. The game alternates between the second and third screen until an incorrect sequence is entered, at which point the player loses.

The following figure (figure 2) shows the UML [1] diagram for Simon.

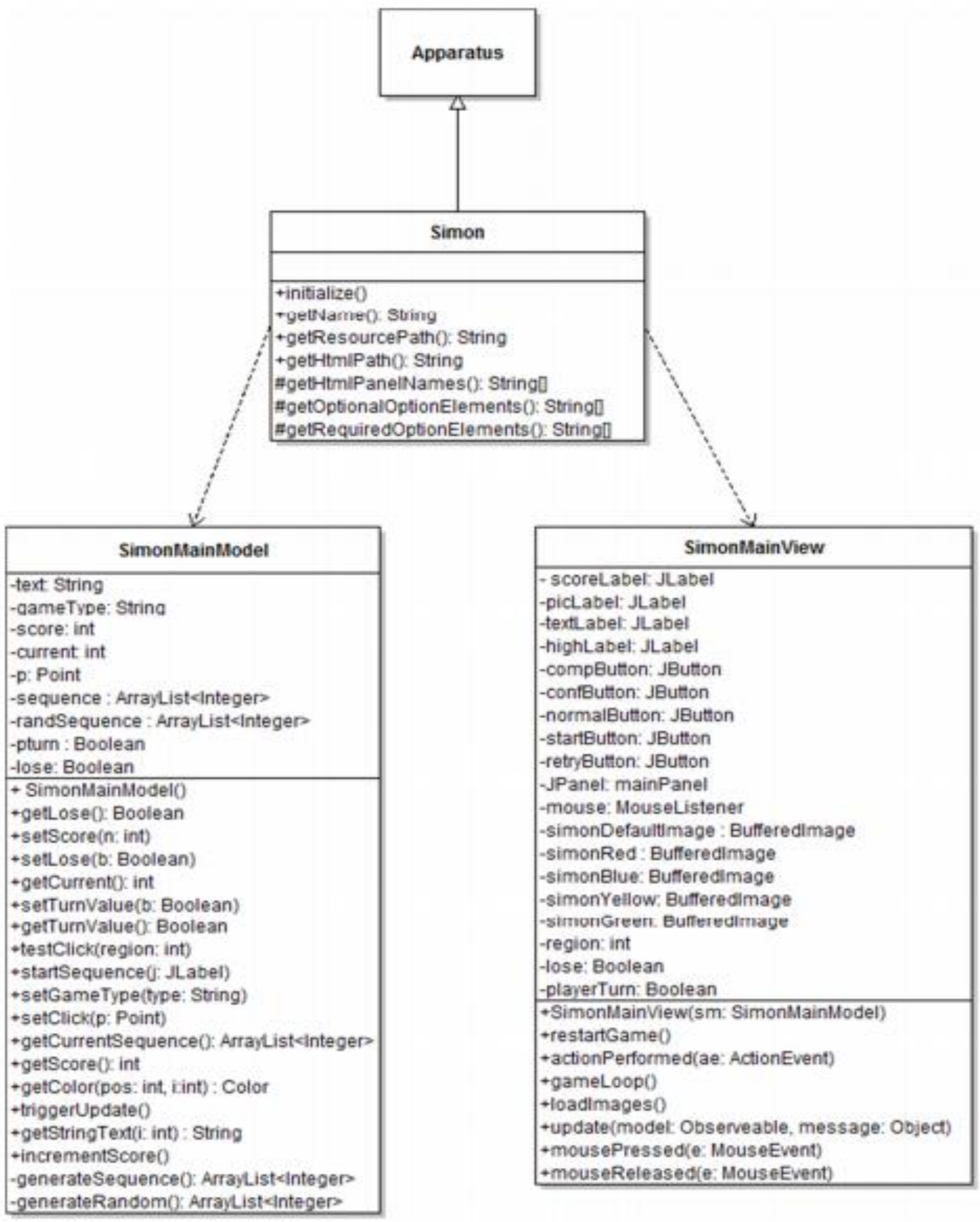

Apparatus
Simon
+initialize()
+getName(): String
+getResourcePath(): String
+getHtmlPath(): String
#getHtmlPanelNames(): String[]
#getOptionalOptionElements(): String[]
#getRequiredOptionElements(): String[]
SimonMainModel
-text: String
-gameType: String
-score: int
-current: int
-p: Point
-sequence : ArrayList<Integer>
-randSequence : ArrayList<Integer>
-pturn : Boolean
-lose: Boolean
+ SimonMainModel()
+getLose(): Boolean
+setScore(n: int)
+setLose(b: Boolean)
+getCurrent(): int
+setTurnValue(b: Boolean)
+getTurnValue(): Boolean
+testClick(region: int)
+startSequence(j: JLabel)
+setGameType(type: String)
+setClick(p: Point)
+getCurrentSequence(): ArrayList<Integer>
+getScore(): int
+getColor(pos: int, i:int) : Color
+triggerUpdate()
+getStringText(i: int) : String
+incrementScore()
-generateSequence(): ArrayList<Integer>
-generateRandom(): ArrayList<Integer>
SimonMainView
- scoreLabel: JLabel
-picLabel: JLabel
-textLabel: JLabel
-highLabel: JLabel
-compButton: JButton
-confButton: JButton
-normalButton: JButton
-startButton: JButton
-retryButton: JButton
-JPanel: mainPanel
-mouse: MouseListener
-simonDefaultImage : BufferedImage
-simonRed : BufferedImage
-simonBlue: BufferedImage
-simonYellow: BufferedImage
-simonGreen: BufferedImage
-region: int
-lose: Boolean
-playerTurn: Boolean
+SimonMainView(sm: SimonMainModel)
+restartGame()
+actionPerformed(ae: ActionEvent)
+gameLoop()
+loadImages()
+update(model: Observeable, message: Object)
+mousePressed(e: MouseEvent)
+mouseReleased(e: MouseEvent)

Figure 2 – The Apparatus is the superclass for Simon. SimonMainModel and SimonMainView make up the parts of the Model/View/Controller.

C) How To Play Simon

Follow these steps to play Simon.

1. Choose one of the three game modes from the main menu (see figure 3).

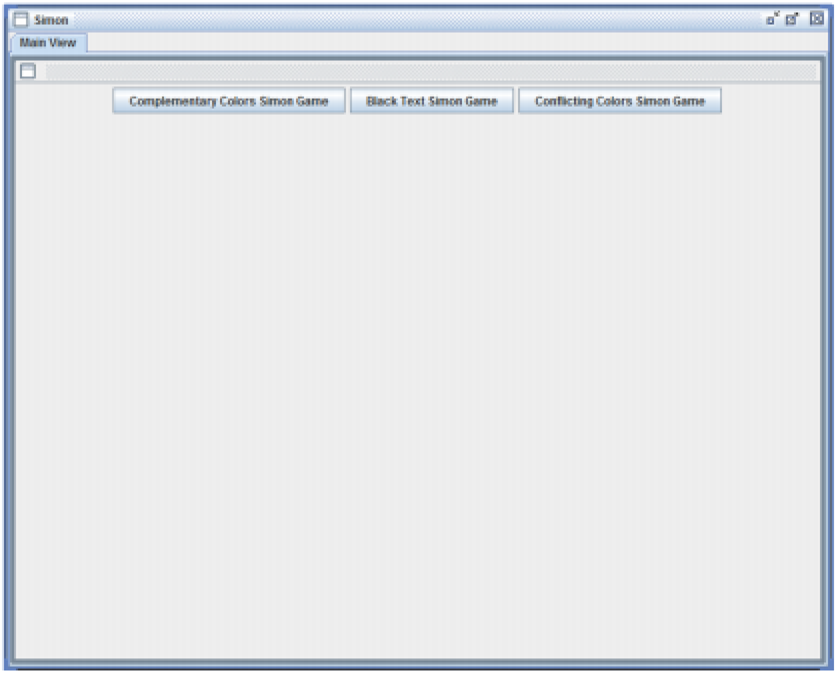

Simon
Main View
Complementary Colors Simon Game
Black Text Simon Game
Conflicting Colors Simon Game

Figure 3 – The main menu for Simon. Choose which game mode to play.

2. Press the "Go!" button to start the game (see figure 4).

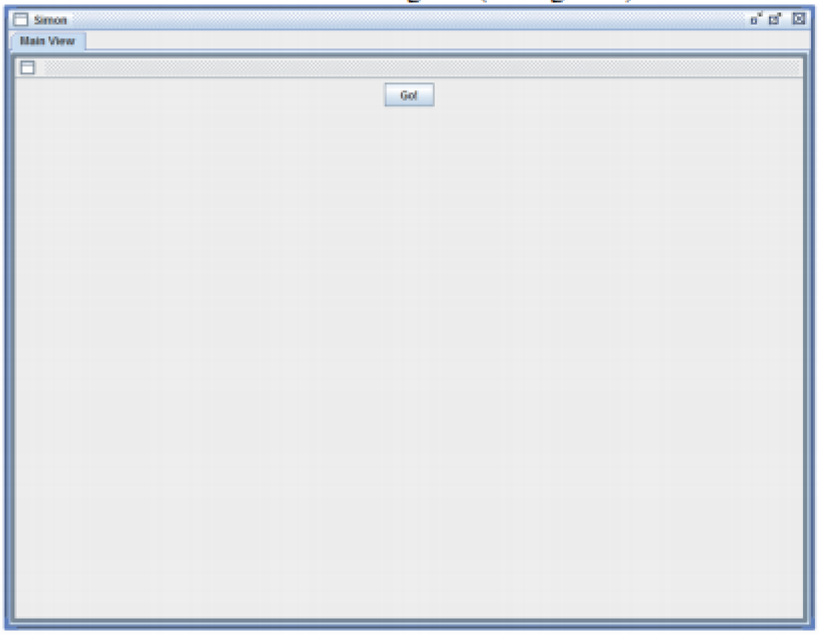
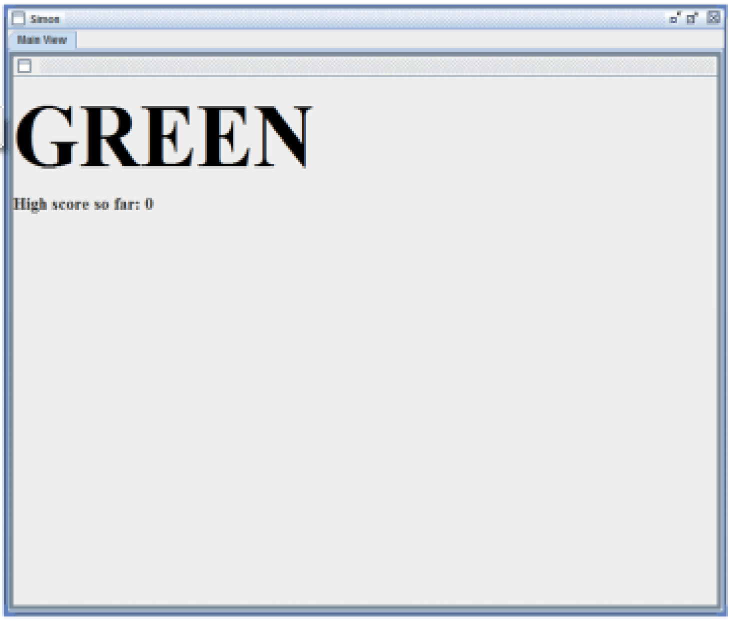


Figure 4 – Once the mode is selected, press "Go" to start the game.

3. Watch the sequence being displayed (see figure 5).

ie document

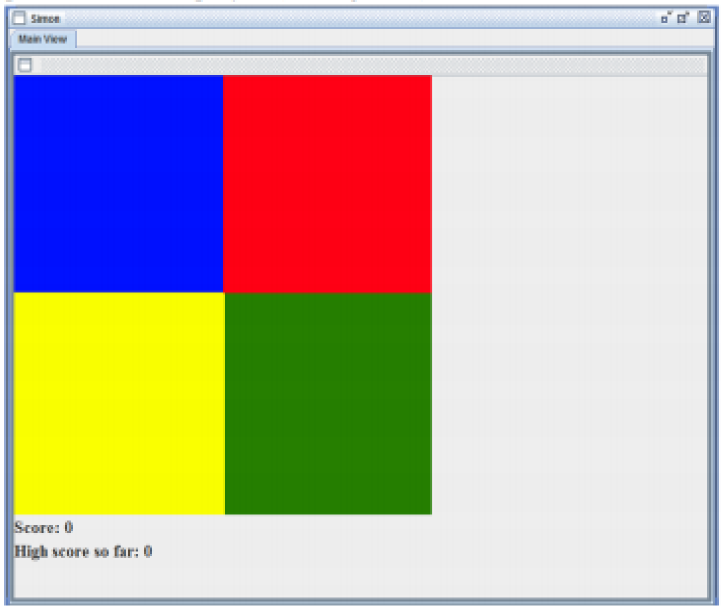


GREEN

High score so far: 0

Figure 5 – This shows one of the colors in the sequence being displayed.

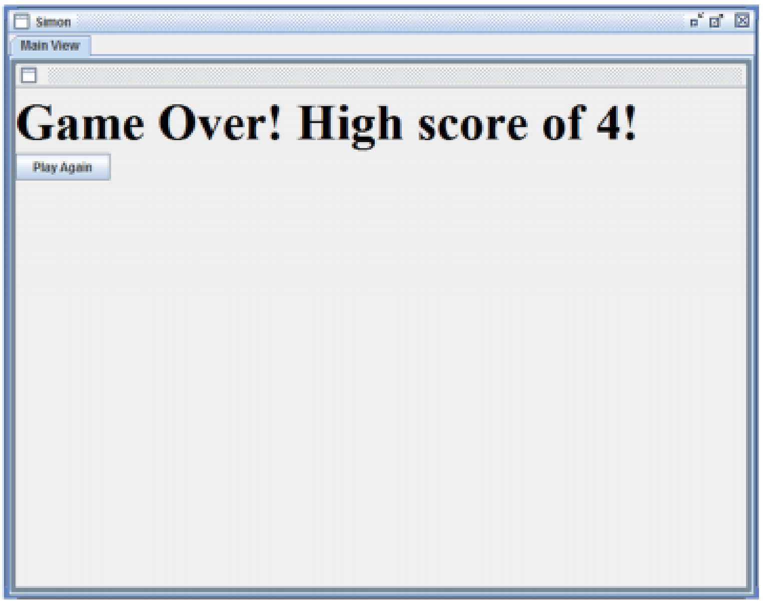
4. Repeat the sequence by pressing the colored squares in the same order that the sequence was presented to the player (see figure 6).



Score: 0
High score so far: 0

Figure 6 – The player must click the squares in the same order as the sequence appeared.

5. Repeat steps 3 and 4 until the player repeats the sequence incorrectly (see figure 7).



Game Over! High score of 4!

Figure 7 – This screen is shown when the player incorrectly repeats the sequence, and offers them the chance to play again.

D) Testing

After Simon was completed, I needed to test it to confirm that it would demonstrate behavior consistent with cognitive load theory. I had five university students attempt the game with conflicting colors, and then with complementary colors. As I had hoped, the scores of the complementary colors were much higher than the scores of the conflicting colors. In one case, there was a 40 point increase when the complementary colors were used. The smallest increase in score was 12 points.

E) Problems

I ran into two major problems while creating Simon. The first problem involved the timing between the model processing each color from the sequence, and the view showing the sequence to the user. The model finished its processing before the view could show every color to the user, resulting in the user missing colors at the end of the sequence. This was solved by putting both of these functions in a separate thread, which allowed them to run independently of the rest of the game.

The second problem was when the sequence contained two identical colors in a row. For instance, if green was followed by another green, the player wouldn't be able to distinguish the fact that it was in fact two colors presented rather than one. To fix this, on every other color I added a period before the text. This provided the effect of the text slightly shifting as each color in the sequence was presented.

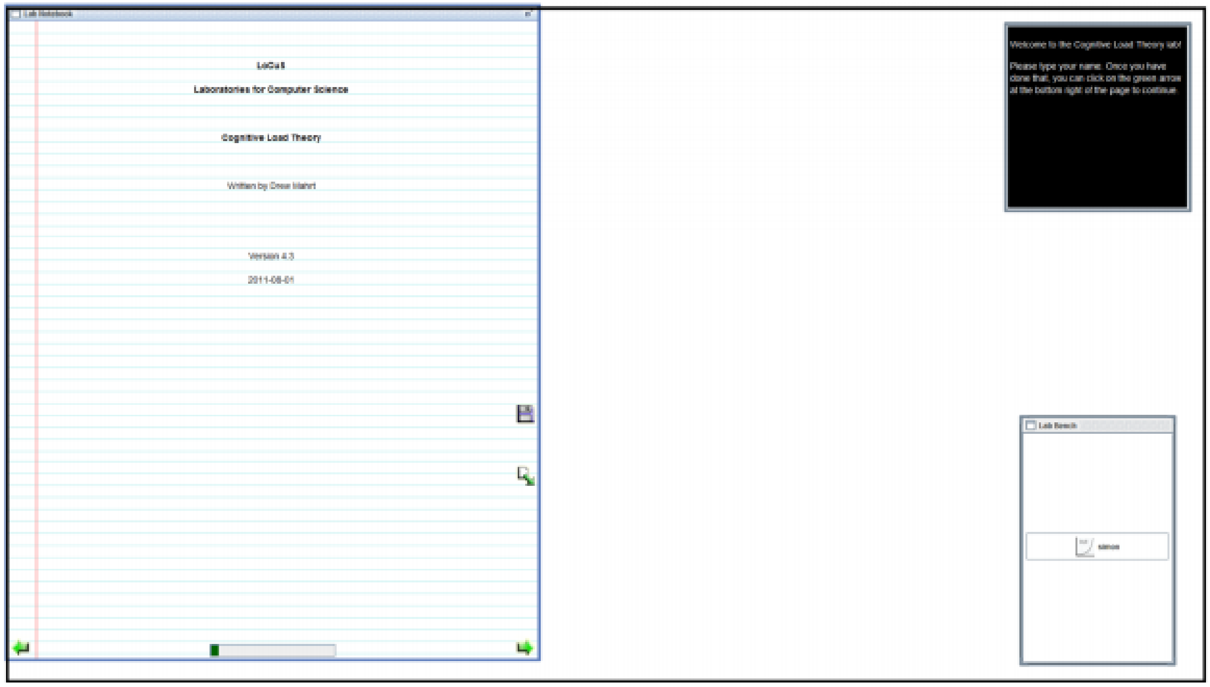
3. Cognitive Load Lab

This section shows the development process for the Cognitive Load Lab. It runs though the planning, execution, testing, and troubleshooting phases of creating the lab.

A) Planning

Planning the lab itself was a careful process which went through many revisions. I had to put myself in the mindset of a high school student trying to learn the key concepts of a theory I spent months studying, as well as providing an example of its applications in Computer Science. The overall structure of a lab was given in a guide provided by the LoCuS developers [2]. It detailed the concept of having a pre-lab which introduced the basic concepts, and then the full lab which went into much further detail and tried to test the students on what they were trying to learn. I started by presenting an overview of what the theory was, and what I hoped the student would accomplish after completing the lab. Next, I had to split the lab section into parts which would progressively teach the student the concepts behind the theory separately, building towards the goal of having a more comprehensive knowledge of Cognitive Load after concluding the lab.

For reference, below is a screenshot showing what the opening screen (figure 8) and pre-lab section of the lab looks like (figure 9). All pages in the lab follow a similar format.



LoCuS

Laboratories for Computer Science

Cognitive Load Theory

Written by Drew Mahrt

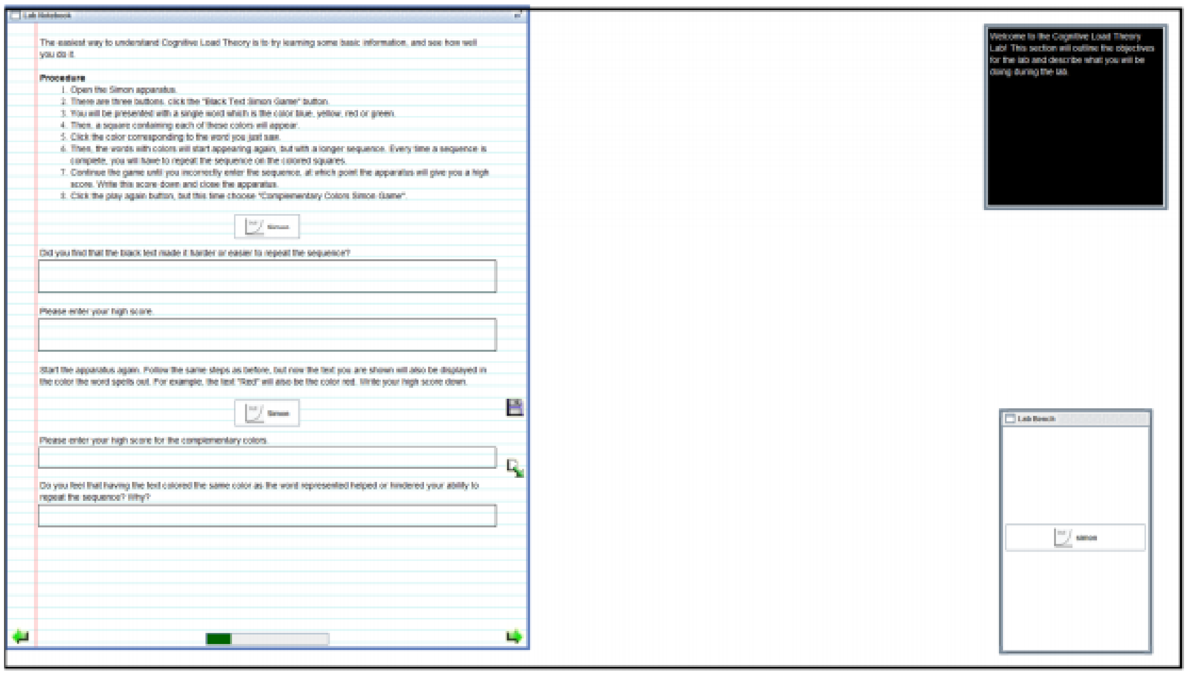
Version 4.3

2011-08-01

Welcome to the Cognitive Load Theory lab!

Please type your name. Once you have done that, you can click on the green arrow at the bottom right of the page to continue.

Figure 8 – This is the front page of the lab notebook. The lab information is displayed on the left. The window in the bottom right shows the apparatus used in the lab. The black box in the top right corner is the Chalkboard. This is used by the lab creator to provide extra information to the student which isn't included in the actual lab notebook.



The easiest way to understand Cognitive Load Theory is to try learning some basic information, and see how well you do it.

Procedure

1. Open the Simon apparatus.
2. There are three buttons, click the "Black Text Simon Game" button.
3. You will be presented with a single word which is the color blue, yellow, red or green.
4. Then, a square containing each of these colors will appear.
5. Click the color corresponding to the word you just saw.
6. Then, the words with colors will start appearing again, but with a longer sequence. Every time a sequence is complete, you will have to repeat the sequence on the colored squares.
7. Continue the game until you incorrectly enter the sequence, at which point the apparatus will give you a high score. Write this score down and close the apparatus.
8. Click the play again button, but this time choose "Complementary Colors Simon Game".

Did you find that the black text made it harder or easier to repeat the sequence?

Please enter your high score.

Start the apparatus again. Follow the same steps as before, but now the text you are shown will also be displayed in the color the word spells out. For example, the text "Red" will also be the color red. Write your high score down.

Please enter your high score for the complementary colors.

Do you feel that having the text colored the same color as the word represented helped or hindered your ability to repeat the sequence? Why?

Welcome to the Cognitive Load Theory Lab! This section will outline the objectives for the lab and describe what you will be doing during the lab.

Figure 9 – This screen shows the procedures for using Simon, and then asks the student to answer various questions about the experiment they just performed.

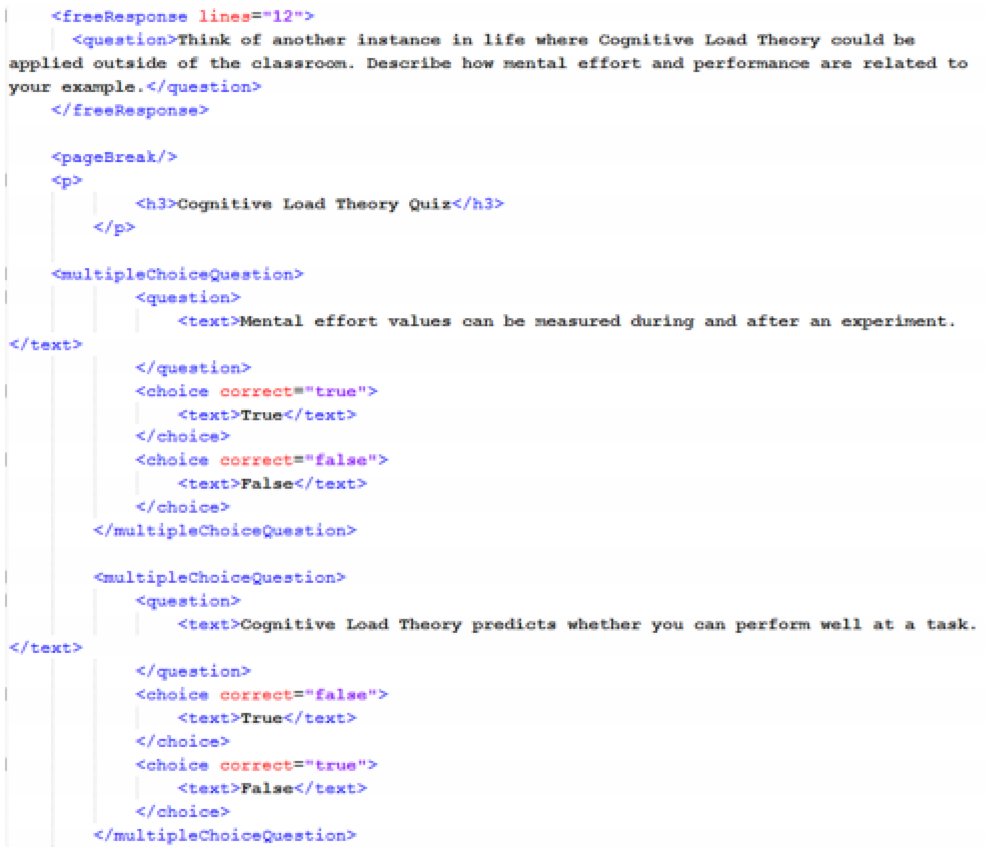
The lab was organized into the following sections:

a. Introduction: “Discovering How We Learn”
b. Pre-Lab: “What is Cognitive Load Theory”
c. “Important Definitions”
d. “Formal Theory”
e. “Measuring Cognitive Load”
f. “Measuring Mental Effort”
g. “Relationship Between Mental Effort, Performance, and Cognitive Load”
h. “Applications in Computer Science”
i. “Cognitive Load Theory Quiz”

B) Execution

The lab itself is completely contained within an XML file, which is supported by various images and data files. The XML file is created in a very specific format provided by the LoCuS developers in the Lab Author’s Guide [2]. Various XML tags are used to define the welcome screen, pre-lab, lab, how to launch the apparatus, and how to quiz the students on information they were learning. By following the guide correctly, the actual writing of the lab was extremely easy and straightforward. The difficult part of the execution was creating the lab in such a way that it taught the necessary information using a good balance between text, images, and the apparatus.

To give a better idea of what the XML code looks like, and how the tags are organized, see the image below (figure 10). In the figure, the code for presenting a quiz to the students is shown. In addition to the questions themselves, the tags allow for options to be set for each type of question.



```
    <freeResponse lines="12">
      <question>Think of another instance in life where Cognitive Load Theory could be
applied outside of the classroom. Describe how mental effort and performance are related to
your example.</question>
    </freeResponse>

    <pageBreak/>
    <p>
            <h3>Cognitive Load Theory Quiz</h3>
        </p>

    <multipleChoiceQuestion>
            <question>
                <text>Mental effort values can be measured during and after an experiment.
</text>
            </question>
            <choice correct="true">
                <text>True</text>
            </choice>
            <choice correct="false">
                <text>False</text>
            </choice>
        </multipleChoiceQuestion>

        <multipleChoiceQuestion>
            <question>
                <text>Cognitive Load Theory predicts whether you can perform well at a task.
</text>
            </question>
            <choice correct="false">
                <text>True</text>
            </choice>
            <choice correct="true">
                <text>False</text>
            </choice>
        </multipleChoiceQuestion>
```

Figure 10 – An example of XML code which displays a quiz to the student.

C) Problems

Problems arose from the creation of the content in the lab itself. I had very little experience having to teach a complicated concept to younger students. My original version of the lab was too advanced for a high school level, and didn't utilize the apparatus enough. After various revisions, I believe that the lab successfully conveys the theory at an appropriate level while providing the student with an enjoyable experience.

4. Conclusions and Future Work

Although there wasn’t time to test the lab on high school students, the lab does present all of the goals it aimed to achieve. I believe that the lab successfully introduces the student to Cognitive Load Theory, and exposes them to the idea that it is applicable in many settings, more specifically Computer Science. In the future, when students test the lab in its beta form, feedback will be considered and the lab will be edited accordingly. Since the lab is meant to be used by high school students, feedback on its effectiveness is paramount to producing a final form of the lab for full deployment in many classrooms. Like many forms of teaching, the lab must continue to evolve over time in order to maintain its relevance and effectiveness. Even though a timeline for when this final product will be available is unknown at this point, beta testing can begin soon.

5. Acknowledgements

I would like to thank Dr. Richard Snodgrass for leading me through this wonderful learning process during the past year. He provided me the opportunity to expand my concepts of computer science outside of the classroom setting and showed me its applications in real-life situations. Without his help, none of this would have been possible.

I would also like to thank other members of the LoCuS team, Sam Martin, Rob Trame, Gavin Simons, Jaimie Sauls, and Andrey Kvochko for providing software support as well as brilliant minds to help develop my ideas.

6. Bibliography

[1] Booch, G., Jacobson, I., & Rumbaugh, J. (1998) The Unified Modeling Language User Guide.

[2] Cheepurualli, R., Gephart, N., Johnston, M., & Snodgrass, R. Lab Author's Guide for LoCuS Project. TAU Technical Document 11, October 2011.

[3] Paas, F., Tuovinen, J., Tabbers, H., & Van Gerven, P. (2003) Cognitive Load Measurement as a Means to Advance Cognitive Load Theory. *Educational Psychologist*. 38(1), 63-71.