Gamification and Serious Game Approaches for Introductory Computer Science Tablet Software

Kevin Browne
McMaster University
1280 Main Street West Hamilton, ON L8S 4K1
brownek@mcmaster.ca

Christopher Anand
McMaster University
1280 Main Street West Hamilton, ON L8S 4K1
anandc@mcmaster.ca

ABSTRACT
In this paper, we overview the design of tablet apps built to teach introductory computer science concepts, and present the results and conclusions from a study conducted during a first year computer science course at McMaster University. Game design elements were incorporated into the apps we designed to teach introductory computer science concepts, with the primary aim of increasing student satisfaction and engagement. We tested these apps with students enrolled in the course during their regular lab sessions and collected data on both the usability of the apps and the student’s understanding of the concepts. Though overall we found students preferred instruction with the apps compared to more traditional academic instruction, we found that students also recommended combined instruction using both traditional methods and the apps in the future. Based on this we conclude that gamification and serious game design approaches are effective at increasing student satisfaction, and make several recommendations regarding the usage and design of educational software incorporating game design elements.

Author Keywords
Gamification; Serious games; E-learning; M-learning; Tablet software usability;

ACM Classification Keywords
H.5.m. Information Interfaces and Presentation (e.g. HCI): Miscellaneous

General Terms
Human Factors; Design; Measurement.

INTRODUCTION
The first year course CS1MA3 was the introductory computer science course in the Department of Computing & Software at McMaster University. The course was a breadth approach to introductory computer science, with students taught different algorithms and how to derive their own algorithms, the basics of functional programming, binary numbers, graphs, and the basics of CPUs and assembly language. All of the ideas are presented with the overall goal of developing the students’ problem solving skills in the context of computer science.

Though the course was recommended for students wishing to pursue computer science in upper years, students from a diverse selection of backgrounds took the course as an elective. In past years, many students have found themselves unprepared for what a “computer course” really entails at the university level. Many students have never been exposed to basic computer science, have never written a computer program or have been taught only the mechanical aspects of programming. Such students experience frustration and dissatisfaction when they are expected to solve new types of problems and use new ways of thinking. This dissatisfaction often leads to disengagement during instruction, and failure to advance in the program.

The authors of this work are also involved in an outreach program at McMaster University that visits elementary school classrooms with activities meant to give students a fun and engaging example of computer science. One of the most successful activities in this program is one that we developed internally; what made this activity successful was how well it engaged students. The activity involved having the students write computer code to fix a simple table tennis video game that was initially presented to the students as being ‘broken’ (i.e. balls would fly through paddles and off the screen because the correct code was not yet present). Our observation and intuition suggested it was the table tennis game element that led to this higher satisfaction and engagement. We were motivated by these observations to try and reproduce this higher satisfaction by introducing educational software containing game design elements into the instruction of CS1MA3.

Our primary aim in conducting this study was to investigate whether integrating educational software incorporating games and/or game design elements into the instruction of introductory computer science could increase student satisfaction and engagement relative to traditional instructional methods. Secondary aims included examining what specific game design features of the apps are effective at increasing satisfaction, whether using the apps leads to a better understanding of the related concepts, and how such educational software can best be integrated into instruction of the concepts.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Gamification2013, October 2—4, 2013, Stratford, ON, Canada. Copyright 2013 ACM 978-1-XXXX-XXXX-X/XX/XX...$10.00.
As a result we designed and created the apps featured in this study, empirically tested them during the Fall 2012 iteration of CS1MA3, and present the results here in a manner in which we hope is reproducible for others teaching these and similar concepts. Based on the strong endorsement of the study participants, we will continue to use all of the relevant applications in the successor course, CS1JC3 (Computational Thinking). We have also used the study feedback to improve the game elements of the binary numbers app, which is currently being used successfully with over 2000 elementary school students as part of the NSERC PromoSCIENCE-supported outreach workshop “Software: Tool for Change”.

In this section we will first overview some terminology and related work.

The potential for tablets as a learning tool is an active area of research and the positive observations reported thus far [2] in using tablets in the classroom, and more importantly the potential for contributing further findings to the existing literature motivated our decision to use iPads for this study.

The “usage of game design elements to motivate user behaviour in non-game contexts” is known as gamification, and is considered by Deterding et al to be a concept warranting deeper empirical exploration [3]. Deterding et al define a serious game to be “a full fledged game for non-entertainment purposes”, where as “gamified applications merely incorporate elements of games” and acknowledge that ‘the boundary between a ‘game’ and an ‘artifact with game elements’ can often be blurry’. Our apps incorporate game design elements to varying degrees, and as such our work by its nature involves some examination of the intersection between gamification and serious games.

There is little literature at present covering gamification and / or serious game approaches to educational tablet software, but what work is available shows encouraging results [4, 5]. Popular educational websites focusing on interactive online programming that use gamification in their design include Codecademy (Codecademy.com) and Khan Academy (Khanacademy.org). However, both websites are focused on teaching computer programming, where as our apps are focused on teaching other computer science concepts.

APPLICATION DESIGN

In this section we will overview the six apps designed to teach binary search, binary numbers, CPU / assembly language, polynomial graphs, quicksort and Dijkstra’s algorithm. A common design element across all of the apps is the incorporation of interactive representations of the relevant concepts. Game elements such as objectives, rewards, penalties, levels, narrative, multiplayer, and in-game assistance are incorporated into the apps to varying degrees. We define in-game assistance to occur when the app provides corrective help or feedback after the user has failed to make the correct actions.

Binary search app

The binary search app shown in Figure 1 begins with a row of coconuts on the bottom half of the screen, and a message telling the user to ‘crack the coconuts to find the golden egg’. When the user taps on a coconut an arrow pointing either left or right appears, depending on which direction the coconut is located. If the user does not follow the correct sequence of taps for binary search, it will be suggested to the user that there is another way of finding the golden egg.

![Figure 1. Binary search app](image)

If the user follows the correct sequence of taps for binary search 5 times in a row, they will be rewarded with a treasure chest that they can open by tapping. When the user taps on the treasure chest they are treated to a screen that tries to explain why a maximum of 4 taps are required to reach the golden egg by showing the user the binary search tree. The app goes on to pan across a series of alternative search trees before exiting automatically.

Binary number app

The binary number app shown in Figure 2 uses both animations of binary number conversion and arithmetic as well as a multiplayer network game to assist in the instruction of binary number concepts. The app contains three different pages: conversion, arithmetic, and more (which contains the game).

The conversion page allows the user to select a decimal number using a slider on the bottom of the screen, and the number is shown converted live into binary at the top of the screen. By clicking one of two arrows the user is able to see an animation of the number being converted from one base to another in either direction.

The arithmetic page allows the user to modify two binary numbers by clicking each digit back and forth between ‘0’ and ‘1’. The user can then select either addition or subtraction and play an animation of the arithmetic taking place.

If the iPad is connected to a wifi network with other iPads using the app, the game screen allows the player to search for others to play the game. The game involves both players first tapping spaces in a grid to represent different binary numbers. The game then sends the decimal numbers that were created in the process to the other player for them to convert into
the correct grid. The first player to convert the other player’s numbers wins the game.

**CPU app**

The CPU app involves a single screen containing a visual representation of memory, registers, operators and a set of instructions as shown in Figure 3. The users execute each instruction by performing actions such as dragging data from memory to registers or registers to memory. In the case of operations, users drag the appropriate operation to an arithmetic logic unit together with register arguments before the result is computed, and must be dragged into the appropriate register.

Correct actions advance the user to the next instruction along with playing an approving sound effect as a reward, and incorrect actions result in a disapproving sound effect as a penalty. There are 3 ‘levels’ total consisting of 3 different sets of instructions for the user to work through.

If at any point the user hasn’t made any sort of action within a few seconds, a ‘Need Help?’ cartoon person appears. By clicking on this the app will animate the correct action automatically and advance to the next instruction.

**Polynomial app**

The polynomial app shown in Figure 4 begins with a blank screen and a one line textbox that users are told to tap. When the user taps on the textbox they are presented a calculator-style menu which allows the user to enter a polynomial.

After the user enters the polynomial they can create the associated graph by dragging the operators, numbers and symbols down into the blank space. When an operator is dragged into the space below there are blank nodes connected to either side to allow the user to build a tree by connecting more operators, numbers or symbols as leaves. If the user makes a mistake they can drag the tree into the garbage bin to start over. If the user wants an entirely new polynomial expression they can tap the ‘reset’ button. The user can optionally turn on additional parenthesis to outline the order of operations.

The user can check whether the polynomial graph is correct or not by tapping a ‘check’ button. If the graph is incorrect the user will receive a red X, and if the graph is correct the user will receive a green check mark instead.

**Quicksort app**

The quicksort app shown in Figure 5 allows the user to step through a demonstration and explanation of quicksort by selecting pivot points and tapping Next. The app starts off with a list of 5 unordered numbers and the user is told to select a pivot upon tapping the ‘Start/Next’ button. Depending on the pivot that the user selects, by clicking on Next repeatedly the app explains step-by-step the swaps that take place in between the need to select a new pivot.
As the list is broken into sublists to which quicksort can be applied, the graph of arrows in the space below is filled in as required. A red X is used to represent empty sublists. The user can click ‘Reset’ at anytime to start off with a new unordered list.

Figure 5. Quicksort app

Dijkstra app
The Dijkstra app shown in Figure 6 starts off with a mostly blank screen that the user is told to tap in order to place the nodes. The first time the user taps the screen a node in the shape of the McMaster University engineering department ‘fireball’ logo is placed as the initial node. The user can continue to tap the screen at different positions to place more nodes. The only restriction on the number of nodes that the user can place is how closely nodes can be placed; the app will just not place down a node if the user taps too closely to an existing node.

Once the user is finished placing the nodes, they can use ‘Start’, ‘Back’ and ‘Next’ buttons to step through a graphical trace of the algorithm’s execution on the graph that they just created. A node that has been visited is marked green, while an eye cartoon and a blue path in the direction of a node are used to indicate the current node and unvisited nodes which are having their tentative distances calculated.

App comparison
The different game design elements incorporated into the apps are presented in Table 1. Notably the quicksort and Dijkstra apps do not contain any game design elements, other than interactive representations of the concepts, a design approach common to all of the apps. We consider an objective to be a goal presented by the app that the user is expected to complete. The polynomial, quicksort and Dijkstra apps do not contain explicit objectives, but rather allow for free form improvisational experiences with the concepts. The binary search, binary numbers and CPU apps all do contain explicit objectives or goals that create a more proper game.

In the case of the binary number app, only a subsection of the app contains a game with explicit objectives, with the other two subsections allowing for improvisational experiences with the concepts. As such, the binary search and CPU apps may be considered serious games, with the binary numbers app containing a serious game, and the remainder of the apps more accurately described as instances of gamification rather than serious games.

<table>
<thead>
<tr>
<th>App</th>
<th>Game Design Elements</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Objectives</td>
</tr>
<tr>
<td>Binary Search</td>
<td>X</td>
</tr>
<tr>
<td>Binary Numbers</td>
<td>X</td>
</tr>
<tr>
<td>CPU</td>
<td>X</td>
</tr>
<tr>
<td>Polynomial</td>
<td>X</td>
</tr>
<tr>
<td>Quicksort</td>
<td></td>
</tr>
<tr>
<td>Dijkstra</td>
<td></td>
</tr>
</tbody>
</table>

Table 1. Game design elements incorporated into each iPad app.

EXPERIMENT DESIGN
In addition to attending lecture sessions, CS1MA3 students each attend a weekly lab 50 minutes in duration. Six weeks of lab sessions were set aside during the course for experiment sessions using the six iPad apps, with each of these six weeks set aside for an experiment with a specific iPad app. Each of the six weeks of sessions itself contained six sessions total; participants were assigned to go to a specific session each
Students were given the same quiz. If students had just

Students in labs which started by using the apps were given

Students were then given either 20 minutes of instruction in

Students were given a short (i.e. 2 minutes) quiz to fill out

The following procedure was followed during each experi-

Experiment session protocol

The participants were also asked to rate their overall performance in any past mathematics courses at any level of study from 1-10, with 1 being “failing performance overall” and 10 being “perfect performance overall”. They were asked to do the same for their overall performance in any past computer science courses at any level.

The participants were also asked to rate their expertise with the following different media and interfaces: mobile phones, mobile video games, iPad / other tablets - general usage, iPad / other tablets - video games, and touch screen interfaces. Participants rated their expertise by selecting one of the following expertise levels, based on descriptions of the expertise levels: ‘no expertise’, ‘some expertise’, ‘typical expertise’, ‘above average expertise’, and ‘expert’. For analysis purposes these expertise levels were assigned the numeric values 1-5 from no expertise to expert.

Usability survey

The participants were asked to rate how much they agree (Likert scale) with the following statements:

S1 The software was easy to use.
S2 It was difficult to learn how to use the software.
S3 I enjoyed using this software.
S4 I thought there was too much inconsistency in this software.
S5 The touchscreen finger gestures required to manipulate the software felt natural.
S6 The software was too slow to use efficiently.
S7 The graphics in the software were effective.
S8 I found the iPad uncomfortable to hold while using the software.
S9 The software helped me to learn the concepts.
S10 I felt confused trying to understand the concepts with this software.
S11 I feel confident that I understand the concepts.
S12 I would not recommend that others try to learn the concepts by using this software.

The participants could choose from: strongly disagree, somewhat disagree, neutral, somewhat agree, and strongly agree.

The design of the usability survey was inspired by features of the System Usability Scale (SUS) [1]. In a similar manner to SUS a usability score can be calculated, by assigning the average response of each statement a score contribution from 0-4. The odd numbered statement’s score contribution is a value of 0-4 from strongly disagree to strongly agree, and the even numbered statement’s score contribution is a value of 4-0 from strongly disagree to strongly agree. Dividing the sum of the scores by 48 and multiplying the result by 100 gives a score with a range from 0 to 100.

The usability survey also contained two questions, each with a blank space for feedback: “What did you like about the software?” and “What didn’t you like about the software?”.

Post-experiment questionnaire

The following questions were asked on the post-experiment questionnaire.

1. Did you prefer learning the concept(s) taught during the tutorial with the educational software or without the educational software? (circle one)
2. Why was this your preference? (blank space)
3. Based on your experience, next year when we teach the concepts taught in this tutorial should we only use the educational software, only use traditional tutorial methods (such as lecturing), or use both? (circle one)

Quizzes

A quiz was developed for each of the six weeks of experiment sessions, with the same quiz being administered three times during each experiment session with the goal of measuring learning as it took place over the session. The quiz questions were short answer form and meant to be easy to quickly complete if the participant understood the relevant concept.
Quizzes were marked in a pass-fail manner such that only quizzes completed 100% correctly were considered a pass. The following types of questions were asked on each quiz:

- The binary search quiz required students to describe the algorithm.
- The binary number quiz required students to perform binary number conversions.
- The CPU quiz required students to trace the execution of assembly instructions given a table for register values.
- The polynomial quiz required students to draw an expression graph for a given polynomial.
- The quicksort quiz required students to describe the algorithm.
- The Dijkstra quiz required students to determine what order Dijkstra’s algorithm visited nodes in a given weighted graph.

**Traditional instruction**

Traditional instruction involved Kevin Browne giving the students a presentation which featured an explanation of the concept(s) followed by examples to illustrate them. In all cases the presentation was followed by a Q&A session with the students to help ensure they understood the concepts.

**RESULTS AND DISCUSSION**

In total 101 CS1MA3 students registered to participate in the study and 6 students did not. The number of participants that actually attended to participate in each week of the experiment sessions varied. The participants were made up of 71 males, 29 females and 1 participant who did not report their gender; 88 were right-handed, 8 were left-handed, 4 were ambidextrous and 1 participant did not report their handedness. The compiled results showed no real significant differences between participants of different genders or handedness, so we omit discussion of these differences from this analysis but still provide this data and other pre-experiment questionnaire data for context.

The average years of study completed at the college or university level was 0.73 ($\sigma = 1.19$) and the average year of study in the participant’s current program was 1.53 ($\sigma = 1.01$). There were 33 participants registered in computer science, 22 in mathematics, 8 in physical sciences, 8 in life sciences, 4 in social sciences, with the remainder in other subjects including humanities, economics, and multimedia. There were 32 students that reported having taken a college or university level mathematics course before, and 65 participants reported they did not. The average participant rating of their performance overall in any past mathematics course was 7.70 ($\sigma = 1.49$), and in any computer science courses was 7.19 ($\sigma = 2.03$).

The data regarding participant expertise with various media and interfaces which was collected in the pre-experiment questionnaire is presented in Table 2.

<table>
<thead>
<tr>
<th>Interface/Media</th>
<th>Avg.</th>
<th>SD</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mobile phones</td>
<td>3.93</td>
<td>1.05</td>
</tr>
<tr>
<td>Mobile video games</td>
<td>3.16</td>
<td>1.12</td>
</tr>
<tr>
<td>iPad / other tablets - general usage</td>
<td>2.95</td>
<td>1.20</td>
</tr>
<tr>
<td>iPad / other tablets - video games</td>
<td>2.84</td>
<td>1.19</td>
</tr>
<tr>
<td>Touch screen interfaces</td>
<td>3.65</td>
<td>1.08</td>
</tr>
</tbody>
</table>

Table 2. Participant expertise data - average and standard deviation

**General results**

The binary search app was the first experiment conducted, and many participants vocalized frustration with our post-experiment questionnaire that initially only contained a question asking them whether they preferred traditional instruction or instruction with the apps. Many participants were adamant that using both forms of instruction was preferable.

As a result we added a third question to the post-experiment questionnaire asking participants whether in the future these concepts should be taught with only the traditional methods, only the apps, or both. One of the strongest results across the experiments performed with each app was that a strong majority of participants recommended using both methods to teach the relevant concepts in the future, as seen in Table 3.

<table>
<thead>
<tr>
<th>Experiment</th>
<th>Preferred</th>
<th>Recommended</th>
<th>App</th>
<th>Lesson</th>
<th>App</th>
<th>Lesson</th>
<th>Both</th>
</tr>
</thead>
<tbody>
<tr>
<td>Binary Search</td>
<td>81% (56)</td>
<td>19% (13)</td>
<td>N/A</td>
<td>N/A</td>
<td>N/A</td>
<td>N/A</td>
<td></td>
</tr>
<tr>
<td>Binary Numbers</td>
<td>68% (42)</td>
<td>32% (20)</td>
<td>3% (2)</td>
<td>11% (7)</td>
<td>85% (52)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CPU</td>
<td>63% (38)</td>
<td>37% (22)</td>
<td>7% (4)</td>
<td>13% (8)</td>
<td>80% (48)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Polynomial</td>
<td>69% (41)</td>
<td>31% (18)</td>
<td>5% (3)</td>
<td>12% (7)</td>
<td>83% (48)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Quicksort</td>
<td>51% (25)</td>
<td>49% (24)</td>
<td>2% (1)</td>
<td>30% (14)</td>
<td>68% (32)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Dijkstra</td>
<td>69% (34)</td>
<td>31% (15)</td>
<td>4% (2)</td>
<td>14% (7)</td>
<td>82% (40)</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 3. Participant preferences and recommendations

Using z-score confidence intervals for these sample proportions, we can say with 99% confidence that the majority of the population prefers instruction with the binary search, binary numbers, polynomial and Dijkstra apps to traditional instruction, and with 95% confidence in the case of the CPU app. Again using z-score confidence intervals we can say with 99% confidence across all the apps that the majority of the population recommends a combination of the app and traditional instruction be used in the future (except for the binary search app where we did not collect this data).

The usability survey scores obtained using the formula described previously are shown in in Figure 7. Interestingly, the usability survey scores follow a very similar pattern to the percentage of participants that preferred instruction with each app; the correlation coefficient between the two is 0.86.
The quiz results presented in Table 4 show the results for the sessions that started with a lesson, the sessions that started with the apps, and the combined results across all sessions. The results are consistent with the majority of participants recommending future instruction using both apps and traditional instruction, as by the completion of quiz 3 participants had experienced both instructional methods and quiz 3 averages were generally higher than those of quiz 2 (which was conducted after participants had experienced only one instructional method).

<table>
<thead>
<tr>
<th>Experiment</th>
<th>Order</th>
<th>Quiz 1</th>
<th>Quiz 2</th>
<th>Quiz 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Binary Search</td>
<td>App-first sessions</td>
<td>25.00</td>
<td>55.56</td>
<td>55.56</td>
</tr>
<tr>
<td></td>
<td>Lesson-first sessions</td>
<td>44.12</td>
<td>73.53</td>
<td>82.35</td>
</tr>
<tr>
<td></td>
<td>All sessions</td>
<td>34.29</td>
<td>64.29</td>
<td>68.57</td>
</tr>
<tr>
<td>Binary Numbers</td>
<td>App-first sessions</td>
<td>56.25</td>
<td>75.00</td>
<td>75.00</td>
</tr>
<tr>
<td></td>
<td>Lesson-first sessions</td>
<td>36.67</td>
<td>66.67</td>
<td>70.00</td>
</tr>
<tr>
<td></td>
<td>All sessions</td>
<td>46.77</td>
<td>70.97</td>
<td>72.58</td>
</tr>
<tr>
<td>CPU</td>
<td>App-first sessions</td>
<td>3.45</td>
<td>54.84</td>
<td>77.42</td>
</tr>
<tr>
<td></td>
<td>Lesson-first sessions</td>
<td>0.00</td>
<td>54.84</td>
<td>77.42</td>
</tr>
<tr>
<td></td>
<td>All sessions</td>
<td>1.67</td>
<td>41.67</td>
<td>70.00</td>
</tr>
<tr>
<td>Polynomial</td>
<td>App-first sessions</td>
<td>3.57</td>
<td>39.29</td>
<td>71.43</td>
</tr>
<tr>
<td></td>
<td>Lesson-first sessions</td>
<td>0.00</td>
<td>31.43</td>
<td>51.43</td>
</tr>
<tr>
<td></td>
<td>All sessions</td>
<td>1.59</td>
<td>34.92</td>
<td>60.32</td>
</tr>
<tr>
<td>Quicksort</td>
<td>App-first sessions</td>
<td>0.00</td>
<td>19.35</td>
<td>53.33</td>
</tr>
<tr>
<td></td>
<td>Lesson-first sessions</td>
<td>5.00</td>
<td>70.00</td>
<td>70.00</td>
</tr>
<tr>
<td></td>
<td>All sessions</td>
<td>1.96</td>
<td>39.22</td>
<td>60.00</td>
</tr>
<tr>
<td>Dijkstra</td>
<td>App-first sessions</td>
<td>8.00</td>
<td>56.00</td>
<td>80.00</td>
</tr>
<tr>
<td></td>
<td>Lesson-first sessions</td>
<td>8.33</td>
<td>70.83</td>
<td>79.17</td>
</tr>
<tr>
<td></td>
<td>All sessions</td>
<td>8.16</td>
<td>63.27</td>
<td>79.59</td>
</tr>
</tbody>
</table>

Table 4. Quiz results (average)

Participants gave ample feedback both orally and through the post-experiment questionnaire as to what they perceived as the relative merits of each type of instruction, as summarized in Table 5. There was practically unanimous consensus on the idea that while the traditional lesson-based method was better for ‘explaining’ or ‘teaching’ the concepts, the apps were better for ‘practicing’ or ‘re-enforcing’ concepts. Most participants who expressed this also suggested that the apps be used after the lecture.

<table>
<thead>
<tr>
<th>Lesson Strengths</th>
<th>App Strengths</th>
</tr>
</thead>
<tbody>
<tr>
<td>More comprehensive</td>
<td>More fun</td>
</tr>
<tr>
<td>Learner can ask questions</td>
<td>Learner can try the app as many times as needed</td>
</tr>
<tr>
<td>Better for explaining concepts</td>
<td>Better for practicing concepts</td>
</tr>
<tr>
<td>Better for those who learn through listening</td>
<td>Better for those who learn through doing</td>
</tr>
</tbody>
</table>

Table 5. Participant perceptions of relative instructional method strength

Binary search experiment
There were 70 participants in the binary search sessions, with 35 in experiment sessions that started off with traditional instruction and 35 that started off with instruction with the app.

Many participants expressed frustration at the end part of the app where a binary search tree is displayed, as they were unfamiliar with binary search trees or the concept of trees in general at the time they used this app.

The most common positive feedback about the app was that it was ‘easy to use’, ‘fun’ and ‘interactive’. That the app guided participants to understanding binary search by correcting incorrect sequences of taps was praised by participants who appreciated discovering the concept through trial and error.

Binary numbers experiment
There were 62 participants total in the binary number sessions, with 29 in sessions that started off with traditional instruction and 33 that started off with instruction with the app.

A few users expressed frustration regarding a lack of instructions and ‘not knowing what to do’. There was some negative feedback regarding the time required to make the multiplayer game work; this was most likely due to difficulty in connecting to the network and finding a partner to play with.

In contrast to the complaints about ‘not knowing what to do’, 15 participants left positive comment feedback as to how ‘easy to use’ the app was to use. Similarly the multiplayer game subsection received more positive feedback than negative and was described as being fun and engaging.

CPU experiment
There were 60 participants total in the CPU sessions, with 31 in sessions that started off with traditional instruction and 29 that started off with instruction with the app.

There was some negative feedback about the app being confusing or hard to use, though some of this was qualified by notes that it was only ‘at first’. There was also some annoyance over the beeping sounds that the app would make as participants performed instructions.
The help feature of the app received much praise from participants for preventing them from becoming stuck at any point.

**Polynomial experiment**

There were 59 participants total in the polynomial sessions, with 26 in sessions that started off with traditional instruction and 33 that started off with instruction with the app.

The most common negative feedback was the lack of instructions or indication as to how the app worked.

Again any negative feedback as to the difficulty of using the app was largely outnumbered by the 14 positive comments stating that the app was ‘easy to use’. Interestingly, despite not containing many explicit game elements, several users still described it as fun or game-like.

**Quicksort experiment**

There were 49 participants total in the quicksort sessions, with 20 in sessions that started off with traditional instruction and 29 that started off with instruction with the app.

The quicksort app was the most poorly received app relative to the others due to several issues. One particular point of frustration was the usage of the red X to represent empty sublists, as the participants found this discouraging.

There was also feedback that the app didn’t do enough in terms of animations to explain the concepts, or involve the participant enough in terms of giving them something to do. Other negative feedback suggested the app was ‘boring’ and cited the lack of a game element.

Positive feedback about the app included praise for its simplicity and that it was easy to use.

**Dijkstra experiment**

There were 49 participants total in the Dijkstra sessions, with 25 in sessions that started off with traditional instruction and 24 that started off with instruction with the app.

Similar to the polynomial app the most common frustration about the Dijkstra app was the lack of instructions or explanations about how to use the app and what the app was doing.

However, the graphical explanation of the algorithm was widely praised. Others praised the app for its simplicity, the ability to step through the algorithm, and the ability to setup the nodes and paths.

**CONCLUSION**

We conclude that integrating educational software incorporating game design elements into the instruction of introductory computer science does increase student satisfaction and engagement, on the basis of the participants’ preference for instruction with the apps and their recommendation that the apps be used as part of future instruction.

Based on the oral and written feedback given to us by the participants and our observations during experiment sessions we can make several recommendations about the effectiveness of specific game design elements. While we recommend that apps contain rewards (e.g. treasure chest, check mark) we do not recommend that apps contain penalties (e.g. disapproving sounds, red X) or even symbols and sounds not meant as penalties but that could be perceived as discouraging. The rewards resulted in positive feedback whereas penalties resulted in frustration. This leads to our second point, rather than penalizing users for incorrect actions as is perhaps more typical in video games made exclusively for entertainment purposes, we recommend that educational software provide in-game assistance in the form of corrective help or feedback. The in-game feedback of the binary search and CPU apps received exclusively positive feedback from participants.

Based on our observations and feedback received from the participants, we suggest that software should augment and enhance, but not replace, traditional instruction. We suggest that software should be used primarily for practicing understanding of concepts, and that it should incorporate game design elements to provide increased user satisfaction.

We believe the results are inconclusive as to whether using the apps resulted in a better understanding of the concepts. For ethical reasons, we did not design our experiments with a control group receiving only traditional instruction—an approach which would have allowed us to compare final exam results. However, based on the overwhelming recommendation to incorporate the apps with traditional instruction in the future, we feel that this was the right decision.

**REFERENCES**


