GUI-Based vs. Text-Based Assignments in CS1

Robert Ball, Linda DuHadway, Spencer Hilton, and Brian Rague
Weber State University
1465 Edvalson St.
Ogden Utah 84408
robertball@weber.edu, lindaduhadway@weber.edu, spencerhilton@weber.edu, brague@weber.edu

ABSTRACT
Teaching CS1 can be daunting. The first courses in the CS curriculum help determine which students will ultimately matriculate into the program. There have been various studies on how to improve motivation and reduce attrition by using visual-based environments and assignments. We performed a year-long study in which we addressed two research questions: 1) How is student performance affected by drag-and-drop GUI assignments when compared to traditional text-based assignments? 2) If given the choice, would students select GUI-based or text-based assignments? For the first question, there was no statistical significance, indicating that student performance is not affected by this visual component. For the second question, we discovered more students selected the text-based assignments over the GUI-assignments. Separating the students into groups based on what they chose revealed that the students that selected the GUI-assignments scored on average one letter grade higher, enjoyed the assignments more and spent less time on the assignments. We recorded the reported motivations behind why students chose to do the GUI-based assignments versus the text-based assignments: Overall, the GUI Group's responses trended toward self-improvement (e.g. more like the real world, improve skills, more challenging) while the Text Group's responses trended toward ease (e.g. easier/simpler, save time). Lastly, at the end of each course we asked the students if, given the hypothetical case in which they were not pressed for time, they would create the Java application with or without a GUI? 93% of the students responded that they would create a GUI Java application.

CCS Concepts
• **Social and professional topics → Computer science education; Computational science and engineering education;**

Keywords
CS1; motivation; graphical user interface; assessment; graphics; Java

SIGCSE '18, February 21–24, 2018, Baltimore , MD, USA
© 2018 Association for Computing Machinery.
ACM ISBN 978-1-4503-5103-4/18/02…$15.00
https://doi.org/10.1145/3159450.3159463

1. INTRODUCTION
Instructing CS1 in an engaging manner is especially critical since students make important decisions about their future academic careers based on experiences in introductory courses like CS1. When looking to improve graduation rates for a program, the first place one might look is individual student performance in those first introductory courses.

As a result, there have been many papers published that focus on both increasing enrollment in early courses and decreasing the amount of attrition impacting the program. One common theme in the literature is to make the introductory courses more interesting and exciting. The particular method that we focus on in this paper is using a more visual-based approach, specifically including GUIs (Graphical User Interfaces) in assignments and using a visual drag-and-drop environment to create those GUIs.

We present the results of a year-long study comprising five different sections of an introductory CS1 programming course using Java and comparing GUI-based assignments versus text-based assignments

For the study, we addressed two research questions:

1. How is student performance affected by drag-and-drop GUI assignments when compared to traditional text-based assignments?
2. If given the choice, would students select GUI-based or text-based assignments?

For the first research question, we hypothesized that using a GUI would positively affect grades. Because our investigation of related work supports the position that using a GUI increases the level of student engagement when compared to traditional command line assignments, we conjectured it would have a positive effect on grades.

For the second research question, we hypothesized that students would choose to do the GUI-based assignments over text-based assignments. We speculated that students would prefer to do GUI-based assignments because of the more prominent visual component.

2. RELATED WORK
There have been many attempts to improve on basic textual programming in introductory programming courses. Researchers have presented a myriad of approaches to try to further engage students in introductory courses using a visual approach.

For example, there are a number of visual environments that have been created. BlueJ is a visual programming environment for the Java programming language that supports the concepts of Object-Oriented Programming [10]. RAPTOR is another example that also focuses on visual representations of objects [2].

Similarly, JPie is a visual representation of class definitions that supports direct manipulation of graphical representations of programming abstractions and constructs. The intent is to help new programmers better understand and have less frustration with Object-Oriented Programming in Java through a more visual interface [6].

In addition to the programming environment used, a number of graphical packages or frameworks have been created to garner greater student interest in programming. We review only a few to provide a general idea of what has been developed.

For example, to improve student learning, Roberts, Picard, and Fredricsson explain the use of graphics to introduce object-oriented techniques in a CS1 course. The visual nature of the assignments tends to be more interesting to students than just text-based assignments [15].

In trying to motivate students and increase enrollment, instructors have often turned to using games in CS courses. Whether the games are traditional board games (e.g. [4]) or graphics-based games, the goal is generally the same. Rajaravivarma explains that using games in CS1 helps students with a sense of ownership and greater passion [13].

Leutenegger and Edginton explain a similar phenomenon of increased student interest and understanding by having game-based programming assignments. However, they took their study one step further and showed that, by making three consecutive courses game-based, they had even greater levels of student satisfaction [11].

Given the overall positive results of game-based class activities, why aren't more instructors using games in their courses, despite the potential to further engage students? Some reasons are that many faculty do not have the graphics background necessary to create and design such assignments, games can potentially have alienating effects in regards to gender, and there are few textbook options for pursuing such a course of action [16].

However, games are not the only option for incorporating graphics into the classroom. Holliday and Luginbuhl explain their use of visual memory diagrams, which are visual representations of memory changes as the program executes. They found a correlation between students' ability to construct visual memory diagrams and students' comprehension of object-oriented concepts [7].

As an alternative to simply adding graphical assignments to Java, there are a number of graphics-based programming languages. In general, these languages are used when instructing younger students and often involve storytelling.

Cooper, Dann, and Pausch review different approaches of teaching Object-Oriented Programming. They particularly find Alice, a 3D programming language, to be useful in reducing the attrition of at-risk students [3]. Similarly, Kelleher, Pausch, and Kiesler find that Alice helps motivate middle school girls to learn computer programming [8]. Scratch and Greenfoot are also visual programming languages aimed at motivating younger programmers. Utting, et al. discuss the differences between Scratch, Greenfoot, and Alice [17].

Although games and stories make good use of graphics, they do have limitations. Not all students that graduate will go on to create games professionally. On the other hand, one of the most practical applications of graphics in modern life, even beyond games, is Graphical User Interfaces (GUIs).

Koffman and Wolz make a case for using GUIs in CS1. They explain the advantages and disadvantages of using the traditional text-based approach versus using a GUI. They explain new packages that can be used to encourage student learning, but they stop short of actually evaluating the usefulness of their approach [9].

Proulx, et al. follow a similar vein. They explain that since GUI controls are in fact objects that the object-first methodology should use GUIs rather than text [12].

Alphonce and Ventura created a small graphics package as part of Java's Swing framework. They use the approach of utilizing both GUIs and general graphics to capture greater interest for students. They point out, as do others that advocate GUIs, that students feel comfortable using GUIs and graphics because that is the world that they live in [1].

Following the approach of using GUIs first, English describes JEWL, an automated assessment of GUI-based programs. If instructors are going to be using GUIs in the classes, then it is reasonable to create an automated grading system for the GUI assignments [5].

Reges reports on an experiment where he creates the outline or skeleton of a GUI program from which his students complete the assignment. He reports that students seem to enjoy the programs more and that the class discussions have been noticeably more fun. However, he indicates that it takes more time for him to create the assignments and that students get confused over how to incorporate their code in with his. He concludes that more study is needed to understand how to use his method beyond his personal classroom experiences [14].

It is clear from the literature cited that there are many approaches to creating greater interest in students using visual approaches. Whether instructors use graphical languages (e.g. Alice and Scratch), games, graphical games, or the use of GUIs, visual approaches promote greater student interest.

The purpose of this paper is not to confirm once again that a visual-based approach in CS1 is valuable, but instead to measure this value. We systematically created an experimental environment where we could directly measure how including a visual component – GUIs in this case – in CS1 assignments either helped or did not help students in both performance and motivation.

Many authors (e.g. [1,12,16]) add a caution when introducing visual components to introductory courses, because the visual component-whether graphics, games, or GUIs, - have the potential to add an overhead of additional student learning that detracts from the main fundamental topics being taught. As a result, we elected to use Scene Builder, a drag-and-drop GUI creator, to decrease the overhead of learning additional non-fundamental concepts in CS1.

3. GUI ENVIRONMENT

In this study, the students used Java version 8. For the GUI portion of the course the students used JavaFX. JavaFX was released in December 2008 and became Oracle's GUI replacement to Swing.

Similar to Swing, JavaFX GUIs can be produced by writing Java code, but they can also be produced in a drag-and-drop program called Scene Builder and then tied to a Java program. Scene Builder was produced to make it possible to drag and drop GUI controls from a palette. (Scene Builder is similar to Visual Studio's drag and drop GUI creator.) After the student has created the visual look and feel of the GUI, Scene Builder produces an FXML file – an XML file that Java 8 can load to produce the GUI designed in Scene Builder. Figure 1 shows an example screenshot of Scene Builder.

For example, a student might create a GUI in Scene Builder by dragging menus, buttons, text fields, and other controls onto the form. The student can then alter the properties of any of the controls. For instance, they could change the font, the alignment of controls, or the layout. After the look and feel of the GUI is designed, the student can then create a Java program to run the GUI.

If the Java program with the FXML file were to run at this point, only the GUI that was designed in Scene Builder would appear. The user can interact with controls by pressing a button for example, but it's important to note that no Java code has been written that reacts to the button's click event.

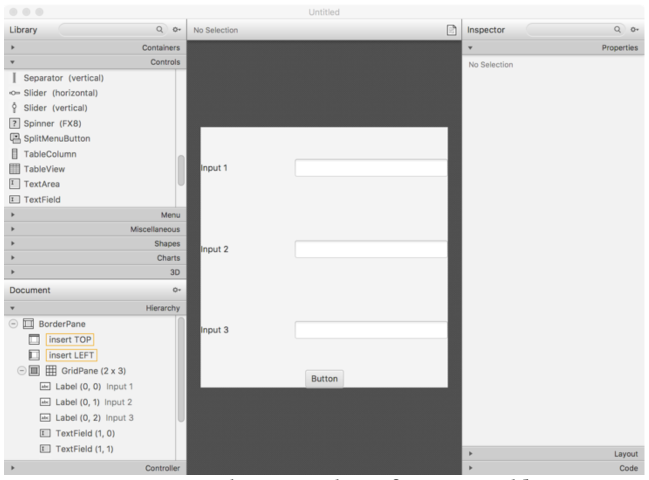

Figure 1. Example screenshot of Scene Builder.

For GUI-based application development, students create a GUI in Scene Builder and save the resulting FXML file. In a separate step, the students create methods in Java to respond to events initiated by the user, which can include receiving input from GUI controls (e.g. text fields) and sending output results to GUI controls (e.g. labels, text areas).

Comparatively, for text-based application development, students interacted with the user by employing System.out.println() to print output and used the Scanner class to acquire input.

To simplify GUI application development, students were given a small Java template that already contained the required JavaFX imports and the code necessary to run the GUI from the FXML file. Not including the imports, the Java template code length was nine lines.

For both text-based and GUI-based applications, the IDE used for creating the Java code for the course was JGrasp. Students were allowed to choose a different IDE if they preferred.

4. EXPERIMENTAL SETUP

The experiment was conducted over a 12-month period. The experiment consisted of two different CS1 sections for Summer 2016, one section for Fall 2016, and two sections for Spring 2017. A face-to-face section was taught every semester and an online section was taught in Summer 2016 and Spring 2017 (see Table 1).

Table 1 CS1 sections offered in each of the three semesters.

	Online	Face-to-face
Summer 2016	17 students	9 students
Fall 2016	***not offered***	23 students
Spring 2017	29 students	26 students

Although it would have been preferable to also have a Fall online section, there was a need for the instructor to teach other CS classes during that time. Also, unlike some universities, Summer classes are allotted the same amount of time as the Fall and Spring semesters. In our university, an individual section runs for about 14 weeks and has an average of approximately 21 students per section.

The following summarizes important information about the sections:

- All five (5) sections of the course were taught by the same instructor with the same textbook and resources.
- All five (5) sections had identical assignments and exams. The only difference between them is that two typos were found in Fall 2016 in the assignments and were subsequently corrected.
- There was a total of one-hundred four (104) unique students that took the course over all five (5) sections.
- For each section of the course, the class was divided into two groups: Group A and Group B. The groups were of equal size and randomly generated. Students were not allowed to self-select. The groups would alternate between text-based assignments and GUI-based assignments for assignments 6-9 (see Table 2).

4.1 Assignments

Table 2 shows a summary of GUI-based and text-based assignments.

The first four assignments were all traditional text-based assignments: the only input and output for the programs was text-based.

The fifth assignment introduced the students to JavaFX and Scene Builder.

Assignments 6-9 had both a GUI-based version and a text-based version. Group A and Group B in each section were alternately assigned the GUI version or the text version. This allowed each group to engage in both types of assignments. Assignments 6-9 were created to test the first research question – how student performance is affected by GUI- or text-based assignments.

Assignments 10 and 13 were GUI-based assignments.

Assignments 11 and 12 could be done either as text-based or GUI-based assignments. In this case, the student chose which version of the assignment they would do. Assignments 11 and 12 were created to test the second research question – what type of assignment would students choose for themselves?

Table 2. The distribution of GUI-based and text-based semester assignments.

Assignments	GUI/text assignment
1	text
2	text
3	text
4	text
5	GUI (introduction to GUIs)
6	Group A: text / Group B: GUI
7	Group A: GUI / Group B: text
8	Group A: text / Group B: GUI
9	Group A: GUI / Group B: text
10	GUI
11	student's choice
12	student's choice
13	GUI

The instructor made great effort each semester to not influence the students in their choice of GUI or text for assignments 11 and 12. The instructor simply stated that there was a choice and did not elaborate further.

For every assignment in each section, we asked the students to report on a scale of 1-4 how much they enjoyed the assignment with 1 being they loved the assignment and 4 being they hated the assignment. In addition, we asked the student to report how long it took them to accomplish the assignment in hours and why they did or did not like the assignment. The exact text that instructed the students on submitting these survey responses follows:

"In **comments section** please answer the following:

- Approximately how long did you spend on this assignment in hours? For example, "1.5 hours" or "3 hours."
- On a scale of 1 to 4, with 1 being the best, and 4 being the worst how much did you enjoy this assignment? For example, "1 - I loved it" or "4 - I hated it."
- Based on your answer of 1, 2, 3, or 4, **why** did you like/dislike the assignment?

The above answers will not affect your grade. By voluntarily answering the above questions you will help us improve this course. Thank you so much for answering the questions!"

4.2 Isometric Assignments

To simplify the process for our CS1 students, most assignments required only one method implementation per assignment. For text-based applications, this meant all code was generally in the main method. For the GUI-based assignments this meant students wrote code in a single method that was called by the primary control, usually a button.

The GUI assignments involved two steps: designing the GUI and creating the Java code. The text-based assignment had one step: writing the main method code.

Consequently, creating a GUI assignment involved a small amount of additional work. In an attempt to offset the additional work required to create a GUI program, the GUI assignments generally had one less requirement than the text-based assignments.

For example, in assignment 8, the text-based assignment asked the user for two numbers: a minimum and a maximum. By comparison, the GUI-based assignment used a slider to get only a minimum, omitting the maximum. The remainder of the requirements for the assignment were identical.

5. RESULTS

Before we investigated our primary research questions, we analyzed the data from the study to determine the existence of any significant relative measures between the groups that might confound our results.

First, we analyzed the difference in all grades between Group A and Group B. (A total of 52 students from the 5 sections were in Group A and 52 were in Group B). Running an ANOVA on all the grades for all assignments of the two groups resulted in no statistical significance.

Second, we analyzed the difference in all grades for online versus face-to-face students regardless of group assignment. Again, there was no statistical significance. There were 46 online students and 58 face-to-face students.

Third, we analyzed the difference in all grades for the students based on when they took the class (i.e. fall, summer, or spring). There was no statistical significance.

5.1 Research Question 1

Our first research question was: ***How is student performance affected by drag-and-drop GUI assignments when compared to traditional text-based assignments?***

Is there a difference in scores between the text-based versus GUI-based assignments? For example, Group A had text-only assignments for Assignments 6 and 8, but GUI-only assignments for Assignments 7 and 9 and Group B had the opposite. How did the assignment scores compare against each other?

First, we ran an ANOVA on the collective grades for the text assignments versus the collective grades for the GUI assignments. The ANOVA showed no statistical significance between the assignments. In other words, regardless of whether the assignment was text-based or GUI-based, one group did not score significantly differently than the other on the same assignment.

Was there a difference between the assignments regarding student enjoyment and time required to complete the assignment? Running an ANOVA on time spent on assignments and how much they enjoyed the assignments resulted in no statistical significance for either analysis. Once again, regardless of whether the assignment was text-based or GUI-based, the groups did not report significantly different amounts of enjoyment nor time spent for the different assignments.

This ran contrary to our hypothesis that students would score better on the GUI assignments. We had presumed that students would enjoy the GUI assignments more and possibly spend more time on the assignments and thus receive higher scores.

This finding is important because one of the major motivations for pursuing this study is that other literature found that students reported in surveys at the end of courses that they generally enjoyed doing visual-based assignments.

However, this raises the question: From the related literature, were the students responding that they liked the visual-based assignments more or that they simply liked the course, the instructor's enthusiasm for the approach, or something else?

What we found in carefully studying the effects of our experiment for assignments 6-9 is that, in essence, when comparing GUI-based with text-based assignments there was no significant difference between (a) assignment grades (b) student enjoyment, or (c) the hours dedicated to the assignments.

5.2 Research Question 2

Our second research question was: ***If given the choice, would students select GUI-based or text -based assignments?***

Students were given a choice to implement assignments 11 and 12 as either a text-based or GUI-based assignment.

Running a t-test found a significant difference: more students chose to implement text-based over GUI-based assignments; [$t(130)=6.214$, $p<0.01$]. 23% of the students chose the GUI-based assignment and 77% of the students chose the text-based assignment (see Table 3).

This was a particularly interesting result because it ran contrary to our hypothesis and the related literature. Given the choice to do a GUI or text-based assignment three-quarters of the students chose to do text-based assignments.

We found that there was no statistical significance in scores between the self-selected GUI and text groups for assignments 11 and 12. There was also no statistical significance in hours spent on assignments 11 and 12.

Table 3. Comparison of the self-selected groups: GUI Group vs. Text Group for Assignments 11 and 12. Only students that completed assignments 11 and 12 were included in the analysis. This removed most of the failing students and students that had earlier dropped the class from the analysis.

	GUI Group	Text Group

Percentage of students in course	23%	77%
Number of students	21	53
Enjoyment (1-4) (lower is more enjoyment; higher is less enjoyment)	1.7	2.21

However, running an ANOVA on enjoyment of text-based versus GUI-based assignments there was statistical significance [F(1, 74)=5.472, p=0.22]. For the 23% of the students that chose to do a GUI-based assignment, they had an average of 1.7 enjoyment compared to a 2.21 enjoyment for the students that chose the text-based assignment. In other words, on a 1-4 scale, the students that chose a GUI-based assignment had a 0.51 (relative to a 4-point scale) higher enjoyment measure than their peers that chose a text-based assignment.

5.3 Further Comparison of Self-Selected Groups (GUI Group vs. Text Group)

Looking at the data, we grouped all the students that decided to do a GUI on either Assignment 11 or Assignment 12 in "GUI Group" and all the other students in "Text Group" (see Table 4).

If the student did not complete Assignment 11 nor Assignment 12 they were not included in the analysis. This removed most of the failing students and students that had earlier dropped the class from the analysis. There were only two students that failed the course that also completed Assignments 11 and 12.

We then compared the groups by comparing their final grade in the course to each other. We found a statistically significant result [F(1,72)=4.297, p=0.04] with the average final grade in the GUI Group of 91.2% and the average final grade in the Text Group of 84.7%.

Running an ANOVA on the number of hours reported (per assignment for all assignments) found statistical significance [F(1,72)=3.662, p=0.059] with an average 1.99 hours (119 minutes) per assignment for the GUI group and 2.67 hours (160 minutes) per assignment for the Text group.

Table 4. Comparison of the self-selected groups: GUI Group vs. Text Group for all assignments

	GUI Group	Text Group
Number of students	21	53
Average final grade for course	91.2%	84.7%
Average hours spent on all assignments	1.99 (119 minutes)	2.67 (160 minutes)
Average Enjoyment (1-4) (lower is more enjoyment; higher is less enjoyment)	1.44	1.78

This shows that the GUI group, which received on average a better grade, also on average spent less time per assignment than the Text Group.

Does this show that GUI development actually saves time? No. Looking at our first research question we found that for the controlled experiment using non-self-selected groups the GUI did not take a statistically significant difference in the number of hours reported by the students.

Running an ANOVA on the average amount of enjoyment found a statistical significance [F(1,72)=3.933, p=0.051] with an average of 1.44 enjoyment for the GUI group and an average of 1.78 for the Text group. (Recall that the lower the number, the more the student liked the assignment.)

In summary, (a) the students in the GUI group on average had a higher final grade (an average of a letter grade difference), (b) took less time per assignment, and (c) enjoyed the assignments more compared to their Text group counterparts.

The next section explains what the students self-reported.

5.4 Rationale for student selections – in their own words:

Although it is interesting to see how the different groups compared to each other in terms of score, enjoyment, and number of hours spent, this study can be complemented by an investigation of the reasons students offered when choosing GUI-based or text-based assignments.

Table 5 shows an aggregation of the reasons for the two groups. This was a free-response survey where the students could respond anyway they preferred. We aggregated the results based on similar responses.

Table 5. Aggregated reasons for choosing to do Assignments 11 and 12 as a GUI-based or text-based assignment.

Reasons for Choosing GUI		Reasons for Choosing Text	
GUI's used in real life	23%	Easier/simpler	51%
To improve skills	23%	Save time	18%
Fun	18%	More familiar with text	10%
Prefer GUI over text	14%	Variety – already did GUI before	10%
More challenging	14%	Miscellaneous	6%
Variety - already did text before	5%	Dislike GUI's	4%
Easier/simpler	5%		

Of particular note is that the students who chose to do the GUI chose to do so for vastly different reasons than the students who chose to the do text-based assignments. Overall, the GUI Group's responses trended toward self-improvement (e.g. more like the real world, improve skills, more challenging) while the Text Group's responses trended toward ease (e.g. easier/simpler, save time).

5.5 Post-course question

At the end of the course, during finals week, we sent emails to students asking the following two questions:

"Question1: Based on your experience in this class, if you were not pressed for time, would you create the Java application with or without a GUI?

Question 2: Why?"

46% of the respondents were part of the GUI Group (e.g. they had chosen GUI on assignment 11 and/or 12), but 93% of all the respondents to Question 1 stated they would use GUI outside of class.

In other words, of the students that chose to answer, approximately half of them had chosen to do assignment 11 and/or 12 as a GUI assignment. However, 93% of the respondents answered that if they were not pressed for time that they would create a GUI application over a text application.

Regarding Question 2, 53% of the respondents mentioned that GUIs have better usability, 26% mentioned that GUIs are easier for the user, and 20% mentioned that GUIs are "fancier" or more "visually appealing" than text applications.

The respondents that indicated that they would not choose GUIs (the remaining 7%) indicated that GUIs take too long to create.

6. INDIVIDUALITY COMPONENT

Like many other authors (e.g. [13]), we noticed that, when students are allowed freedom of expression, they often take advantage of it. Perhaps one way to express individuality is to make the assignment your own.

Although text-based programs can also be individualized, the GUI-based assignments had a greater capacity for design customization than the text-based programs.

Student text-based submissions generally differed in the exact text displayed and perhaps in the order of input/output. Structurally, most text-based assignments that received full credit were remarkably similar.

However, when comparing GUI-based assignments that received full credit, differences are easily noticed. Most students added color, added background images, changed the layout of the controls, or provided one of many individual touches to the GUI. As a result, no two GUI-based assignments were ever identical

7. CONCLUSION

In the beginning, we had two research questions. Our first research question was: ***How is student performance affected by drag-and-drop GUI assignments when compared to traditional text-based assignments?***

When comparing GUI-based with text-based assignments, there was no significant difference between (a) assignment grades (b) student enjoyment, or (c) the hours dedicated to the assignments. This ran contrary to our hypothesis that students would score better on the GUI assignments.

This finding is important because other literature found that students reported in surveys at the end of courses that they generally enjoyed doing visual-based assignments. However, this raises the question: Were the students responding that they liked the visual-based assignments or that they simply liked the course, the instructor's enthusiasm for the approach, or something else?

Our second research question was: ***If given the choice, would students select GUI-based or text -based assignments?***

Given the choice to do a GUI or text-based assignment slightly more than three-quarters of the students chose to do text-based assignments. Further, we found that there was no statistical significance in scores or in time spent on those particular assignments, although the students that chose to do the GUI-based assignments did enjoy their assignments more than the students that chose to do the text-based assignments.

Looking more in-depth into the students that chose to do a GUI-based assignment (GUI group) versus the students that chose to do a text-based assignment (text group), we found that students in the GUI group on average (a) had a higher final grade (an average of a letter grade difference), (b) took less time per assignment, and (c) enjoyed the assignments more compared to their Text group counterparts.

We also found the reported motivations of why students chose to do the GUI-based assignments versus the text-based assignments differed. Overall, the GUI Group's responses were more directed toward self-improvement (e.g. more like the real world, improve skills, more challenging) while the Text Group's responses were more directed toward ease (e.g. easier/simpler, save time).

We plan to follow up with the students from this year-long study and track their overall academic performance. In particular, we plan to track how many of the GUI Group versus the Text Group graduate with a CS related degree. In addition, we plan to track how their overall GPA compares, and how many years it takes the students in each of the respective groups to graduate.

8. REFERENCES

[1] Alphonce, C., & Ventura, P. (2003, October). Using graphics to support the teaching of fundamental object-oriented principles in CS1. In ***Companion of SIGPLAN '03*** (pp. 156-161).

[2] Carlisle, M. C. (2009). Raptor: a visual programming environment for teaching object-oriented programming. ***Journal of Computing Sciences in Colleges***, *24*(4), 275-281.

[3] Cooper, S., Dann, W., & Pausch, R. (2003, February). Teaching objects-first in introductory computer science. In ***ACM SIGCSE Bulletin*** (Vol. 35, No. 1, pp. 191-195). ACM.

[4] Drake, P., & Sung, K. (2011, March). Teaching introductory programming with popular board games. In ***Proceedings of the 42nd ACM technical symposium on Computer science education*** (pp. 619-624).

[5] English, J. (2004, June). Automated assessment of GUI programs using JEWL. In ***ACM SIGCSE Bulletin*** (Vol. 36, No. 3, pp. 137-141).

[6] Goldman, Kenneth J. "An interactive environment for beginning Java programmers." ***Science of Computer Programming*** 53.1 (2004): 3-24.

[7] Holliday, M., and David Luginbuhl. "Using memory diagrams when teaching a Java-based CS1." ***Proc. of the 41st Annual ACM Southeast Conference***. 2003.

[8] Kelleher, C., Pausch, R., & Kiesler, S. (2007, April). Storytelling alice motivates middle school girls to learn computer programming. In ***SIGCHI'07*** (pp. 1455-1464).

[9] Koffman, E., & Wolz, U. (2001, February). A simple java package for GUI-like interactivity. In ***ACM SIGCSE Bulletin*** (Vol. 33, No. 1, pp. 11-15).

[10] Kölling, M. & Rosenberg, J., Guidelines for teaching object orientation with Java. In Proceedings of the 6th annual conference on Innovation and Technology in Computer Science Education (Canterbury, England, June, 2001), 33-36.

[11] Leutenegger, S., & Edgington, J. (2007). A games first approach to teaching introductory programming. ***ACM SIGCSE Bulletin***, *39*(1), 115-118.

[12] Proulx, V. K., Raab, J., & Rasala, R. (2002). Objects from the beginning-with GUIs. ***ACM SIGCSE Bulletin***, *34*(3), 65-69.

[13] Rajaravivarma, R. (2005). A games-based approach for teaching the introductory programming course. ***ACM SIGCSE Bulletin***, *37*(4), 98-102.

[14] Reges, S. (2000, May). Conservatively radical Java in CS1. In ***ACM SIGCSE Bulletin*** (Vol. 32, No. 1, pp. 85-89).

[15] Roberts, Eric, and Antoine Picard. "Designing a Java graphics library for CS 1." ***ACM SIGCSE Bulletin***. Vol. 30. No. 3. ACM, 1998.

[16] Sung, K. (2009). Computer games and traditional CS courses. ***Communications of the ACM***, *52*(12), 74-78.

[17] Utting, I., Cooper, S., Kölling, M., Maloney, J., & Resnick, M. (2010). Alice, greenfoot, and scratch--a discussion. ***ACM Transactions on Computing Education (TOCE), 10(4), 17.***