Mobile App Development to Increase Student Engagement and Problem Solving Skills

Sonal Dekhane
Xin Xu
Mai Yin Tsoi
School of Science and Technology
Georgia Gwinnett College
Lawrenceville, GA 30043, USA
sdekhane@ggc.edu, xxu@ggc.edu, mtsoi@ggc.edu

ABSTRACT

This paper describes a project designed to promote problem solving and critical thinking skills in a general education, computing course at an open access institution. A visual programming tool, GameSalad, was used to enable students to create educational apps for mobile platforms. The students worked on a game development project for the entire semester, incorporating various skills learned throughout the semester. Pre and post quiz analysis showed a significant improvement in students' ability to design comprehensive solutions to a given problem. Survey results also showed increased student engagement, high interest in computing and a "better" understanding of information technology.

Keywords: Creative problem solving, Critical thinking, General education, Mobile computing

1. INTRODUCTION

With the advancement of technology, computer hardware and software have become essential tools not only for science and engineering fields, but also for business and liberal arts disciplines. For example, in physiology, computers have been used to assist psychological assessment (Fowler, 1985, p.748); in the business world, computers have made e-commerce the norm; in biology, computer programs have been developed to estimate gene genealogies (Clement, 2000, p.1657). To be successful in their academic studies and in their future career, today's students need to be able to adapt to a dynamic environment surrounded by new technologies. Thus, basic computer literacy is not enough to stay competitive in the current workforce. It has become essential that students develop a deeper understanding about computing and adequately apply computing skills, such as creating and manipulating digital graphics. More importantly, the problem solving skills and critical thinking ability developed and honed through the application of these computing skills are crucial to a student's future success in the face of constantly evolving technology regardless of their major. The traditional programming language courses are usually considered to be effective in fostering these skills. However, learning how to program in a language such as Java or C++ has been proven to be difficult, even for computing majors (Bennedsen & Caspersen, 2007, p.32) (Dodds et al., 2008, p.266). As a result, enrollment and retention rates in computer science (CS) and information technology (IT) programs have suffered (Uludag et al, 2011, p.183) (Computing Research Association, 2011) and students therefore lose out on prime opportunities to develop their problem solving skills and critical thinking ability. Researchers have investigated and discovered that traditional programming courses fail to connect programming and CS concepts with students' diverse interests and backgrounds (Forte & Guzdial, 2005, p.248). The authors of this paper also observed that the strict syntax of traditional programming languages become the primary focus of the course and as a result students are unintentionally discouraged from solving problems and from expressing their creativity. This phenomenon is even more prevalent among students in programming courses that are not majoring in CS. Various researchers have attempted to develop different strategies to improve student performance in introductory programming courses. These strategies include:

a) Addressing the issue from the social aspect by applying pair-programming (Nagappan et al., 2003, p359) (Williams et al., 2000, p.98) (Carver et al., 2007, p.115) (McDowell et. al, 2006, p.136) and collaborative learning (Teague & Roe, 2008, p.147).
b) Increasing students' interest in CS by using themes that are attractive to students. Successful results have been reported by using multimedia approach (Guzdial & Ericson, 2007), game approach (Kölling & Herriksen, 2005, p59), and animation approach (Crawford & Boese, 2006, p.156).

c) Using visualized programming to introduce core concepts before more advanced and in-depth courses are offered (Johnsgard & McDonald 2008, p.129).

In support of this strategy, many visualized programming tools have been developed. Popular ones include:

1. Scratch: This tool provides an environment where users can create animations, games and music by dragging and dropping the pre-defined programming blocks in the right places. The original targeted audience was young users aged 8 to 16 years (Malan & Leitner, 2007, p.223). The use of Scratch (Rizvi et al., 2011, p.19) has enabled the development of a successful CS curriculum and an interdisciplinary course to promote computational thinking (Ruthmann et al., 2010, p.351).
2. Alice: Alice provides 3-dimensional characters, scenes and environments that users can manipulate and alter to create their own interactive animated stories. This creative activity teaches students basic programming concepts without the frustration caused by the steep learning curve of programming syntax. The Storytelling Alice, a version of Alice, is considered appropriate for middle school students (Kelleher et al., 2007, p.1455). Overall, studies have shown that this strategy has been successfully adopted at several college level CS0 courses (Dougherty, 2007, P.145) (Mullins, et al., 2009, p.136) (Wellman et al., 2009, p.98).
3. Lego MindStorms: In this strategy, this tool provides users with a kit that includes both software and hardware. Students can create a robot and control it through a visualized programming interface. The original targeted users were students in grades K-12. However, over the years, it has been successfully adopted to introduce programming to college students as well (Klassner & Aderson, 2003, p.12) (Lawhead et al., 2002, p.191) (Cliburn, 2006, p.1)
4. Kodu: Kodu is a visual programming tool used to create games for the PC and Xbox platforms. It is recommended for students ages 8 and above. The Kodu classroom kit for educators includes lesson plans and activities (Microsoft Research FUSE LABS).

Introducing programming to students using visualized tools is not a novel idea, but developing applications for mobile devices is a relatively new concept. In the past, the tools and environments for mobile app development required a certain amount of software development expertise and were usually considered as options only for professionals. However, in recent years, visualized programming tools for mobile devices have been created. Currently, the dominant ones are GameSalad for Mac operating system and AppInventor for the Android platform. GameSalad has also been able to support the Windows and the Android platforms since 2012. These tools effectively enabled non-professional, general users to create animations, games and other apps without extensive prior programming knowledge. This software is robust as well; it is not only suitable for academic purposes but also useful for professional animators and game developers. While providing a welcomed opportunity for users to express their creativity, these new strategies reduce the required level of prior expertise in programming language and effectively help shift the classroom's focus towards problem solving and critical thinking. Moreover, the popularity of mobile devices and apps amongst students and their ubiquity cannot be ignored; this only enhances the students' interest and engagement in the learning process. Other benefits of developing animations and games for mobile devices using visualized tools include a relatively short learning curve and thus a decrease in frustration for the students and immediate visual feedback to students. These benefits have been found to positively help in engaging students in the CS0 classroom (Wolber, 2011, p.601). As students are increasingly finding success in the development process, they inherently improve their computing skills and therefore will hopefully become more productive as they enter the workplace in this increasingly digital world.

The authors took into consideration the importance of Information Technology (IT) fluency, the role of programming in IT and the difficulty in learning programming (especially for non-majors). These factors drove the decision to address student problem-solving skills in the general education course named Digital Media. In this study, the intervention chosen was the visual programming tool, GameSalad. The goal of this project was to provide an active, engaging and exciting learning environment for students not majoring in CS so they could gain intermediate level computing skills and develop their problem solving skills. The course project was designed to enable students to develop apps and games for the iOS platform on Apple devices, thus leveraging the increasing student interest in mobile devices, such as the iPad, and mobile applications.

In summary, the goals for this project were to improve student problem solving skills and critical thinking so our future college graduates can productively contribute to today's technology-driven workplace. By providing an engaging learning environment via iPad mobile game development using a visual programming tool that bypasses common introductory syntax issues, the authors aimed to increase student interest in computing and to sustain student engagement in the general education IT course Digital Media.

2. BACKGROUND RESEARCH

Many IT/CS degree programs in colleges and universities offer an introductory level IT course for major and non-major students. Traditionally, this introductory course teaches programming using programming languages such as Java, C++ and Visual Basic. Over the years, these introductory courses have been identified as obstacles for student retention in computing majors (Turner et al., 2007, p.24). The focused and linear approach of teaching each student the complexities and vocabulary of the language's syntax does not often promote creativity and inquiry-based learning. Due to the extensive amount of syntax required to create a computer program, students are left to memorize and accept traditional programming concepts without investment or engagement in the topics. As a result, students often do not recognize or appreciate the problem-solving opportunities within the software development process; students then become disengaged and thus tend to move towards other areas of study, causing a depletion of brain power and incoming fresh talent into the CS industry.

In order to recruit and retain students, many efforts have been put into designing an interesting and engaging introductory course. Approaches such as implementing a computer game theme into the course can motivate students to a certain degree, since many students are somewhat familiar with and interact with computer games in their lives (Cliburn & Miller, 2008, p.138) (Leutenegger & Edgington, 2007, p.115). These efforts focus on increasing students' interests in IT/CS and on attracting them into the IT/CS programs. The authors' goal is to promote problem solving and critical thinking skills through this general education course. The assessment and evaluation strategies are therefore different than other similar studies. Other studies usually use indicators such as enrollment improvement or increased retention rate to assess the result. The authors assessed students' problem solving skills in addition to their computing attitudes to evaluate the success of this project.

Students still have to eventually master the language specific content and struggle with the syntax in order to competently develop programs. Using visualized programming is an engaging teaching methodology and does not require prior programming knowledge, but the shortcoming of the traditional visualized programming tools such as Alice and Scratch is that it is hard to directly apply it into the real world as the program usually only runs on a computer. With mobile game development, students can test their product on mobile devices and receive immediate feedback, which relates the learning process to their real life.

With the explosive usage of mobile devices such as smartphones and tablets, these handheld devices have been portrayed in a positive light, even so far as to make them a fashion element to generate enthusiasm from the students. Visualized mobile game tools provide students the opportunity to fully demonstrate their creativity in a structured environment where they are not completely hindered by their lack of programming knowledge. The resulting mobile apps developed by the students relate directly to their lives in that the final product can be directly uploaded and shared with many other mobile device users. All these characteristics make mobile game development appealing for students. (Uludag et al., 2011, P.183) shared their successful experience of using App Inventor to motivate students by creating apps for Android system. (Spertus et al., 2010, p.325) and (Wolber, 2011, p.601) also piloted mobile development using App Inventor and found the experience very rewarding in terms of student reaction and learning outcomes.

In this paper, we present our experience of using GameSalad to develop apps and mobile games for iOS and our experience with testing those apps on the Apple iPad. In the following sections, we will describe the institutional context of the course that incorporated the mobile game development component, the free visualized game development tool GameSalad, the project implementation details and the assessments. We will then share both student and faculty perspectives on this project.

3. RESEARCH METHOD

3.1 Institutional Context

Georgia Gwinnett College (GGC) is a premier 21st century liberal arts college that emphasizes graduating students with strong technical skills. To this end, all GGC students are required to take two information technology courses. The first course in this sequence is an introductory computing course (ITEC 1001) that focuses on both conceptual knowledge and skills development. For the second course, students may choose from either Introduction to Programming (ITEC 2120) or Digital Media (ITEC 2110). Our enrollment statistics indicate that most non-CS major students prefer to take Digital Media over Introduction to Programming. Introduction to Programming is currently taught using the Java programming language and is widely perceived among students as a difficult course. The Digital Media course, on the other hand, focuses on the theoretical concepts and practical aspects of working with various digital media. The prevalence of digital media in the everyday world of today's students makes this course more popular. Both of these courses are intended to improve the IT fluency of our students, to inculcate logical thinking, and to hone their problem solving skills. Digital Media was designed to meet these goals and to clear some of the common student misconceptions about the computing field by leveraging the students' interest in digital and social media. Students worked on the Mac operating system in this class.

At GGC, multi-section course instructors have some flexibility in designing their courses. For the Digital Media course all sections are required to cover theory and practice of image editing, audio editing, video editing and animation. Instructors are free to add topics of their choice to this list. This is where GameSalad fit into our course. No changes were made to the required modules.

3.2 GameSalad

GameSalad is a game development tool that is designed to empower everyone to create games for various platforms without regard to their proficiency in a specific programming language. This tool is freely available online for download and for use on Mac and Windows operating systems. The tool provides easy "drag and drop" features that allow individuals to create the games. Research indicates that a better approach to introducing programming concepts to students is to use such "drag and drop" tools that eliminate the complexity of the syntax of a programming language and resulting compilation errors, thus decreasing initial frustration with the course. The immediate visual feedback of such tools has also shown to be effective in engaging the students in computing. The tools used in this study included GameSalad Creator and GameSalad Viewer. GameSalad Creator is a freely available tool that can be downloaded from www.gamesalad.com to develop and preview the application. GameSalad Viewer is another free tool that can be downloaded and used to test GameSalad applications on mobile devices such as iPhone and iPad. Currently, there are two versions of the Viewer; iOS viewer and Android Viewer.

3.3 Classroom Experience

GameSalad was introduced in the Digital Media course in Fall 2011. During the first phase of the project (Fall 2011, Spring 2012) data about students' computing attitudes was gathered. During the second phase of the project (Fall 2012, Spring 2013) pre and post-tests were used to assess the problem solving skills of the students in addition to collecting data about their computing attitudes. The project implementation itself did not change during the second phase of the project. When integrating GameSalad into our Digital Media course our goals were: 1) to leverage student interest in mobile technology and apps to enhance problem-solving skills and 2) to increase student engagement in a computing course by providing an active learning environment. Also, as part of our college's commitment to undergraduate research experience, we aimed to enable the students to work collaboratively and to use library and online resources for research purposes. The GameSalad component also afforded an excellent opportunity to the students to integrate their image editing and audio-editing skills gained throughout the semester into their game.

During the class, students were given a simple game to play on an iPad device provided to them. This game was created using GameSalad and targeted specific computing concepts, which were the learning outcomes for that day. After playing the game the students were asked to identify the objects, also called as actors in the game. For each actor the students then had to identify the properties and the various behaviors of the actors. Through this initial question-and-answer session, students were led to discover the common questions that developers ask when designing a game. The instructors modeled the development process for students. This process began with problem identification, then moved to solution design and once the problem and design were established, moved to the creation process. Many students were eager to start working with the GameSalad tools without thoroughly thinking through the game's objectives and design. As a result, these students quickly realized that a lot more pre-planning is needed in order to effectively create a working game. The students' experiences reinforced the learning objectives and the process for building effective computing solutions

The instructors helped students to explore and discover various features of Game Salad. With the instructors' guidance the students developed the game that they played at the beginning of the session, thereby learning specific computing concepts such as classes, objects, attributes, methods, conditional statements, variables, etc. By following this process, not only were the students encouraged to make design decisions, but they were also required to test and debug their games. By coupling interactivity, creativity, and inquiry-based learning, instructors aimed to illustrate to the students the process of determining the requisite factors that needed to be addressed for their mobile app to effectively work. This process was the cornerstone for solving Digital Media project issues; by working through their individual investigations, students were able to hone their critical thinking skills. It was imperative that the instructors gave no direct instructions on how to create the mobile apps, besides the basic guidance on how to use GameSalad. Indeed, students were asked to research answers to their own questions or errors during the design and testing processes. Not only did this encourage the Digital Media students to become self-reliant and to collaborate with peers, but this also modeled for the students how the outside world of software development goes about creating mobile apps – in an iterative, problem-solving cycle with lots of changes, mistakes, and testing.

In this Digital Media course, the students also used various image editing, audio editing, video editing and animation applications. As part of their GameSalad project, the students worked in pairs to create an educational application based on a specific concept in their field of study. They first conducted online research to investigate existing educational applications available on the AppStore and Android Market. The students also had an opportunity to test apps of their choice on iPad devices provided by the college. The next step was to research basic game and user interface design principles for mobile platforms. Based on these findings and guidance from the instructor, the teams created a simple game design document that explained the main idea and the game flow of their unique game. With feedback from the instructor, the teams then created their games using GameSalad and tested it on iPads. The teams used GIMP and GarageBand to create images and the soundtrack for their game.

The following figures depict different modes of a simple biology game created by students, named "Cell Bio Reference". This game included the logic to detect touch and drag events, collisions and to keep score. It begins by showing instructions on app usage. The app has 3 different modes. In the Study mode the users are shown labeled diagrams accompanied by a list of terms. The user can tap the terms to go to the Glossary mode for a definition of the term. The Glossary has touch enabled navigation buttons to go back or move on to other modes of the game. In the Practice mode the user is taken through several review questions and answers. The Practice mode also has a game that has several objects floating on the scene. More objects appear on the scene as the game continues. The user's goal is to drag all plant cells into the plant cell box, animal cell objects on to the animal cell box and so on. This is a timed game and the user scores points every time a correct match is made. There are 3 types of cells to study along with 26 glossary terms and 20 review questions.

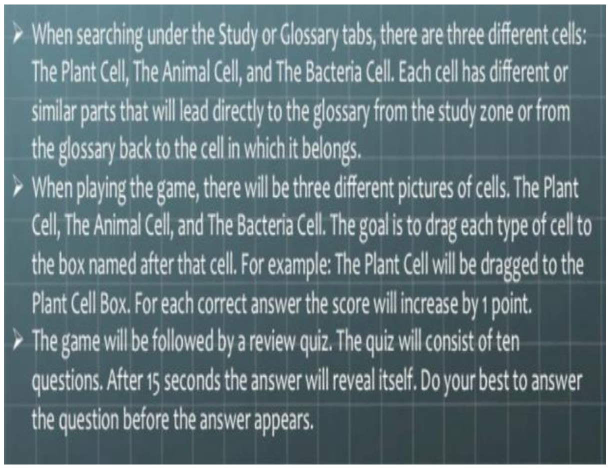

- When searching under the Study or Glossary tabs, there are three different cells: The Plant Cell, The Animal Cell, and The Bacteria Cell. Each cell has different or similar parts that will lead directly to the glossary from the study zone or from the glossary back to the cell in which it belongs.
- When playing the game, there will be three different pictures of cells. The Plant Cell, The Animal Cell, and The Bacteria Cell. The goal is to drag each type of cell to the box named after that cell. For example: The Plant Cell will be dragged to the Plant Cell Box. For each correct answer the score will increase by 1 point.
- The game will be followed by a review quiz. The quiz will consist of ten questions. After 15 seconds the answer will reveal itself. Do your best to answer the question before the answer appears.

Figure 1 App Usage Instructions

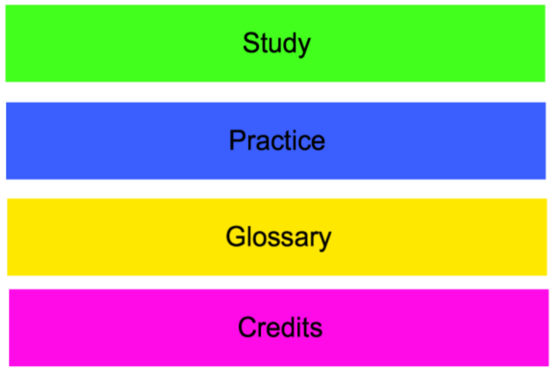


Figure 2 Main Menu, Tap Enabled

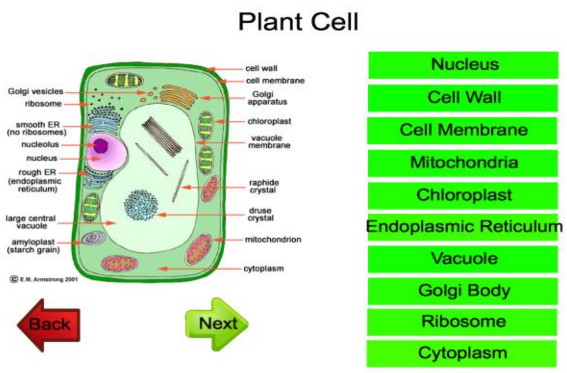
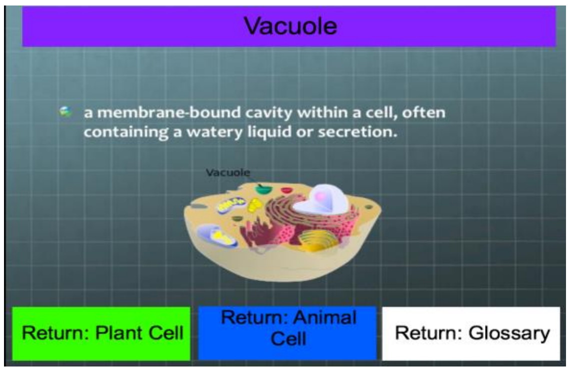
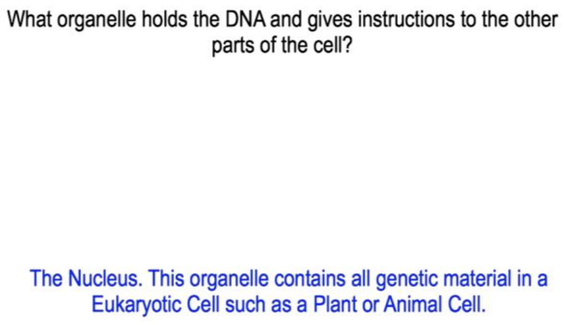


Figure 3 Study Mode Linked to Glossary


Vacuole

a membrane-bound cavity within a cell, often containing a watery liquid or secretion.

Figure 4 Glossary


What organelle holds the DNA and gives instructions to the other parts of the cell?

The Nucleus. This organelle contains all genetic material in a Eukaryotic Cell such as a Plant or Animal Cell.

Figure 5 Review Mode

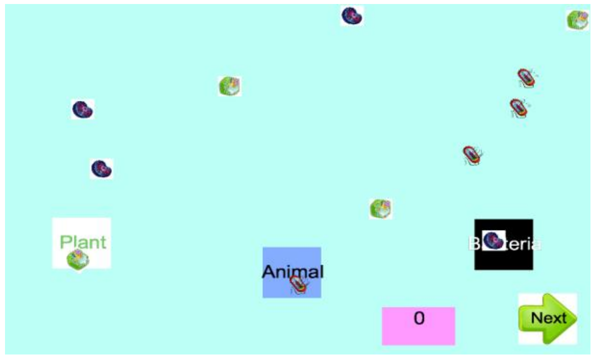


Figure 6 Practice Game

3.4 Data collection

To evaluate the impact of this project, we administered a "Computing Attitudes" survey at the end of each semester. We also administered a pre and post quiz to assess the problem-solving skills of our students. In total, 70 students participated in the survey during the Fall semester of 2011, 65 students in Spring semester of 2012 and 70 students in Fall semester of 2012. In the second phase of the project, we designed problem-solving assessments. The study had 29 students participate in the pre and post quiz in Fall 2012 and 28 students participate in Spring 2013.

In the survey, students were asked about their background (gender, age, college year, major), computer usage habits and previous experience with touchscreen mobile devices (if any). The survey also included questions about the students' attitudes towards computing, which were adapted from existing instruments. Student engagement was assessed using questions based on (Garrison et al., 2008). A sample statement read, "*Compared to other courses the amount of interaction I experienced with my ITEC 2110 instructor has increased*". Four such statements were used to gather data about the perceived amount and quality of student-student and student-faculty interaction. Finally, the survey also gathered student feedback on the mobile game development experience using GameSalad. The students rated the survey statements on a Likert scale of 1-5, 1 being "Strongly Disagree" and 5 being "Strongly Agree" with the statement given.

Pre and post quizzes were designed to assess students' problem-solving skills and were based on the assessment method proposed by (Deek et al., 1999, p.317). For both pre and post quiz the students were asked to play a simple game on their iPad. They then had to answer questions that required them to formulate the problem, design the solution and plan test cases. The games that students used for the pre and post quiz were similar in nature. A detailed list of possible outcomes for each of the phases was developed and used as a scoring rubric.

The semester long project had several intermediate deliverables that provided additional insight into student work. The research reports were used to evaluate the achievement of our secondary goal of using online and library resources for research. The educational apps themselves were used to evaluate solution quality (reliability and correctness), as suggested by (Deek et al., 1999, p.317).

4. RESULTS

4. 1 Quantitative Results

The background information of all the students that participated in the end of semester survey is as shown in figure 7. Gender distribution was almost equal and the majority of the students were aged 17 to 21 years. Only 20% of the students were computing majors; 52% of the students were sophomores.

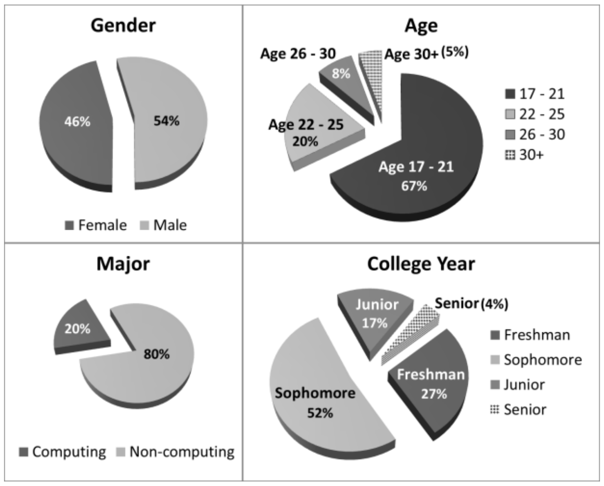


Figure 7 Participants' Demographic Information

Figure 8 shows the computer usage distribution of the participants. The majority of the students had never written any type of computer program or created computer games previously; however, most of them had used an iPhone/iPod/iPad before. When asked about the apps used on mobile devices, the participant responses showed a wide range. Applications ranged from games such as "Angry Birds" to popular social media apps and personal organization apps. There was no mention of educational apps used for studying.

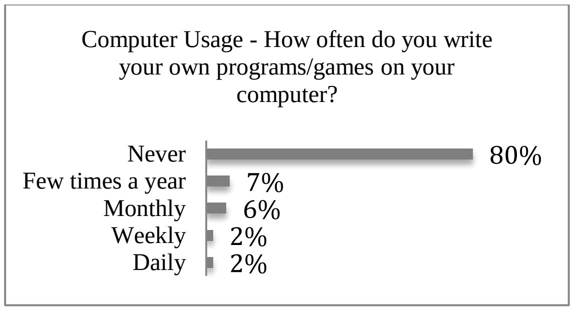


Figure 8 Participants' computer usage information

The chart in figure 9 shows the average values of the student computing attitudes. Student responses in all three semesters overlapped considerably. Students strongly agreed with statements indicating that computers are fun and useful and that computing is a logical and creative activity. The survey participants strongly disagreed with statements indicating that computers are boring or that computer jobs are all about programming. The participants' confidence and willingness to learn new software tools was also high.

As shown in figure 10 although most students agreed that creating mobile games was a fun activity, they also indicated that they perceived the project as difficult. Also, although students responded very positively to wanting to learn more about computing, the interest in computing careers was average.

As shown in figure 11 students also indicated increased amount and quality of interaction with faculty and other students. Interestingly, pivot tables did not show any significant impact of factors such as gender, age, major and prior experience with mobile devices on students' computing attitudes.

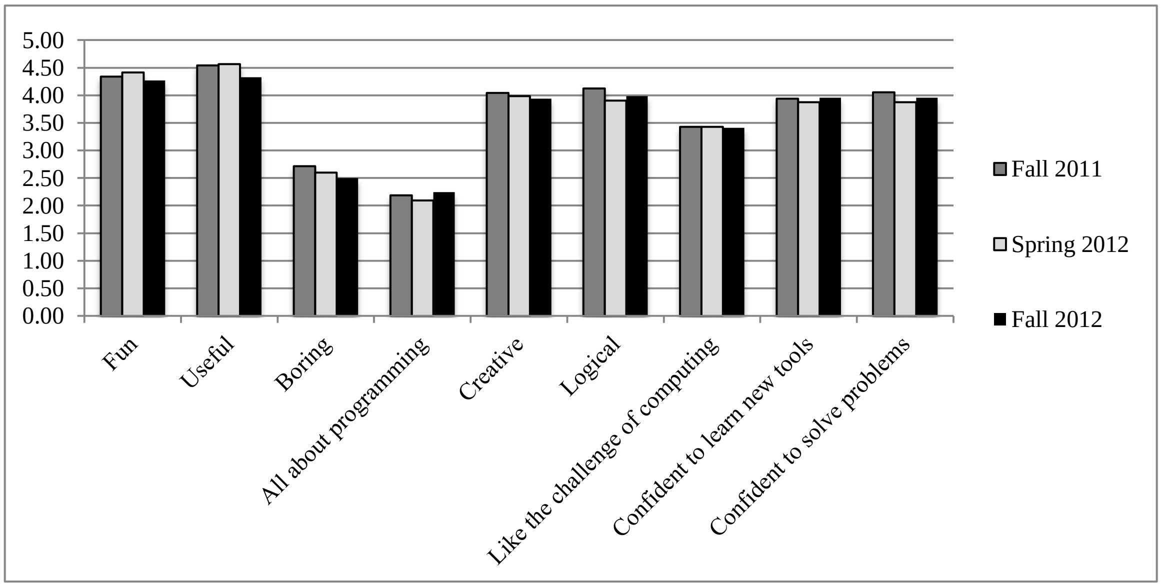


Figure 9 Computing Attitudes

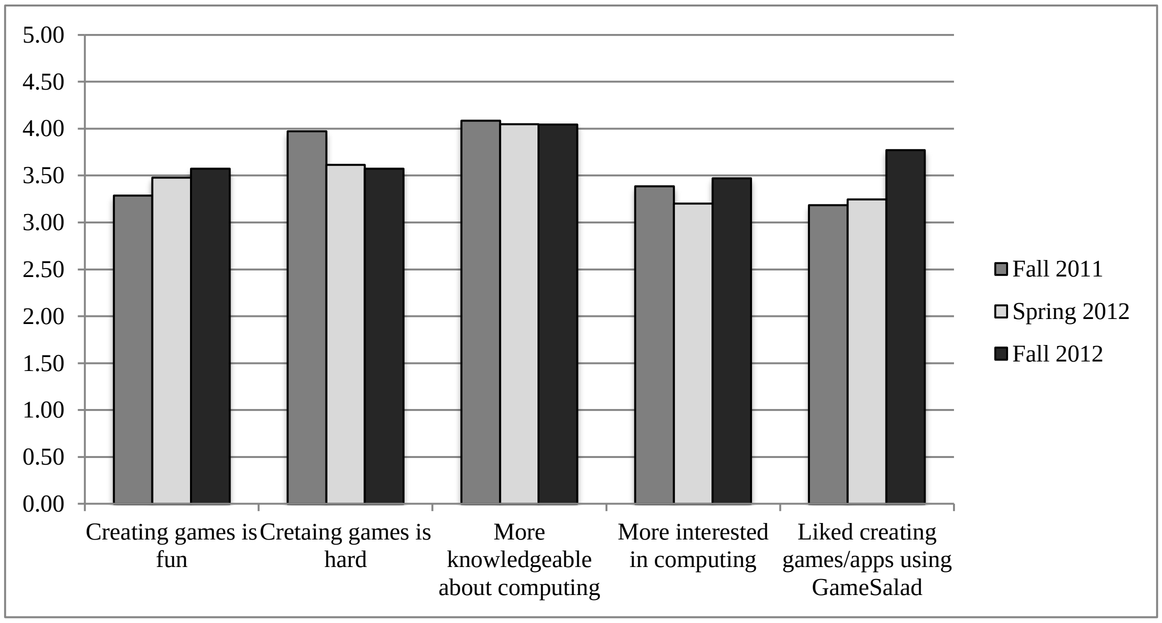


Figure 10 Game Creation Impact

In the pre and post problem solving assessment results students demonstrated significant improvement in the design and test planning areas. In Fall 2012, the problem formulation scores did not show any change, but in Spring 2013 problem formulation scores also improved along with the others. Running paired samples t-tests confirmed our findings, the results indicate (in the 95% confidence interval):

- The mean difference between problem formulation pre and post quiz scores is between 1.99% and 13.21%, with the post quiz scores being higher.
- The mean difference between solution design pre and post quiz scores is between 22.51% and 35.75%, with the post quiz scores being higher.
- The mean difference between test planning pre and post quiz scores is between 7.83% and 13.52%, with the post quiz scores being higher.

These numbers are encouraging. Even though most of these students will not take a programming course, they obtained some of the benefits of such a course conducted in a non-traditional manner, using inquiry and creativity to help them learn problem-solving and critical thinking. The results of the pre and post quiz in Fall 2012 and Spring 2013 are shown in figures 12 and 13.

Students' games were scored using a rubric adapted from (Deek et.al. 1999) to evaluate solution quality. Of the projects graded, 76.67% of games submitted were reliable, working for all valid inputs and responding to all invalid inputs. Out of all the projects, 83.33% of the games submitted were consistent with their game designs and worked correctly.

4.2 Qualitative Results

The open-ended questions in the survey gathered student feedback about the strengths, challenges and recommendations for this course. The most recurring idea observed from the student responses was to spend more class time on GameSalad. Following strengths were identified:

- Ability to use a wide variety of software
- Interaction with industry experts

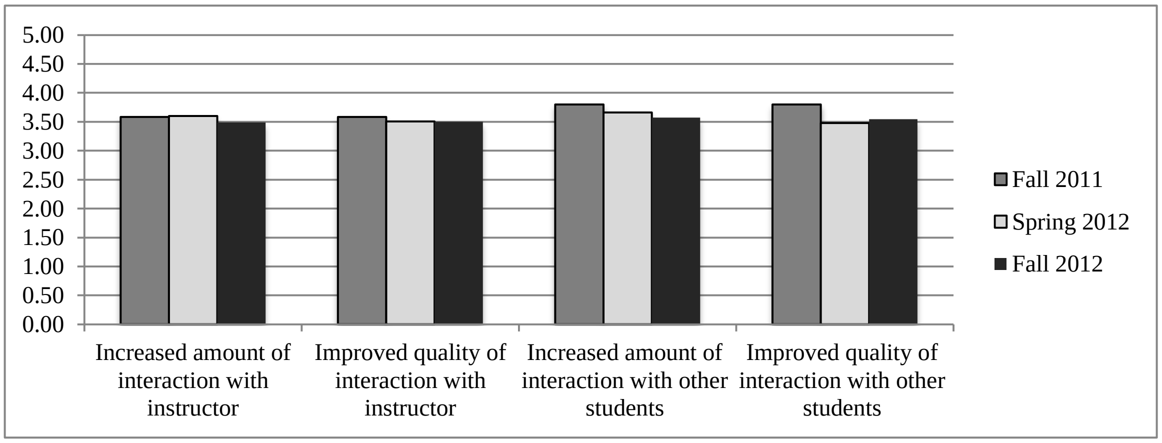


Figure 11 Student Engagement

- Increased amount and quality of student-faculty interaction
- Increased amount and quality of student-student interaction
- Ability to create mobile games

Among the challenges, the amount of work done and amount of time spent were the most common themes. Many students indicated their desire to spend more class time on GameSalad. They found the process of game creation challenging, but fun. They also complained that they spent a lot of time on the project, but indicated they were satisfied with their accomplishment.

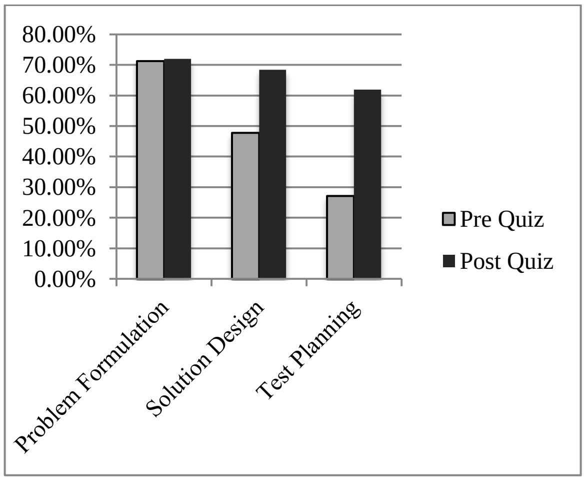


Figure 12 Fall 2012 Average Scores on Problem Solving Assessment

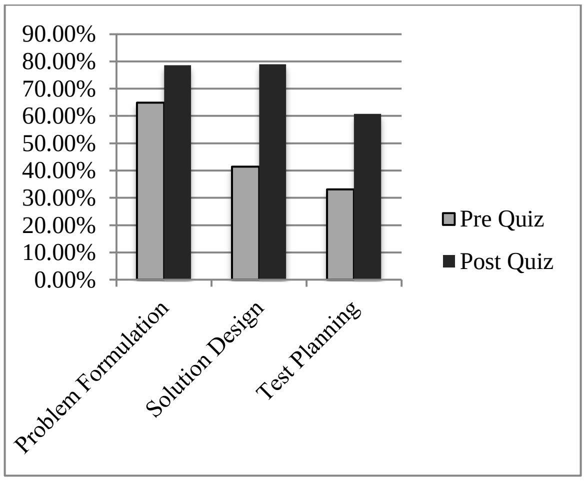


Figure 13 Spring 2013 Average Scores on Problem Solving Assessments

The pre and post quizzes included reflection questions to gain insight into students' problem solving processes. The most significant difference we noticed in student responses from pre to post quiz was their confidence. When asked about their problem solving experience, most students indicated they felt more knowledgeable and confident in solving the problem on the post quiz. Even though the games in the quizzes were unlike any of the games students created or studied, they noted they applied the same problem-solving process as they had learned in class. Reflection questions suggested students understood the methodology of software design and were able to apply it to different contexts. Another significant difference from pre to post quiz was of planning test cases. When asked to describe the solution design process on the post quiz, many students described various test cases they considered when designing the solution. Finally, student responses also indicated that the visual nature of the problem (ability to play the game) played a key role in formulating the problem correctly. This may explain the insignificant difference between the pre and post problem formulation scores.

4.3 Faculty Perspectives

Faculty discussions were useful in identifying the strengths and challenges of this approach. From faculty perspective using mobile game development was an ideal way of integrating logical thinking and problem solving in the curriculum for students that opted out of the introductory programming course. iPads were used in the class by students to test the end result (game) they were expected to create in class on that day. This provided an excellent way to discuss how many actors were needed, what their attributes would be and what their behaviors would be. This problem-solving approach worked very well for small class activities, which the students could complete in an hour and a half. It allowed the instructors to demonstrate the problem-solving process. For the larger project, the students sought instructor's assistance on specific issues. They were able to use the problem-solving process to design their own game and implement it. The hands-on active learning approach also created an environment of high engagement among students as well as between students and faculty.

Faculty also identified challenges with this approach, mainly those of time constraints and compatibility. More time could not be allotted to GameSalad activities in the semester due to other course objectives and requirements that had to be met. The unavailability of Mac operating system at students' home affected the amount of time students spent on GameSalad outside of class. Constant updates to both GameSalad and the Mac OS were also major challenges that faculty faced. The process of preparing iPad devices to test GameSalad apps on them is complex and cannot be done in class with students. This had to be done separately before the semester began. During the class session student products were uploaded on the iPad devices. Since this process requires Wi-Fi communication between the iPad and the computer it might pose network security challenges on some campuses.

An alternative to GameSalad is AppInventor, which was formerly managed by Google. AppInventor is similar to GameSalad and can be used to create mobile apps/games for Android platform. It is important to note that AppInventor has also gone through a few major changes and was unavailable for some period of time as it switched hands from Google to MIT. It seems to the authors that this is cutting-edge technology, still in its infancy and constantly evolving. But the ease of creation of mobile apps, the visual impact and student interest in mobile apps are too significant for educators to ignore.

5. CONCLUSION

The popularity of mobile devices and mobile apps, the interactive drag and drop game creation software and the immediate visual feedback provided by them can be leveraged to engage students in computing classes. A visual game development tool can be used to inculcate problem-solving and logical thinking skills among students. In this project, students investigated existing educational mobile apps, conducted research on mobile game development and user interface design principles and created an educational app in their area of academic interest using GameSalad. The participants reported the game development experience to be both challenging and fun. The participants' self-reported attitudes towards computing were positive. The student-faculty and student-student engagement was reported higher as compared to other classes. Finally, the pre and post quizzes demonstrated significant improvement in students' problem solving skills. The faculty expressed some concerns regarding platform dependence and frequently updating versions of the software. However, they also reported that the positive results greatly outweighed the concerns expressed. Moreover the availability of other applications such as AppInventor and constantly improving mobile platforms and game creation software make mobile app development an attractive learning module to be included in the introductory computing curriculum.

6. ACKNOWLEDGMENTS

The authors would like to thank Georgia Gwinnett College's STEM Mini Grant program for their funding. The authors would also like to thank Mr. Richard Price for reviewing the paper.

7. REFERENCES

Bennedsen, J., & Caspersen, M. E. (2007). Failure rates in introductory programming. ACM SIGCSE Bulletin, 39(2), 32-36.

Carver, J. C., Henderson, L., He, L., Hodges, J., & Reese, D. (2007, July). Increased retention of early computer science and software engineering students using pair programming. In Software Engineering Education & Training, 2007. CSEET'07. 20th Conference on (pp. 115-122). IEEE.

Clement, M., Posada, D. C. K. A., & Crandall, K. A. (2000). TCS: a computer program to estimate gene genealogies. Molecular ecology, 9(10), 1657-1659.

Cliburn, D. C. (2006, October). Experiences with the LEGO Mindstorms throughout the undergraduate computer science curriculum. In Frontiers in Education Conference, 36th Annual (pp. 1-6). IEEE.

Cliburn, D. C., & Miller, S. (2008, March). Games, stories, or something more traditional: the types of assignments college students prefer. In ACM SIGCSE Bulletin (Vol. 40, No. 1, pp. 138-142). ACM.

Computing Research Association. (2011). Taulbee Survey 2009-2010. Retrieved August 19, 2013 from http://www.cra.org/resources/taulbee.

Crawford, S., & Boese, E. (2006). ActionScript: a gentle introduction to programming. Journal of Computing Sciences in Colleges, 21(3), 156-168.

Deek, F. P., Hiltz, S. R., Kimmel, H., & Rotter, N. (1999). Cognitive assessment of students' problem solving and program development skills. Journal of Engineering Education, 88(3), 317-326.

Dodds, Z., Libeskind-Hadas, R., Alvarado, C., & Kuenning, G. (2008, March). Evaluating a breadth-first CS1 for scientists. In ACM SICSE Bulletin (Vol. 40, No. 1, pp. 266-270). ACM.

Dougherty, J. P. (2007). Concept visualization in CS0 using ALICE. Journal of Computing Sciences in Colleges, 22(3), 145-152.

Forte, A., & Guzdial, M. (2005). Motivation and nonmajors in computer science: identifying discrete audiences for introductory courses. Education, IEEE Transactions on, 48(2), 248-253.

Fowler, R. D. (1985). Landmarks in computer-assisted psychological assessment. Journal of Consulting and Clinical Psychology, 53(6), 748.

Garrison, D.R. & Vaughan, N.D. (2008). Blended Learning in Higher Education: Framework, Principles and Guidelines. Jossey-Bass.

Guzdial, M., & Ericson, B. (2007). Introduction to computing & programming in Java: a multimedia approach. Pearson Prentice Hall.

Johnsgard, K., & McDonald, J. (2008, April). Using Alice in overview courses to improve success rates in programming. In Software Engineering Education and Training, 2008. CSEET'08. IEEE 21st Conference on (pp. 129-136). IEEE.

Kelleher, C., Pausch, R., & Kiesler, S. (2007, April). Storytelling Alice motivates middle school girls to learn computer programming. In Proceedings of the SIGCHI conference on Human factors in computing systems (pp. 1455-1464). ACM.

Klassner, F., & Anderson, S. D. (2003). Lego MindStorms: Not just for K-12 anymore. Robotics & Automation Magazine, IEEE, 10(2), 12-18.

Kölling, M., & Henriksen, P. (2005). Game programming in introductory courses with direct state manipulation. ACM SIGCSE Bulletin, 37(3), 59-63.

Lawhead, P. B., Duncan, M. E., Bland, C. G., Goldweber, M., Schep, M., Barnes, D. J., & Hollingsworth, R. G. (2002, June). A road map for teaching introductory programming using LEGO© mindstorms robots. In ACM SIGCSE Bulletin (Vol. 35, No. 2, pp. 191-201). ACM.

Leutenegger, S., & Edgington, J. (2007, March). A games first approach to teaching introductory programming. In ACM SIGCSE Bulletin (Vol. 39, No. 1, pp. 115-118). ACM.

Malan, D. J., & Leitner, H. H. (2007). Scratch for budding computer scientists. ACM SIGCSE Bulletin, 39(1), 223-227.

McDowell, C., Werner, L., Bullock, H. E., & Fernald, J. (2006). Pair programming improves student retention, confidence, and program quality. Communications of the ACM, 49(8), 90-95.

Microsoft Research FUSE LABS (n.d.). Kodu. In Kody Game Lab Community. Retrieved Dec 18, 2013, from http://www.kodugamelab.com/.

Mullins, P., Whitfield, D., & Conlon, M. (2009). Using Alice 2.0 as a first language. Journal of Computing Sciences in Colleges, 24(3), 136-143.

Nagappan, N., Williams, L., Ferzli, M., Wiebe, E., Yang, K., Miller, C., & Balik, S. (2003, February). Improving the CS1 experience with pair programming. In ACM SIGCSE Bulletin (Vol. 35, No. 1, pp. 359-362). ACM.

Rizvi, M., Humphries, T., Major, D., Jones, M., & Lauzun, H. (2011). A CS0 course using scratch. Journal of Computing Sciences in Colleges, 26(3), 19-27.

Ruthmann, A., Heines, J. M., Greher, G. R., Laidler, P., & Saulters II, C. (2010). Teaching computational thinking through musical live coding in scratch. In Proceedings of the 41st ACM technical symposium on Computer Science Education (pp. 351-355). ACM.

Spertus, E., Chang, M. L., Gestwicki, P., & Wolber, D. (2010). Novel approaches to CS 0 with app inventor for android. In Proceedings of the 41st ACM technical symposium on Computer science education (pp. 325-326). ACM.

Teague, D., & Roe, P. (2008). Collaborative learning: towards a solution for novice programmers. In Proceedings of the tenth conference on Australasian computing education-Volume 78 (pp. 147-153). Australian Computer Society, Inc.

Turner, E. H., Albert, E., Turner, R. M., & Latour, L. (2007). Retaining majors through the introductory sequence. ACM SIGCSE Bulletin, 39(1), 24-28.

Uludag, S., Karakus, M., & Turner, S. W. (2011). Implementing IT0/CS0 with scratch, app inventor for android, and lego mindstorms. In Proceedings of the 2011 conference on Information technology education (pp. 183-190). ACM.

Wellman, B. L., Davis, J., & Anderson, M. (2009). Alice and robotics in introductory CS courses. In The Fifth Richard Tapia Celebration of Diversity in Computing Conference: Intellect, Initiatives, Insight, and Innovations (pp. 98-102). ACM.

Wolber, D. (2011, March). App inventor and real-world motivation. In Proceedings of the 42nd ACM technical symposium on Computer science education (pp. 601-606). ACM.

Williams, L., Kessler, R. R., Cunningham, W., & Jeffries, R. (2000). Strengthening the case for pair programming. Software, IEEE, 17(4), 19-25.

AUTHOR BIOGRAPHIES

Sonal Dekhane is an Associate Professor of Information Technology at Georgia Gwinnett College (GGC). At GGC, Dr. Dekhane contributed to curriculum development and led program assessment initiatives. She is committed to increasing student engagement in her classroom and experiments with a variety of pedagogical techniques to achieve this goal. She has successfully used mobile technology to enhance students' learning experience. Her work has been presented at national and international conferences and published in national and international journals such as International Journal of Mobile and Blended Learning. Dr. Dekhane is currently involved in projects focusing on increasing women's participation in computing at GGC.

Xin Xu received her Ph.D. degree in Computational Analysis and Modeling with a concentration in network analysis and modeling from Louisiana Tech University. She has been working as an Assistant Professor of Information Technology at Georgia Gwinnett College since fall 2007. She found her passion for teaching while working as a graduate teaching assistant. Her current academic interests include researching the impact of using multimedia on student learning outcomes and developing web animations to assist STEM teaching and learning, as well as researching and developing strategies to promote technology in woman and underrepresented minorities.

Mai Yin Tsoi, a native of California, is an Associate Professor who has been teaching General and Organic Chemistry since she joined the faculty at Georgia Gwinnett College in 2007. She received a Bachelor's in Chemistry from Vassar College, NY, a Master's in Organic Chemistry from the Georgia Institute of Technology, and a Doctorate of Philosophy in Science Education from the University of Georgia. She has conducted oceanographic research at the Massachusetts Institute of Technology, Texas A & M University, and the U.S. Naval Research Center in San Diego, CA. Dr. Tsoi was named Gwinnett County Teacher of the Year and was the 2nd finalist in the Georgia State Teacher of the Year competition, both in 2006, for excellence in teaching high school chemistry and physics. Her published papers and current research center on the use of technology in science teaching and learning, with a focus on mobile learning and software development. Recently, her scholarly work has garnered the 2011 Excalibur Award from the Technology Association of Georgia and the 2012 Catalyst Award in Mobile Innovation from Blackboard, as well as an honorary induction into the Golden Key International Honor Society for her leadership and service.

ISSN 1055-3096