Outreach for Improved Student Performance: a game design and development curriculum

Katelyn Doran, Acey Boyce, Samantha Finkelstein, and Tiffany Barnes

Department of Computer Science
University of North Carolina at Charlotte
9201 University City Blvd., Charlotte, NC, USA
1-704-687-8577

ABSTRACT

We present a curriculum for computer science outreach using Game Maker. This curriculum has been adapted over six iterations of a 10-week, middle school apprenticeship on Game Design and Development. Through multiple iterations we have adjusted for many of the issues one could expect to encounter when running a similar outreach program. While many outreach curricula are independent from coursework, our video game design curriculum is targeted to address generalized student learning objectives (Math and English Language Arts) and designed for integration into middle school classrooms. We demonstrate that students' language arts and math classroom performance have improved with participation in this apprenticeship.

Categories and Subject Descriptors

K.3.2 [**Computers and Education**]: Computer and Information Science Education – *curriculum, computer science education.*

General Terms

Curriculum design, middle school

Keywords

Education, middle school, game design and development

1. INTRODUCTION

Among high school students taking the national Advanced Placement Computer Science exam, only 17% are women and the combined percentage for African-American and Hispanic students is about 6% (Durndell, 2000). This discrepancy in race and gender representation in Computer Science, as compared to the general population, is referred to as the *digital divide*. The STARS Alliance (starsalliance.org), an organization that spans 32 schools in the United States, is working to close the digital divide in computing disciplines through multi-faceted interventions, many of which are targeted at K-14 students. Through the STARS Alliance, we have been conducting game development outreach to middle and high school students every semester from Fall 2006 to the present. Our work has evolved into a computing after-school program for middle school students. We believe, as Margolis suggests, that one way to close the digital divide is to help students move from learning *how to use* technology to learning *how to create* technology (2003). This philosophy is central to our outreach, where we help students become creators.

We partner with Citizen Schools, a non-profit organization that offers after-school services to middle school children with the long-term goal of teaching leadership and professional skills through apprenticeships, which are hands-on learning experiences with experts (citizenschools.org). For the past six semesters, we have led a Citizen Schools game design and development apprenticeship that teaches students how to develop a game prototype using Game Maker (yoyogames.com). Game Maker is an object-oriented programming environment that provides students with a drag-and-drop interface for creating the functionality of their object classes, which control in-game items. This software enables inexperienced students to learn high-level aspects of computing without being overwhelmed by issues of syntax. In this paper, we present our lessons learned over six semesters of outreach, our refined curriculum, and report on our program's effectiveness at improving student performance in standard curriculum as compared to other apprenticeships.

ITiCSE'12, July 3–5, 2012, Haifa, Israel.

2. RELATED WORK

Games can be used to motivate students to engage in their coursework and to foster interest in programming (Repenning, 2007; Cliburn 2008, Kurvovsky 2007). It has been shown that using games to teach computer science concepts and providing students with game-based homework assignments results in higher class attendance (Burd, 2007), better grades, and increased student retention (Clua, 2006). Perhaps this is because games provide a context that many students can relate to, making the abstract ideas of computer science more socially relevant and grounded. The potential of games as a pedagogical tool has motivated a surge in game-based learning to clarify confusing concepts (Barnes et al., 2007, Leutenneger, 2007, Sweedyk, 2005) and tool development to increase student engagement (Bayliss, 2006, Clua et al., 2006). The results of research on the use of games in the computing curriculum suggest they are both popular with college students and effective: after departments began to implement game-based curriculums, professors noted an appreciable difference in attendance and retention, with students strongly preferring game-based assignments (Cliburn, 2008, Clua et al., 2006). Additionally, many students state that their reason for enrolling in a computer science program was due to their desire to create games (Parberry et al. 2005, Wallace et al., 2009).

At the middle school level, where the goal is to generate future interest in computing, not to teach technical programming skills, games are a particularly promising tool. As a result, in our outreach sessions, we teach programming logic and structure and avoid overtly introducing programming syntax, which we feel may turn students away from the field rather than attract them.

3. EVOLUTION OF A PROGRAM

3.1 Single-Session Outreach

We began with *single session* outreach sessions at the middle, high school, and early college levels from 2007-2009. Our focus for these sessions was on the "My First Game" tutorial, provided by YoYoGames, which features a simple game called "Catch the Clown." During these sessions, students responded best when we encouraged them to make mistakes rather than avoid it. For instance, when students first ran the game, the clown began moving in a random direction…right off the edge of the screen. Instantly hands were raised and students cried out "He went off the screen!" Calmly, we replied, "well, did you tell him not to?" What better way to teach the concepts of collision and AI than by showing the effects of not including necessary code? This resulted in our intentional inclusion of several *guided errors* in our curriculum. These errors encourage the students to utilize debugging logic, familiarize themselves with how game elements look when they go wrong, and avoid similar errors in the future.

3.2 Extended Outreach – First Iteration

In Spring 2009, we began teaching our first 10-week Game Maker Apprenticeship with Citizen Schools. Our aim in this initial foray into extended outreach was to give our apprentices a full understanding of the game design and development process. We made several mistakes, including

- Forming unreliable student pairs
- Allowing students to make any game they wished
- Not managing student expectations

We chose to have students work in pairs to model the success of pair programming in computing college courses. However, this did not work well for this iteration, as certain students were absent frequently or dropped out of the program. In these cases, some students were left without a partner and struggled to keep up with the pace of the apprenticeship. Had we worked with our Citizen Schools partners to identify students with behavioral and attendance issues, we could have likely reduced the impact of these issues. Additionally, had our lessons been more generalized, the absence of one partner from a group would not have resulted in that group falling behind.

Allowing students to make any game they could imagine meant we had to provide different instruction for each pair of students. All nine student pairs had their own game idea, varying from platform-based games to basketball games. To accommodate this, we had to have nine separate lessons every week to address the completely different games being made. While one group was learning how to add combat into their game, another was learning how to make a scrolling background. With only two instructors, we were quickly overwhelmed.

The largest problem we faced were student attitudes towards the apprenticeship. Students joined expecting to make *Halo* while we would have been impressed with *Space Invaders*. Young students, with no way of knowing how complex programming and game development can be, will have unrealistic expectations of what they can create in ten weeks. Our students had lofty goals for their games and, when they were unable to attain these goals in ten-weeks, they grew increasingly disappointed and frustrated throughout the apprenticeship.

3.3 Extended Outreach – Second Iteration

Based on the lessons learned in Spring 2010, we updated our approach to include more structure and more development time. Our primary focus during the Fall 2009 apprenticeship was to keep the students' ideas in check by making it clear to them the results they could expect by the end of the apprenticeship to avoid disappointment with their final products. We managed student expectations most effectively by showing them sample Game Maker games and avoiding referencing unfamiliar games.

To help create realistic expectations, we asked students to imagine they were working for a game design company. Students were asked to utilize the ten-week program to create prototype games that we could then play test to select our next big title. It was stressed that prototypes focus on the core game play mechanics rather than artwork and appearances. By informing students that they would be creating testable prototypes, rather than complete games, they had far more realistic expectations.

We limited the number of game types students could choose to make to three, which still let students feel they could choose what to make but helped reduce our preparation and support loads. Student groups could choose to make a platformer, a scrolling shooter, or a dungeon crawler. This allowed us to prepare just three lesson plans per session, rather than one for each student group. It also made it possible to maintain student pairs, because if a student's partner left the program they could easily join another pair working on the same type of game rather than be left with the task of completing an entirely unique game on their own.

Interestingly, groups making games within the same genre still felt their resulting game was unique and expressed their own creative ideas – even if many of the differences were only in the selected graphics. The end result was that students this session appeared to be far more satisfied with their final products than students in the previous session. We continued using this variation of our curriculum through Spring 2011.

3.4 Extended Outreach – Third Iteration

In Fall 2011 several changes were made to our existing curriculum. Most importantly, the games were limited to one game type – the dungeon crawler game. This change allowed for more focused lessons to the whole group followed by independent work by each student team. With this new approach, our curriculum resembles more of a traditional teaching style and better fits the Citizen Schools guidelines to focus on depth of subject.

In both of our previous iterations, we had many lessons where students simply followed along with the steps the instructors were completing. To amend this, we began placing more emphasis on instruction time and connecting with the students in order to help them better relate to the material and enable them to complete more tasks independently. Each lesson changed from having one primary goal to be completed to having a small lesson paired with a series of activities featuring a decreasing level of instructor guidance. This curriculum combines the lessons learned from six semesters of outreach and is our recommended curriculum.

4. RECOMMENDED CURRICULUM

Here we present an overview of our 10-week curriculum, which incorporates the lessons learned from our experiences. It was developed to engage 15-20 students in a once-per-week game development apprenticeship for 10 weeks. The goal of the curriculum is to foster student *creation* of technology, rather than

student *use* of technology, as a way of attracting middle school students to computing and improving their overall classroom performance. Each 80-minute lesson follows the same basic structure including set objectives, a hook, a mini-lesson, activities, and a final check for understanding.

Table 1: Curriculum Scope and Sequence

Lesson 1: Catch the Clown	• ***Introductions*** • ***Hook:*** *Favorite game* • ***Mini-lesson:*** *Game Development Process* • ***Activity 0:*** *Play Catch the Clown* • ***Activity 1:*** *Create Game Objects* • ***Activity 2:*** *Debug Game* • ***Activity 3:*** *Expand Game* • ***Check for Understanding (CfU):*** *Exit Ticket*
Lesson 2: Main Characters and Basic Levels	• ***Hook:*** *Favorite main character* • ***Mini-lesson:*** *What a main character does* • ***Activity 1:*** *Create a placeholder character* • ***Activity 2:*** *Create user interactions* • ***Activity 3:*** *Create basic level* • ***CfU:*** *Review main character purpose and vocabulary*
Lesson 3: Treasures and Goals	• ***Hook:*** *Best in-game reward* • ***Mini-lesson:*** *The purpose of rewards in games* • ***Activity 1:*** *Create treasure that adds points* • ***Activity 2:*** *Create goal points* • ***Activity 3:*** *Create additional levels* • ***CfU:*** *Review if-then logic and pseudo-code*
Lesson 4: Sprites and Advanced Levels	• ***Hook:*** *Best art in a game* • ***Mini-lesson:*** *Importance of art in game design* • ***Activity 1:*** *Have pictures taken* • ***Activity 2:*** *Create advanced levels* • ***Activity 3:*** *Tie levels together* • ***CfU:*** *Review game art's value, explain game structure*
Lesson 5: Enemies and Title Screens	• ***Hook:*** *Best game enemies* • ***Mini-lesson:*** *How enemies affect game difficulty* • ***Activity 1:*** *Create Title Screen* • ***Activity 2:*** *Create Lives controller* • ***Activity 3:*** *Create basic enemy* • ***CfU:*** *Review pseudo-coding, discuss enemy importance*
Lesson 6: Attacks and Advanced Enemies	• ***Hook:*** *Best game ability* • ***Mini-lesson:*** *Balancing game attacks and enemies* • ***Activity 1:*** *Group pseudo-codes attacks and advanced enemies* • ***Activity 2:*** *Implement attacks* • ***Activity 3:*** *Implement advanced enemies* • ***CfU:*** *Review balancing and pseudo-coding principles*
Lesson 7: Variables and Loops	• ***Hook:*** *Best super power* • ***Mini-lesson:*** *How power-ups impact a game* • ***Activity 1:*** *Group pseudo-coding* • ***Activity 2:*** *Implement 360 attack* • ***Activity 3:*** *Add power up to game* • ***CfU:*** *Discuss pseudo-code, variable, and looping*
Lesson 8: Play Testing	• ***Hook:*** *What we've learned* • ***Mini-lesson:*** *Game design from beginning to end* • ***Activity 1:*** *Finish all art assets* • ***Activity 2:*** *Finish all level design* • ***Activity 3:*** *Play test game completely* • ***CfU:*** *Discuss game design process*
Lesson 9: The Elevator Pitch	• ***Hook:*** *A time we've spoken for an audience* • ***Mini-lesson:*** *Watch 6th grader T.Suarez's TEDx talk* • ***Activity 1:*** *Lecture on presentations, pitches, and constructive criticism* • ***Activity 2:*** *Prepare elevator pitches* • ***Activity 3:*** *Practice & critique pitches with partner* • ***CfU:*** *Present pitches for the class*
Lesson 10: Final Practice	• ***Hook:*** *Practice pitches* • ***Mini-lesson:*** *Review set up for final presentation* • ***Activity 1:*** *Final Play Test* • ***Activity 2:*** *Practice Group 1* • ***Activity 3:*** *Practice Group 2* • ***CfU:*** *Discuss final presentations, congratulate students*

4.1 In Depth Look at a Lesson

To provide a better explanation of how our Lessons cover materials and address certain classroom concerns, we present an in-depth look at one of the lesson plans from the curriculum – Lesson 2: Main Characters and Basic Levels.

4.1.1 Overview

Students will create player-controlled game objects that will serve as the main character of games. These objects will be controlled by keyboard interactions, which the students will program. Students will also create some basic, initial levels for their main characters to navigate.

4.1.2 Objectives

- Explain the uses of specific technological instruments and tools used in the apprenticeship
- Define technical terms and vocabulary associated with the apprenticeship

4.1.3 Connections

This lesson will focus on some key concepts in Computer Science including if-then statements and pseudo code

4.1.4 Materials

- A data projector capable of connecting to a laptop
- A projection screen
- Dry erase board and markers
- Enough computers for students to work in pairs

4.1.5 Lesson Preparation

- Space: Any classroom space is fine, but it is ideal to have space where students can easily sit side by side in pairs. Access to functional power outlets is helpful when working with laptops.
- Group: Students will be divided into pairs for this lesson; these pairs will remain consistent throughout the apprenticeship.
- Resources: Computers should have Game Maker already installed as well as pre-formatted sprite images. Ideally, the Game Maker short cut and the sprite images folder should be located on the desktop.

4.1.6 Hook: Who is your favorite main character?

- Do Now / Warm Up – Instructors introduce themselves and their favorite main character. We then go around the classroom, having each student introduce him- or her-self and state his or her favorite main character. Students can name any character, even if they are not from a video game, but cannot repeat a character that has already been named.
- Transition – After sharing their favorite main character, we ask students what it is that makes those characters their favorites. It is likely that all student answers will mention contributions main characters make to games, stories, and television shows. We will use these responses to talk about what contributions a main character makes to a video game during the Mini-Lesson.
- Student Perspective – Students might mock each other over their favorite characters. However, it should be emphasized that everyone has different preferences on characters, video games, and almost everything in life. One of the strengths of game

design is the ability to make a game that includes all of the things one likes while leaving out the things they do not.

Mini-Lesson: What a Main Character does

The instructor gives an overview of what a main character does in general, and within the context of a video game. The focus should be on how main characters further plots, change stories, and create atmosphere in games. Possible examples are Mario (whose relationship with Princess Peach drives the game's plot), and The Hero of Oakvale from Fable (where the main character's choices change the story).

4.1.7 Activity 1: Create the Game Object

This will be a group-based activity with the Instructor leading by completing the task where it can be seen on the projector. Students should follow along, completing the steps with their partner on their computers, and participate in the discussion.

1. Create a placeholder sprite for the main character. This can be done as a group with students taking the lead, using the skills they learned in Lesson 1.
2. Associate the new sprite with a new game object . Again, students provide the steps based on the previous lesson.
3. Students should repeat the above steps, on their own, to create the Wall object.
4. Instructor should confirm that students have completed step three, and review the process one more time to ensure each student grasps how to create sprites and objects.

- Students might get caught up in what image is used for their Main Character. If this happens, remind them that a future lesson will focus on creating custom sprites using the students' own pictures and that these will then serve as the main characters, so art should not be a focus right now.

4.1.8 Activity 2: Program User Interaction

We will now add user interactions to the main character game object; these will control how the character moves when certain buttons are pressed.

1. Instructors talk through the code that controls the main character movement and write this out as pseudo code
2. Students implement the pseudo code in Game Maker, adding in movement for Up, Down, Left, and Right when the corresponding arrow keys are pressed
3. Have students create a basic room where they place walls and their character to test the new interactions. Students should rapidly realize that their character can walk off the screen and through walls.
4. Talk through the pseudo code for what actions should occur when the main character collides with the wall
5. Students will implement the pseudo code made in step four

Using pseudo code at this point helps make a connection beyond Game Maker to Computer Science in general. This also helps establish good problem solving practices, as students are encouraged to think about a solution before implementing it.

4.1.8.1 Activity 3: Create Basic Levels

Students should now have a moveable main character that cannot walk through walls. Students should use the objects they have created to design and implement new levels to navigate.

1. Do a quick refresher on how to create a new room and place elements inside
2. Have the students create their own, unique rooms that will serve as the game levels. Maze-like layouts are a great place to start.
3. Ask the students to share what new features and design decisions they made.

At this point in the apprenticeship, students will not yet be able to connect their levels to have one, linear game. Let the students know that this will be done in the future, but for now they will have to test each room separately.

4.1.9 Check for Understanding

- Ask students what the steps are to create a new game object.
- Ask students what role a main character serves in a game.
- Ask students to state what the pseudo code is for telling a character to speed up when the Space Bar is held.
- Ask students what other activities a game object could have, previewing Lesson 3 on Treasure and Goals.

4.2 Lesson Breakdown

As previously noted, lessons all follow the same basic format. Lesson 1 differs slightly with additional introductions and an extra activity that take up 10 minutes of the lesson time. Lessons 8 and 9 have a slightly different focus, but otherwise the format is the same. Here we will discuss the primary components of each lesson plan and what they contribute to our curriculum.

4.2.1 The Hook

Each lesson begins with an initial, 10 minute Hook activity. The purpose of the hook is three-fold. First, it provides an opportunity to learn (and remember) the names of students participating in the program. Secondly, it forms a connection between the students, their interests, and the subject matter of the day's lesson. Finally, it provides an important way for instructors to evaluate student improvement in speaking to an audience. Each Hook invites the students to introduce themselves to the class and share an opinion relating to a certain prompt.

For example during the Hook from Lesson 2 (detailed in 4.1.6), a student who selects Mario as their favorite main character probably likes cartoon-style, platform games while a student who selects Master Chief from the Halo series is likely partial to first-person shooter style games with high levels of combat. By identifying these preferences through the Hook activity, instructors can better connect to the students and help guide them in making a game that will suit their personal tastes.

Early in the apprenticeship, the Hook is handled informally. As the program continues, slight changes are made to provide students with experience presenting. These changes include standing up, removing hands from pockets, and projecting voices. A new change to how the Hook response must be provided is added each week until students are responding in a manner that closely reflects how they will deliver their final presentations. In this way, students are practicing for these presentations throughout the apprenticeship, not just in the final two lessons.

4.2.2 Mini-Lesson

From the Hook, each lesson moves into a 10 minute Mini-Lesson. The Mini-Lesson addresses one component related to the overall lesson plan. Mini-Lessons seek to provide context for the students to help them better understand how their apprenticeship activities are relevant to Game Design as a field.

By using the student responses from the Hook within the Mini-Lesson, the material has a direct connection to student interests,

thus engaging them in what is otherwise more of a lecture than a hands-on component.

4.2.3 Activities

Lesson activities provide the hands-on component to our curriculum. Each Activity engages the students and requires their participation on computers. Activities total 55 minutes of each lesson. We provide a breakdown of time for each below, but these times sometimes differ due to the specific needs of a lesson.

4.2.3.1 Activity 1: We do (10 minutes)

This activity is referred to as the "we do" activity, which means the instructors complete the steps with the students following along. The instructor completes each step in Game Maker on their computer, which is projected for the students to see. Students follow along on their computers as the instructors complete each step of the activity. Activity 1 always provides the basic knowledge necessary for completing the Lesson.

4.2.3.2 Activity 2: We do/ You do (20 minutes)

Activity 2 is the "we do/you do" activity, building on Activity 1 to create a new, more complex game element. Instructors prompt the students here, but students are expected to determine the proper steps and their order. Instructors will ask the students what steps are necessary and guide them in the right direction as they seek to complete these steps. This work is done with a group discussion, where students are provided the opportunity to help each other and amend one another's incorrect responses. Only when the group reaches a consensus for the next step do we actually move forward into implementation.

Activity 2 is also where we incorporate pseudo code. For many Lessons, in the "we do" portion of the activity, instructors present and explain pseudo code to the students. The students then complete the "you do" portion by implementing that pseudo code within the Game Maker environment. The interaction that instructors have with the students during Activity 2 helps to maintain the classroom pace and ensure that students keep up with and understand the material.

4.2.3.3 Activity 3: You do (25 minutes)

The third and final activity builds upon the topics covered in the previous two activities as well as those from all previous Lessons. During Activity 3, students are given a task to complete and then left on their own to complete it. In order to ensure that students are challenging themselves and truly mastering the material, rather than simply relying on their instructors, we use the first 5-10 minutes of Activity 3 for students to work with their partners with no instructor input. We have seen that, when given the option, students will tend to ask for help before they attempt to recall on their own how to complete a task. By leaving the students to their own devices for the first part of Activity 3, we encourage them to explore what they are capable of doing independently. Often, no instructor help is needed.

4.2.4 Check for Understanding

The Check for Understanding is a means of assessing how well the concepts taught in a lesson were understood by the students. This component of a lesson can take a variety of forms, including formal "Exit Tickets" that serve as written evaluation of the topics. We initially intended for our curriculum to utilize Exit Tickets, but found that the students did not enjoy completing what was, essentially, a quiz after 75 minutes of instruction, so we adopted a more informal method.

Our Check for Understanding is a series of "teach-back" questions where the students are prompted to explain certain concepts in their own terms, walk through the steps of how to complete a task, or describe some insight they had during the lesson. If students struggle to respond to teach-back questions, it allows instructors to adjust their plans to cover the topics later in more depth. The Check for Understanding is also used to preview the next lesson with questions as shown in Lesson 2 (section 4.1.10).

5. RESULTS

Working with Citizen Schools, we have collected data from Fall 2009-Spring 2011, four of our six previous iterations of this apprenticeship. While our intent is to encourage a lifelong student interest and enthusiasm for computing, our short-term goal is to help middle school students improve performance courses as we believe that doing so will provide us with a better basis for encouraging the integration of our curriculum into classrooms as part of the standard school-day.

In the four semesters for which we have data, we have consistently achieved improvement of below-B grades and maintenance of A-B grades among students in both English Language Arts (ELA) and Mathematics (the only courses with data collected through Citizen Schools). To better evaluate our performance, we compare our results to the average results from all Citizen Schools Apprenticeships. This allows us to compare our results to other, similar after school programs. These results do not include our most recent iteration for which our recommended curriculum was implemented. However, our version of the curriculum seeks to address shortcomings from our previous iterations including those that exist in the results presented here and we are confident that it is the strongest version of our work to date.

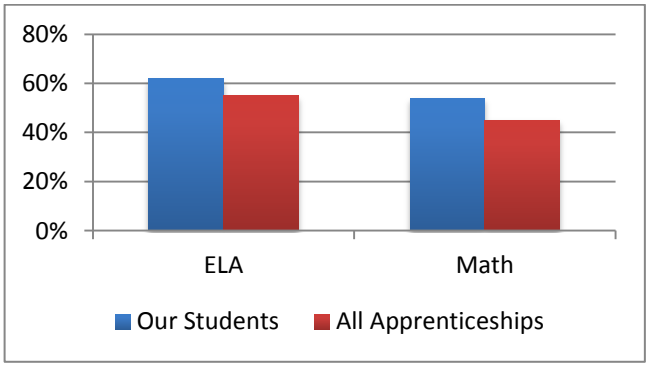

Figure 1: Fall '09 percentage who maintained/improved grade

From Fall 2009 (Figure 1) and Fiscal Year (FY) 2010 (Figure 2), we show high percentages of both grade maintenance and improvement, consistently exceeding the average performance of other apprenticeships. In addition, students in our apprenticeship demonstrated growth in both oral communication and leadership skills (Figure 4). In Spring 2011, 100% of our apprenticeship students maintained their A or B math grades. While no grade improvement is shown for this iteration, it is important to note that only 2 students were making below a B in ELA and only 3 in math, with such a small population as candidates for grade improvement we cannot draw strong conclusions, either positive or negative, regarding this particular measure (Figure 3).

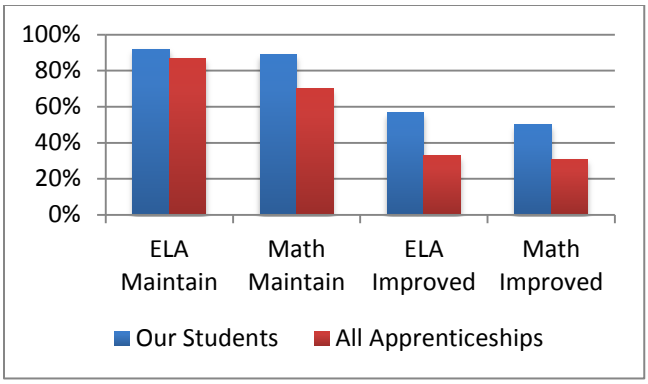

Figure 2: '10 percentage who maintained/improved grade

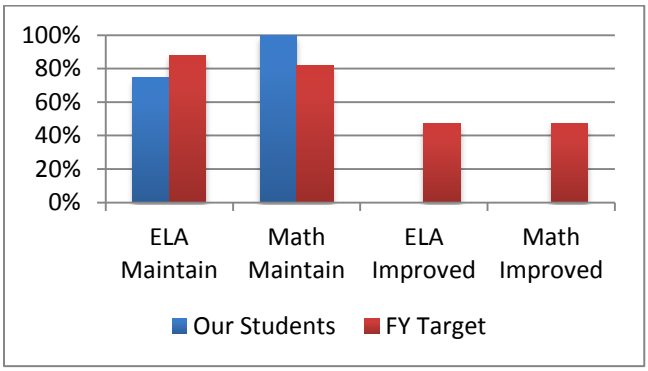

Figure 3: Spring '11 percentage who maintained/improved grade, compared to Citizen Schools FY Target goals

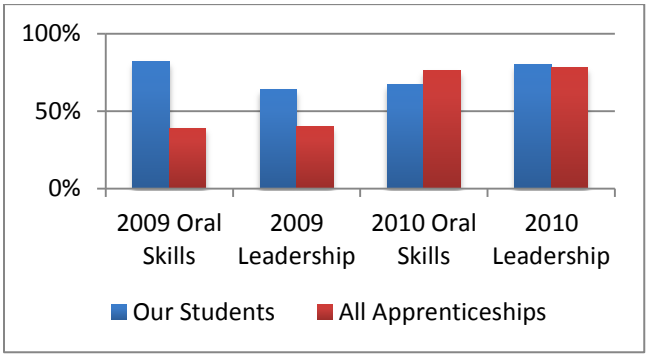

Figure 4: Percentage of students showing growth in Oral and Leadership skills

6. DISCUSSION and CONCLUSION

Having implemented this curriculum in multiple apprenticeships with Citizen Schools for over 80 middles school students, we feel that we have successfully addressed the issues we encountered during our early work and maximized student creativity and technical contribution while also contributing to improved student performance. This belief is strengthened by the increasingly positive feedback we have received each semester, from both our students and the Citizen Schools personnel.

7. FUTURE WORK

We are awaiting an annual report from our local school system that will provide us with a way to track how students who have participated in our apprenticeship perform in high school. Our hope is to see our apprenticeship alumni engaging in higher levels of math and science with strong grades.

8. ACKNOWLEDGMENTS

Our thanks to STARS Alliance for assisting us in our outreach efforts, UNC Charlotte's Chancellor's Diversity Fund for awarding us the funding necessary to provide our own mobile laptop lab to utilize during our outreach sessions, the Citizen Schools organization for their partnership and the work they do on behalf of underprivileged students across the country, Citizen Schools at MLK and Eastway middle schools for their guidance and support during our apprenticeships, and Cassie McIntyre and Michael Kubiak from Citizen Schools. This work was partially supported by NSF grants for the STARS Alliance (1042468, 0739216, 0540523), Culturally Situated Design Tools (0634342), and Game2Learn (0757521).

9. REFERENCES

[1] Barnes, T., Richter, H., Powell, E., Chaffin, A., and Godwin, A. 2007. Game2Learn: building CS1 learning games for retention. SIGCSE Bull. 2, 3 (Jun. 2007), 121-125.Ding, W. and Marchionini, G. 1997. *A Study on Video Browsing Strategies*. Technical Report. UMD at College Park.

[2] Bayliss, J. D., & Strout, S. 2006. Games as a "flavor" of CS1. SIGCSE Bull. 28, 1, 500-504.Tavel, P. 2007. *Modeling and Simulation Design*. AK Peters Ltd., Natick, MA.

[3] Burd, B., Goulden, J., Ladd, B., Rogers, M., and Stewart, K. 2007. Computer games in the classroom, or, how to get perfect attendance, even at 8 AM. SIGCSE Bull. 29, 1 (Mar. 2007), 496-496.

[4] Cliburn, D. C. & Miller, S. Games, stories, or something more traditional: the types of assignments college students prefer. SIGCSE Bull. 40, 1 (Feb. 2008), 128-142.

[5] Clua, E., Feijo, B., Rocca, J., Schwartz, J., das Gravas, M., Perlin, K., Tri, R., & Barnes, T. 2006. Game and interactivity in computer science education. ACM Siggraph 2006 Educators Program (Boston, MA, Jul. 30 – Aug 3, 2006). SIGGRAPH '06. ACM, New York, NY,

[6] Durndell, A., Haag, D., Asenova, D., & Laithwaite, H. 2000. Computer self efficacy and gender: East and West Europe." Women, Work, and Computerization: Charting a Course to the Future, E. Balka and R.K. Smith, Eds. Elsevier Science, 78-85.

[7] Kurvosky, S. 2009. Engaging students through mobile game development. In Proc. 40th ACM Technical Symposium on CSE (Chattanooga, TN, Mar. 4-7, 2009).

[8] Leutenegger, S. & Edginton, J. 2007. A games first approach to teaching introductory programming. SIGCSE Bull. 39, 1. Mar. 2007), 115-118.

[9] Margolis, J., Holme, J.J., Estrella, R., Goode, J., Nao, K., Stumme, S. 2003. The computer science pipeline in urban high schools: access to what? For whom? *IEEE Science and Technology*, 22(3), 12-19.

[10] Parberry, L., Roden, T., & Kazemzadeh, M. B. 2005. Experience with an industry-driven capstone course on game programming: extended abstract. SIGCSE Bull. 37, 1 (Feb, 2005), 91-95.

[11] Repenning, A. and Ioannidou, A. 2008. Broadening participation through scalable game design. SIGCSE Bull. 40, 1 (Feb, 2008), 305-309.

[12] Sweedyk, E., deLact, M., Slattery, M.C., & Kuffner, J. 2005. Computer games and CS education: why and how. SIGCSE Bull. 37, 1 (Feb. 2005), 256-257.

[13] Wallace, S. A., Russell, I., & Markov, Z. 2008.Integrating Games and Machine Learning in the Undergraduate Computer Science Classroom. In GDCSE 2008: Game Development in Computer Science (Celebrity Century, Feb. 28-Mar. 2, 2008).