LEARNING MODULES

GK-12 DISSECT at New Mexico State University

Title: Massachusetts Institute of Technology's App Inventor: Mini Golf Tutorial

Author: Jezreel Bassett and Melody Hagaman

Discipline or Area: Computer Science, App Development

Teacher: Melody Hagaman

School: Centennial High School

Subject of class: Beginning Computer Science

Grade: 9 - 12th

COVERAGE OF COMPUTATIONAL TOPICS

Vocabulary: Variables, Branching, Iteration, Loops, Parameters, Debugging

Android Components: Sprites and Sprite Collision, Clock, Interactive Gestures (Fling, TouchUp, and TouchDown)

OBJECTIVES

This module should inspire students to continue in application design and computer science by allowing them to complete a working fun application while teaching them the basics of computer programming.

EQUIPMENT AND MATERIALS

App Inventor is an open-source web based application designed for beginning programmers to create software applications for the Android operating system. It is maintained by Massachusetts Institute of Technology (MIT).

Required Equipment for Each Student: a computer with Internet access, a personal Google Gmail account (to create an App Inventor account).

Recommended Equipment for Shared in Classroom: Access to an Android device (This is not required due to App Inventor's built in emulator but highly suggested due to the emulators many flaws and limited features.)

Required Materials: Attached at the end of this document are the debugging modifications: Tutorial link: http://appinventor.mit.edu/explore/ai2/minigolf.html

BACKGROUND AND REFERENCES

This is an introductory application utilized so students can explore and learn to use the software, App Inventor. The students should already be familiar with the CT vocabulary terms and this exercise will reaffirm their understanding. The debugging section of this exercise is designed to challenge the students' understanding and prevent blind copying of the tutorial.

PROCEDURE

This module can be completed individually or in small groups; our students worked in pairs, the same pairs all semester (they created a joint Gmail account to share work), both students require computers to complete their individual tasks. The pair consists of a driver and a navigator, the driver is the student actively coding whereas the navigator is the student reading through the tutorial and directing the driver. The students are instructed to switch roles periodically, this is done to reinforce teamwork and insure both students are paying attention and actively assisting each other.

In preparation for this module the instructor should complete tutorial and have a working application. Doing this will familiarize the instructor with App Inventor enough to present the layout to the class and answer questions. Also this is required to produce debugging problems, our class had three different debugging problems to solve. A debugging problem is a copy of the completed code saved under a new file (Ex. Error1) that has been altered to create an operation problem in the game but not to create a warning or an error. This is designed to test the students' understanding of the code and introduce them to real world debugging.

First the students are instructed to create a joint Gmail account for the pair (this will only have to be completed once of this pair for the entire time they work together), using that account they created an App Inventor account.

The instructor then gives a quick run-through of App Inventor's layout and how it relates to the tutorial. Remind the students of the relationship between vocabulary and the tutorial components.

The students are then assigned their roles as driver and navigator and finally assigned the tutorial. The instructor is then free to answer questions as they arise. This section of the module took our class two full class periods (the majority of the student completed it during this time). Allow them to load the code to device and play around; after all it is a game. The better they know the games operation the easier they will find the error during debugging. All students are encouraged to finish early and modify their code for a higher score.

Debugging is the final stage of this module; assign the three debugging files to the students. They will need to be individually downloaded to App Inventor as well as the Android device, and then rigorously tested to determine what the errors are. Instruct them to observe and write down not only what components are malfunctioning but also what is the trigger to the malfunction; these will both be clues to finding the error. When they find

the error they are to write down or demonstrate to the instructor what was wrong and how they fixed it. This took another full class period.

What were the "learning goals?"

Become familiar with App Inventor, complete a working Android App, reinforce computer science vocabulary and methods, and introduce debugging.

How did you introduce CT?

Introducing App Inventor software and developing an Android App are a great ways to introduce computer technology to students for it provides programming experience with a functional end result.

How could you assess the understanding of CT in this module?

To assess the students understanding of the software App Inventor the instructor will collect and run the completed app for average score, and a completed improvement for an excellent score. An incomplete App is a poor score; this is a problem for they would not be able to perform the debugging section (All of our students completed the tutorial, for they can access App Inventor from anywhere and work on it in outside of the classroom). The debugging could be assessed for correctness, as there are three variations there are four possible scores; completed all, completed two, completed one and none completed.

NOTES AND OBSERVATIONS

What were challenges you encountered in the overall development of the module?

The biggest challenges encountered were that of group work and lack of problem solving skills. Navigators were often not participating, resulting in confusion when it was time to switch. If students came to a problem or didn't understand the tutorial they would either stop and ask for help or become distracted.

To solve both problems, the instructor had to be active with the class, walking around monitoring progress. The debugging section was developed to build understanding in hopes to combat the stall of progress in light of questions.

What was successful?

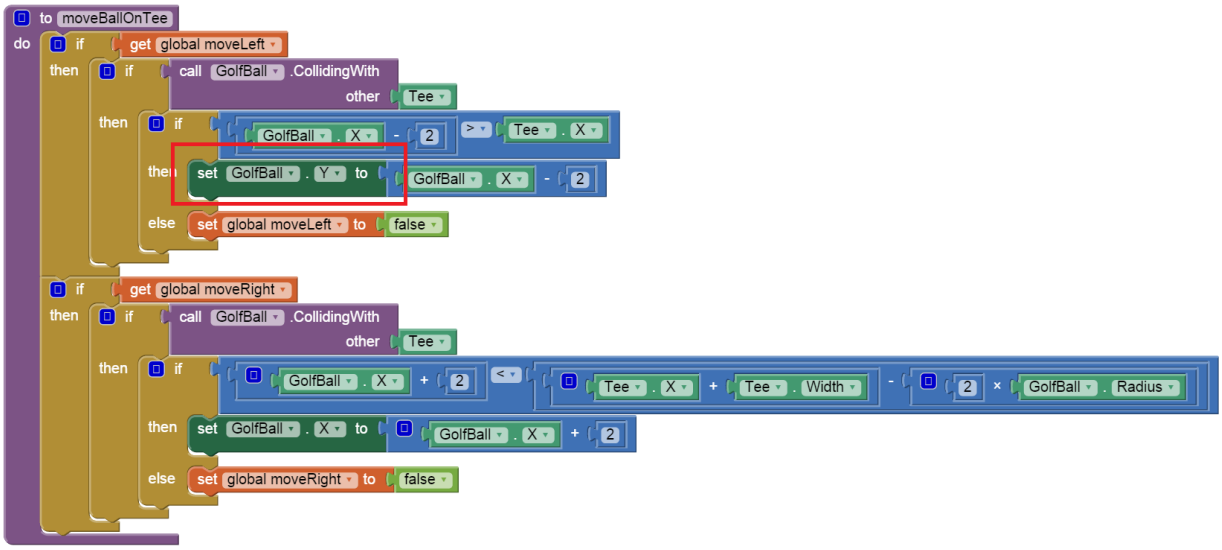
The module was highly successful, all students completed the tutorial and most attempted the modifications. A few completed all the debugging problems, most completed two and everyone completed at least one of them.

How was the students' reception to the content of the module?

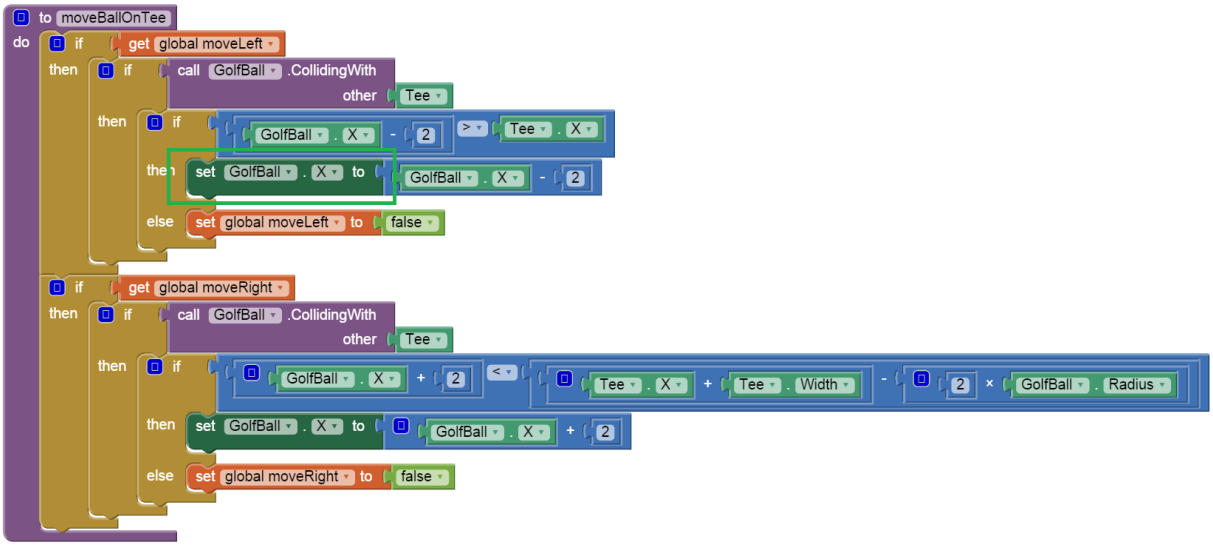
The students enjoyed developing the game and making it their own. The debugging section was so interesting they seemed to enjoy it the most, treating it like a puzzle. The instructor plans to implement it in all upcoming modules.

Debugging Examples:

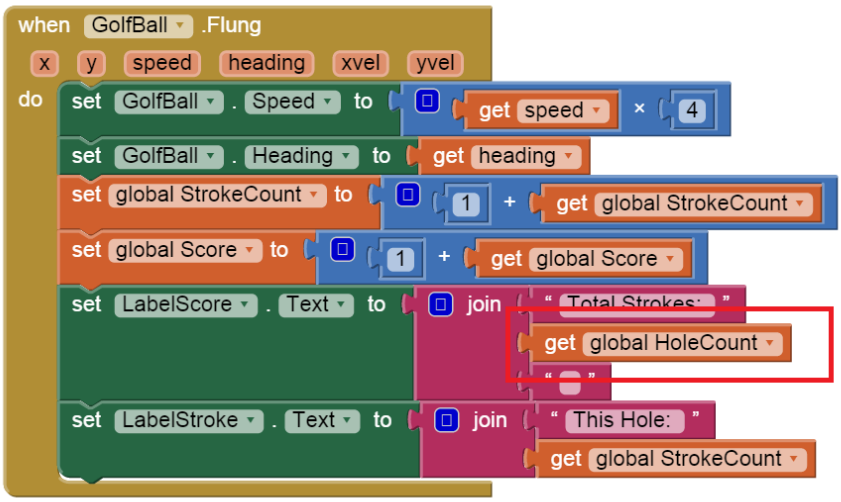
Problem #1 (The error is in the red box)



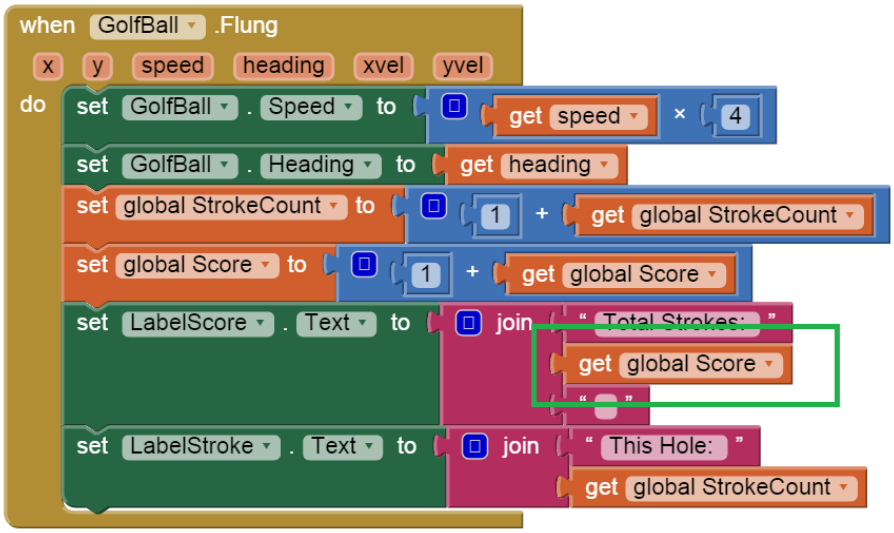
Solution #1 (The solution is in the green box)



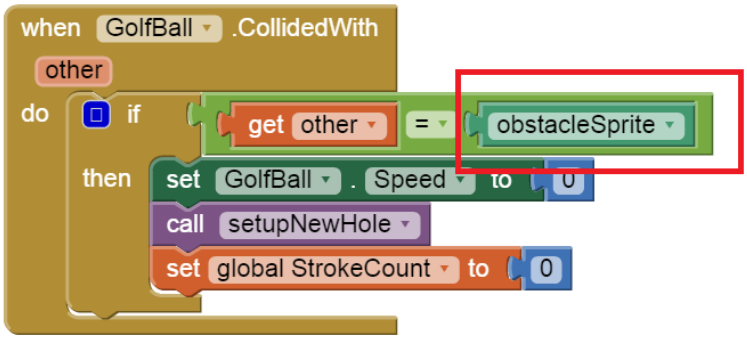
Problem #2 (The error is in the red box)



Solution #2 (The solution is in the green box)



Problem #3 (The error is in the red box)



Solution #3 (The solution is in the green box)