# Exploring the use of Android OS in CS2

Mark Goadrich and Jacob Jennings
Department of Mathematics and Computer Science
Centenary College of Louisiana
Shreveport, Louisiana 71104
Email: {mgoadric,jjenning}@centenary.edu

Matthew Jadud
Department of Computer Science
Allegheny College
Meadville, PA 16335
Email: matthew.c@jadud.com

***Abstract*—Smartphones are increasingly being used in classrooms, with a focus on either upper-level software engineering and capstone projects with iOS or Android OS, or a lower-level introduction with AppInventor. Our work Android4CS2 provides tools for introducing Android in the middle ground of CS2 courses on data structures and algorithms. Android's basis in Java enables us to leverage the Model-View-Controller pattern as part of a motivating course paradigm of mobile development without disrupting standard course topics. We believe that structuring a CS2 course within the Android framework will motivate students to not only gain a solid understanding of data structures, but also to appreciate sound software engineering principles.**

## I. Introduction

Our goal is to explore the use of Android-based mobile devices for motivating student learning regarding data structures and algorithms—material traditionally referred to as CS2 in the computer science education community. Specifically, we have developed a series of programming projects for use on Android-based devices that allow students to focus on developing core data structures first and then integrating the details of the interface and user interaction.

Our use of Android in the classroom aims to strike a balance between approaches that might be used when teaching novice programmers such as AppInventor [1] or students that are in an upper-division, special-topics course on ubiquitous computing [2]. We are interested in reaching students who have introductory knowledge of a programming language and are on the verge of tackling large software engineering projects. Our goal is to provide these students with a "loosely sandboxed" environment where they can focus on material core to the study of data structures and algorithms while simultaneously creating opportunities for creative exploration above and beyond the material presented in any given laboratory or homework assignment.

### A. Leveraging Games

At our respective institutions, we currently make use of Drake's *Data Structures and Algorithms in Java* [3]. While this text is fairly traditional in its structure, covering stacks, queues, linked structures, trees, hash tables, etc, Drake motivates all of his programming exercises through simple board and card games. Chapter one introduces students to an abstraction for a Die, and proceeds to implement Beetle, a game where players race to assemble a six-legged insect. Games like Reversi motivate the use of matrices, while various card games such as War and Go Fish demonstrate the use of stacks, queues, and linked lists.

We have found that students react well to this text, not necessarily because they are programming games [4], but moreso because they understand the problems clearly. Students can use dice, cards and playing pieces to trace their code; having an intuitive understanding of how the program should work makes it much easier for them to design their programs. The games also provide a social context for discussing programming, where it is very rewarding for students to play the result of a programming assignment with a partner.

### B. Leveraging Mobility

Mobile devices provide an additional benefit: students can easily share their work. Often, students of computing are hard pressed to share their efforts with classmates and family. Now, instead of asking, "What do I have to do to get this to run on my own phone," students quickly shift to wondering "How do I make it so I can shake my phone to roll the die?" While this may be a challenging question before the topic of interfaces has even been broached, we believe devices that students use and relate to on a daily basis have the potential to be incredibly motivating in an intrinsic sense [5].

An added benefit of Android development is the clear motivation for the exploration of the Eclipse IDE. In many departments, decisions regarding tools (e.g., BlueJ vs. Eclipse vs. command-line), and when to switch between them can be a contentious decision that is difficult to motivate for both faculty and students. The fact that Eclipse is the tool of choice for Android development provides a simple, honest motivation to help students overcome the relatively small frustrations of learning a new tool.

## II. Preparation and Planning

The use of new curricular technology (e.g., robots, phones or Arduinos) requires additional planning and infrastructure on the part of the instructor. First and foremost, a course based on mobile phones must make the phones available to students, and the phones need to be charged and ready for use in-class, during laboratory periods, and outside of class. We know from the literature that courses based on technology with limited accessibility can negatively impact both learning and student's reactions to the course [6].


Fig. 1. Idiot's Delight Android application

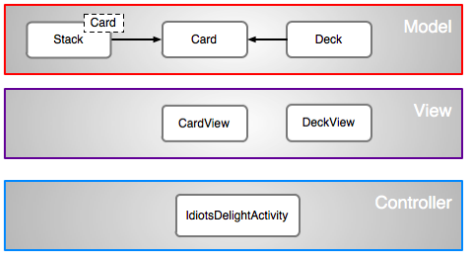


Fig. 2. Decomposition of Idiot's Delight for Android

Having used Drake's text previously, Android with its reliance on the Java programming language was a natural choice for our integration of data structures and algorithms in a mobile context. While the mobile device provides a motivational context and room for advanced students to explore, our courses are fundamentally about data structures and algorithms. For this reason, we set out to develop applications based on Drake's use of games that clearly separated the study and implementation of structure from the mechanics of implementing software on the Android platform.

Planning began during the summer of 2010, and in fall 2010 the second author undertook the implementation of 10 "wrappers" for examples from Drake's text, one of which is detailed in Section III. Each of these wrappers are translations of Drake's games into the Model-View-Controller paradigm, where the interface and user interaction elements (the View and Controller) are separated out from the underlying data structures (the Model). We intentionally leave out the Model portion of the application, as this is the code we want our students to write as part of their assignments. We have licensed these wrappers under the GPL and made them available at `http://android4cs2.googlecode.com/`.

Too often, motivating technologies become an afterthought that students begin to see as a burden. While these wrappers are usable outside of the context of Drake's text, we did not need to make any large curricular changes to our course to accommodate the use of our Motorola Droids. In our courses, students can continue to use their text as a core resource while working on developing structures to support the execution of programs on a mobile device.

## III. Exploring Stacks: Idiot's Delight

A sample wrapper that illustrates our approach is Idiot's Delight, a solitaire card game. A screenshot of our application can be seen in Figure 1. From the student's point of view, this game has three fundamental abstractions: the `Card`, the `Deck` of undealt cards, and four `Stacks`. These abstractions form the core of the Model for this application.

In the text, students are introduced to the `Stack` interface (`push()`, `pop()`, `peek()`, and `isEmpty()`), and are asked to provide a stack implementation that leverages arrays in the form of a class called `ArrayStack`, or one with nodes called `ListStack`.

Our Android implementation cleanly separates the MVC pattern into the class `IdiotsDelightActivity` (the Controller) and two classes implementing Views ( `CardView` and `DeckView`), shown in Figure 2. This allows students to concentrate more precisely on the Model portion of the code within our course, while instructors can then assist the students as they integrate their data structures into the larger application. This also makes the motivation for discussing interfaces and layers of abstraction clearly relevant, as they are able to see their code working seamlessly within these wrappers with only the interface to guide the interactions between the Model, View, and Controller.

After gaining familiarity with the necessary data structures, interested students can revisit each assignment with a focus on understanding the Android wrapper itself. An instructor can highlight how Android projects are built using the same abstractions of interfaces and inheritance the students are learning in class.

## IV. Conclusions and Future Work

Android4CS2 is currently being piloted in Spring 2011, where we will be evaluating our initial explorations in this space through combination of student journals and surveys, along with outcome assessments based on student retention and course evaluations. We believe this combination of mobile device and games will facilitate a deeper understanding in the students of the CS2 course topics. While the pace of the course has been slower than previous years due to the additional Android content, we have already experienced a number of benefits from this approach, including our expected increase in student motivation and exploration.

While our above applications make use of Drake's text, we are currently implementing separate assignments that can highlight the intrinsic sensors of mobile phones such as the GPS, accelerometer and compass. We plan to augment these with exercises from other textbooks as well as assignment repositories such as the SIGCSE Nifty Assignments website at http://nifty.stanford.edu/. In addition, we are exploring the use of these applications beyond the CS2 course as a tutorial on Android programming, where students would implement and alter the Views and Controller classes for this familiar set of Models.

## ACKNOWLEDGMENT

The authors would like to thank the anonymous reviewers for their helpful comments, their current students enrolled in the pilot courses, and Google for the support of Motorola Droids for the classroom through an Android Education Grant.

## REFERENCES

[1] D. Wolber, "App inventor and real-world motivation," in *Proceedings of the 42nd ACM technical symposium on Computer science education*, ser. SIGCSE '11. New York, NY, USA: ACM, 2011, pp. 601–606. [Online]. Available: http://doi.acm.org/10.1145/1953163.1953329

[2] J. B. Fenwick, Jr., B. L. Kurtz, and J. Hollingsworth, "Teaching mobile computing and developing software to support computer science education," in *Proceedings of the 42nd ACM technical symposium on Computer science education*, ser. SIGCSE '11. New York, NY, USA: ACM, 2011, pp. 589–594. [Online]. Available: http://doi.acm.org/10.1145/1953163.1953327

[3] P. Drake, *Data Structures and Algorithms in Java*. Pearson Education, 2006.

[4] N. Whitton, "Motivation and computer game based learning," in *Proceedings of ASCIILITE*, 2007.

[5] M. Apiola, M. Lattu, and T. A. Pasanen, "Creativity and intrinsic motivation in computer science education: experimenting with robots," in *Proceedings of the fifteenth annual conference on Innovation and technology in computer science education*, ser. ITiCSE '10. New York, NY, USA: ACM, 2010, pp. 199–203. [Online]. Available: http://doi.acm.org/10.1145/1822090.1822147

[6] B. Fagin and L. Merkle, "Measuring the effectiveness of robots in teaching computer science," in *Proceedings of the 34th SIGCSE technical symposium on Computer science education*, ser. SIGCSE '03. New York, NY, USA: ACM, 2003, pp. 307–311. [Online]. Available: http://doi.acm.org/10.1145/611892.611994