Explore – Create – Present: A Project Series for CS0

David B. Adams
Grove City College
100 Campus Drive
Grove City, PA 16127
724-458-2157
dbadams@gcc.edu

ABSTRACT
This paper describes a four project sequence used in our Introduction to Computer Science course focused on allowing the students extensive creative freedoms within a highly motivational administrative framework. Each project reinforces foundational CS concepts like branching, looping, sequence, variables and the notions of objects in a very concrete and visual way with instant feedback to the students. The key element though is that the four projects all use completely different interfaces and environments allowing students to see the same basic ideas from several perspectives in order to reinforce a more correct mental model of the concepts. This method allows for a more natural step toward abstraction of the fundamental ideas and should help students be more successful in later CS courses.

Categories and Subject Descriptors
K.3.2 **[Computer and Information Science Education]**: Computer Science Education.

General Terms
Design

Keywords
CS0, Alice, LEGO, Scratch, AutoHotkey

1. INTRODUCTION

The CS0 course at Grove City College has been delivered once per year since 2006 and is listed as a required course for incoming Computer Science (CS) freshman in the fall semester though non-majors are also allowed to take the course. The course was introduced to provide our students with an overview of CS and to help prepare them for later courses in the CS major program. An additional goal was to make our students and our major more visible on campus through the use of playful and nonthreatening projects to help undecided majors learn about our CS program. The class has evolved into a gentle introduction to CS with a reputation for being fast, fun and interesting, a consequence of covering such a wide range of material leaving most of the details to later courses.

The primary focus of the paper though is in the details of our series of projects used in the course. The idea is to have a series of projects that use an environment that is easy for the students to pick up without much instruction from the course coordinator. Teams of students explore the environment, discover what they need to know in order to complete the assignment, create a project that complies with their desired level of performance and then present their project for the class. The idea is that students pick up required skills through the exploration and discovery process out of necessity in order to satisfy the project requirements. Each project uses a different environment so only the basic CS concepts are transferrable from one project to the next. Students understandably do not become experts in each of the environments but the desire is for them to draw from their project experiences a consistent and correct model of fundamental and core CS ideas as a natural result of discovering the same concept multiple times, each time from a slightly different perspective.

The next section of the paper explains the basic course outline, a "breadth-first" look at CS. Section 3 details the four project sequence using Scratch, Alice, LEGO NXT Educational Kits, and AutoHotkey. Section 4 describes some of the perceived impacts of CS0 on our program and the paper concludes with proposals for future work in the class.

2. BASIC COURSE FORMAT

The basic course structure follows a "breadth first" approach to computer science and is a required course for computer science majors at our institution. Using a bottom up approach we survey different layers of abstraction found throughout a computing system closely following the ordering of topics from our chosen textbook, Computer Science Illuminated [1] by Nell Dale and John Lewis. The basic outline of topics for the course is as follows:

Binary Values and Data Representation
Gates, Circuits and Hardware Components
Basic Problem Solving
Low-Level Programming Languages
High-Level Programming Languages
Introduction to Abstract Data Types and Algorithms
Operating Systems
Spreadsheets, Databases, AI, Simulation, and Graphics
Networks and the World Wide Web

The class also includes more or less weekly class discussions of ethical issues, some taken directly from the book and others garnered from recent happenings in the news. The sometimes heated discussions provide the class with a good opportunity for interaction and are the basis for further exploration outside of class. In addition to normal assessment of the topics we cover, the ethical issues are assessed in essay form on each of the three exams given throughout the term. Students are made aware that in order to perform well on ethical essay questions they must not only formulate a good argument based on the details and facts presented on one of the many ethical issues covered during class discussion but also include details from similar or related issues not covered in class. The use of persuasive essays on ethics in CS0 is not uncommon [2] and, for some schools, actually fulfills a more general writing requirement for their program though that is not the case for our course.

Outside of standard lecture and ethical discussions, classroom time is used for project introductions and demonstrations. We use teams of three to five students for our project sequence, rotating the teams each iteration so that students can work with a different set of people for each project. Project introductions are handled by the course coordinator and usually only take about 25 minutes of lecture time to demonstrate the new tool the students will be exploring for the next few weeks. We do not really teach them to use the tool, students pick up required skills through exploration and discovery out of necessity in order to satisfy the project requirements. Instead, we try to demonstrate how to explore the environment and show where to go to learn more through self study. Each of the projects, save for, perhaps arguably, the last, has a very gentle learning curve and rewards exploration with instant feedback and allows the students to apply some of the high level ideas in lecture to a specific tool. Student demonstrations and presentations then occupy one lecture period for each of the four projects spaced throughout the term. The course is set up as a three hour course, typically meeting on a MWF fifty minute schedule and does not have an additional lab requirement.

3. FOUR PROJECT SEQUENCE

3.1 Scratch

The first project, given on day two of the semester with a due date about three weeks away, is the Scratch project. Scratch is a programming language and environment developed at MIT [3] that allows users to create interactive animations, games, music and art using a visual drag and drop, snap together, block interface. It is appropriate for use with children as young as eight years old but is still interesting and engaging for Intro to CS students at the college level. Scratch has a very intuitive interface and a gentle learning curve that makes it ideal for a project that assumes nothing about the student‟s background in programming.

Scratch is designed to help users encounter a realistic design loop that includes construction of creative ideas, experimentation with the implementation of those ideas, sharing those ideas for review by peers and then incorporating comments and suggestions for improvement in a new creative idea or extension to the current project. While sharing is an important part of the Scratch design and is made easy with Web publishing features built right into the environment, our class goal is targeted primarily in the exploration and discovery of tools to accomplish a groups design goals. The review process then is contained within a particular group working on the project and sharing is limited to the class and visiting faculty on project due dates. Scratch makes it easy for students to learn about control structures like selection, looping and sequence through the use of visual programming blocks and constructs that can be pulled into the working area and immediately executed or experimented with by double clicking on the blocks. Blocks snap together to form larger sequences and provide immediate feedback to the students as they slowly begin to expand their comfort range around constructs found to be useful or interesting. Students can engage in simple projects in the first few moments after being introduced to the interface and appear to be eager to share their work as it has an immediate “cool” factor, presumably through the easy access to motion graphics and sound, that causes them to share their ideas with dorm mates and friends not even in the class. This behavior has been observed many times through the comments students make at the beginning of lecture and seems to be positive reinforcement to help spread the word that computer science is fun and interesting.

The project specification is designed to allow the students the maximum amount of freedom while still giving them a good sense of how much effort is involved in getting a good grade. A tiered specification sheet is distributed to the students to indicate what must be accomplished in order to get a “D” grade. Though the specification changes from year to year, performance at the “D” range generally may include construction of a short scene in their mini-world demonstrating at least a certain number of scripted behaviors, control structures, and actors on the scene (sprites in this case). To perform at the “C” grade level students must accomplish all of the requirements at previous levels and satisfy an additional series of requirements. These may include things like demonstrating abstraction through the use of combining basic blocks into a larger construct that performs a higher level action that can be used again later. An example might be building a series of blocks that cause a sequence of sprites to appear to jump, walk, run, or perform some action like eating. Similarly each tier asks the students to explore more features of the environment and to discover more depth about the topics we are beginning to cover in class as well as topics they will see again in lecture after the project is due. In order to perform at the A+ range we always list that the students must impress the instructor and the audience.

In practice students could aim at the "D" level with minimal effort and much of the projects learning potential would be lost but in practice, since the projects are all group projects, students are driven by peer pressure to push harder. Using a classroom technique borrowed from the Building Virtual Worlds course offered at Carnegie Mellon University, during the demonstration phase of a piece of technology the instructor for the course will show an example of the best work submitted so far from past offerings of the course and hail it as work that the average group will output at completion of the project. This sets the bar high for incoming groups and by showing what is possible causes them to work hard to create an impressive final product. Although it seems unrealistic for the trend to continue for many years to come, thus far, the students always seem to trump past performance on the projects and continue to raise the bar for future classes.

3.2 ALICE

The second project is given using Alice 2.0. Although Alice 3.0 is nearly available [4] it is still not known to the authors if that platform can fulfill the same role as the previous offering. Alice takes the students into a world of 3D models and has an environment that is a bit less forgiving than Scratch. The drag and drop interface though is similar and the learning curve is gentle enough that the students can easily tinker with new parts of the interface to discover new techniques they can incorporate into their project. Although there have been numerous publications and studies on using Alice as a first programming language the goal for this course is just to encounter a new tool with familiar programming elements and to build something fun. Here again, it is not our goal to have the students become expert programmers in the Alice environment but rather to have them discover in an explorative way the similarities between programming with Scratch and programming with Alice. This process of comparing environments and having students encounter the same ideas in different guises is the foundation on which the four project sequence is built upon. Ideally students will begin to build abstract models for themselves that help them classify the abilities they are given in each of the different environments. This allows them to see the similarities in sequencing, flow of control, and the flavors of object oriented programming that present themselves in different ways for each project. More data collection is underway to try to support the validity of this conjecture.

Like the Scratch project the method for the Alice project is much of the same procedure. Students are given a very loose specification detailing what tools they need to make use of in order to perform at a particular grade level and each level adds more tools and more depth to the experience. The students are only given a very simple introduction to the environment and are expected to discover everything else that they require in order to succeed. The project length is again about three weeks and the same technique for showing off the strongest project from previous terms and claiming it is average work is used to set the expectations high for both what they should be able to accomplish and what they imagine is possible.

The Alice environment tends to be a bit more frustrating for the students either because of errors they find with the software environment or simply the slight increased difficulty in discovering things in the environment with very little instruction. Projects, however, are generally of high quality and give the students the opportunity to talk about design trade-offs that they had to make during development because of particular limitations they encountered in the software environment as a tool.

3.3 LEGO NXT EDUCATION KITS

The third project makes use of LEGO Mindstorm NXT Education Kits. The education kits are reasonably priced collections of parts that can be given to each team for the duration of the assignment. LEGO Mindstorms are gaining in popularity in various courses including CS0 [3] and CS1[5] and are a nice addition to the project sequence for our CS0 course. The primary concern is not in building interesting robots from available pieces but rather using a robot and building software to allow the robot to accomplish some task with an associated story. Rather than dictate even the task that the students are to perform, the project specification provides enough freedom for the students to come up with their own task and then demonstrate that the robot, of their own design or the default Tri-bot configuration, can in fact accomplish the task in a fairly autonomous way.

Grading for this particular project is a bit more difficult because although it has the same structure of techniques and grade tiers with regard to the specification and the meager demonstration of the environment, there is typically less of a distinction for division of labor for the groups. This forces the groups to work more closely together on this project since, although they each have the design software on their own machines, real testing requires that they share use of the robot.

The LEGO Mindstorm robots provide a very interesting and observable project for the students. Observable in the sense that students outside of the program can often observe them playing with the kits in public areas and become interested in what they are doing. There is a conscious effort to provide visible projects that have some sort of informal "show and tell" appeal giving the students an opportunity to show off some of the things they are learning. For the past two years we have asked select student groups to demonstrate their projects for visiting prospective student groups and have made digital recordings of other project presentations available for the same purpose.

Throughout the LEGO project the students are again encountering the same basic building blocks of programming in yet another format. At this point in the term the students will also have done some simple machine level

and assembly programming as well as explored some details about procedural and object oriented design strategies for problem solving. Fun examples of past work include a sentry robot that measures the area around the robot in a 360 degree sweep and after recording the dimensions fires a rubber band at any would be trespassers that are detected within the perimeter. An impressive example included a robot that could explore a floor map terrain in search of objects, finding three objects of different sizes. Once the robot found an object it would determine if it was the big object, the medium object, or the small object using various creative uses of the limited sensors and then sort the objects on the side of the map according to their size. The interesting thing about this project was that the objects and the robot could all be placed randomly on the map without interfering too much with the high chance of success.

The robots provide a concrete way for the students to discover some of the interesting problems in autonomous robot control programs and allow them to get a glimpse into a possible research track they could explore with a degree in computer science. The only real complaint students typically have with the LEGO systems is in the limitation of parts for their projects. So, while we feel the kits are accomplishing the goal for which they were purchased, it would be nice to slowly add bins of parts that the groups could draw upon to enhance their creations while still maintaining the same basic kits for each group. The last offering of the course augmented the basic kits with an extra bin of parts for each group and allowed for increased complexity for the students that wanted to make use of the additional resources. Issues of part loss and breakage are minor at this point as the students only have the kits for about three weeks allowing us to replace parts fairly easily from spare kits.

3.4 AUTOHOTKEY

The final project uses an open source utility for Windows called AutoHotkey. AutoHotkey allows users to quickly automate tasks in Windows sending keystrokes and mouse clicks automatically to the environment. Although the use of AutoHotkey in the classroom appears to be novel we do not feel that it is the only choice for this project as the goal of the project, as before, is simply to reinforce ideas and exploration in a new environment the students have likely not encountered before. The primary reason AutoHotkey was selected was because it allows the students to learn a new system, applying the fundamental programming concepts they have been using all semester long in an abstract way that more closely resembles the abstractions they will encounter in future programming courses. The advantage AutoHotkey has over just providing them with a full featured language for one project is that it has an extensive set of tutorials that come with it allowing the students, with only a 30 minute demonstration, to use the exploration and discovery techniques they have applied to the last three projects again in working with a new environment. Students can immediately begin automating simple tasks and gathering building blocks that they can then apply to their final project demonstration.

The final project demonstration is a bit more defined in some ways because all students are required to automate the play of a simple game. The particular game selected for the class is a bizarre kind of online multiplayer game that is free to play where the players move around in a tiled world, planting and harvesting things to acquire materials that can then be used for various building or advancement purposes. The game, called Toadwater, is not very fun to play and requires an enormous amount of repetitive behaviors in order to be successful. Add to this the fact that the game is played slowly with built in delays in almost every action make it the perfect target for creating a program that plays the game for you where these programs are commonly referred to as “bots”. Although there is a captcha system in place in the game to help prevent excessive automation, it does not interfere too much with the student projects. Instead, captchas provide an interesting unknown that can appear at any moment forcing the scripts to abort the current operation and instead notify a nearby human that help is needed. Here again, the choice of game is somewhat arbitrary but it is important to use something that is easily automated and has fairly simple rules as writing bots for complex games can be extremely challenging.

The freedom for the project comes in what actions the students choose to automate in the game and the complexities involved in the automation. Interesting projects automate many actions and are seamlessly autonomous save for the interruption of a captcha block. It would be nice to have a set of games that would all work well for this project so that the students could choose something that interested them more but we have found that allowing the students to choose without restrictions the game they want to automate typically results in them being defeated by better engineered game play that is not automation friendly.

The AutoHotkey project is a nice final project that gives the students an interesting tool to continue to play with in later courses. At the end of the most recent offering an informal survey was taken to determine how many of the students had used AutoHotkey after the project to automate something for themselves and about 50% (15 of 31) indicated in the affirmative. This is considered a great victory in that the students continue to play with the tool and explore even after the threat of grading has passed. A more formal look at this result will be interesting in future work.

4. IMPACTS OF CS0 ON OUR PROGRAM

The CS0 course is still relatively new at our institution. The first course was taught in the Fall of 2006 and used Alice for all of the projects in the term. It seemed apparent that advanced skills in Alice were less transferrable to other programming tasks than the simple skills used in the first few projects. This planted the seed for an idea to use multiple types of projects in the course instead where the transferrable skills from each system were the same

abstractions we were trying to teach them about general programming. The project sequence evolved first adding LEGO mindstorm robots and then the Scratch program. Finally, in the last two iterations of the course the AutoHotkey project was added. We are fairly happy with the current incarnation of the course and wanted to share some of our experiences with the rest of the community to perhaps see more formal studies on this observed transfer of skills from one programming system to the next.

We do not have enough data to speculate about how well our CS0 course prepares our students for CS1 but the general feeling is that the students are better prepared and better informed about what they can expect to encounter in the computer science program.

5. FUTURE WORK

Future work will be primarily focused on refining the CS0 course offering and trying to keep the project sequence as friendly as possible to explorative, trial and error type learning. As new options become available, like Alice 3, they will be considered and adopted as appropriate. Four projects seems to be the maximum that can fit in a 15 week term for these types of show and tell type presentations and so with each new offering most likely something will have to be replaced. It is interesting the way the current project sequence goes in that the students see a direct link between the Mindstorm robots and the 'bot' they build to play a game using AutoHotkey. There are very similar issues encountered in both projects involving the automation of the robot and the in game character or avatar. In addition to course refinement, additional effort is being made to support the following:

Goals

1. Multiple environments in a single CS0 term are not detrimental but rather helpful in providing a firm conceptual model of programming if the learning curve is gentle enough.
2. This particular model can improve retention, success rates, and positively affect recruitment in the CS program.
3. Knowledge transfer occurs between projects and groups demonstrated by the depth of exploration in the final project.

6. REFERENCES

[1] Dale, N. and Lewis, J., Computer Science Illuminated. Sudbury, MA: Jones and Bartlett, 2007.

[2] Cliburn, D. C. 2006. A CS0 course for the liberal arts. In Proceedings of the 37th SIGCSE Technical Symposium on Computer Science Education (Houston, Texas, USA, March 03 - 05, 2006). SIGCSE '06. ACM, New York, NY, 77-81.

[3] Peppler, K. Kafai, Y. B. 2007. Collaboration, Computation, and Creativity: Media Arts Practices in Urban Youth Culture. In C. Hmelo-Silver & A. O'Donnell (Eds.), Proceedings of the Conference on Computer Supported Collaborative Learning, New Brunswick, NJ.

[4] Dann, W. and Cooper, S. 2009. Education Alice 3: Concrete to Abstract. Commun. ACM 52, 8 (Aug. 2009), 27-29.

[5] Eggert, D. W. 2009. Using the LEGO Mindstorms NXT robot kit in an introduction to C programming class. J. Comput. Small Coll. 24, 6 (Jun. 2009), 8-10.