A Framework for Enhancing the Social Good in Computing Education: A Values Approach

Michael Goldweber
Xavier University
Cincinnati, Ohio, USA
mikeyg@cs.xu.edu

John Barr
Ithaca College
Ithaca, New York, USA
barr@ithaca.edu

Tony Clear
AUT University
Auckland, New Zealand
tony.clear@aut.ac.nz

Renzo Davoli
Università di Bologna
Bologna Italy
renzo@cs.unibo.it

Samuel Mann
Otago Polytechnic
Dunedin, New Zealand
Samuel.Mann@op.ac.nz

Elizabeth Patitsas
University of Toronto
Toronto, Canada
patitsas@cs.toronto.edu

Scott Portnoff
Downtown Magnets High School
Los Angeles, California, USA
srport@alum.mit.edu

ABSTRACT

This paper addresses two interrelated problems currently confronting computer science education, motivating students while simultaneously providing them with the skills they'll need to solve complex interconnected problems. We describe a framework for motivating computer science students by adding the context of social good to introductory computing assignments. Adding the context in this manner also goes some way to addressing the need for graduates to have skills, attributes and behaviours appropriate to contributing to social good outcomes. Accompanying this, we provide 14 concrete examples of introductory computing projects that convey and reinforce computing's social relevance and potential for positive societal impact.

Categories and Subject Descriptors

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

General Terms

Design, Experimentation

Keywords

Introductory Programming Projects, Computing for the Social Good

1. INTRODUCTION

This paper addresses two interrelated problems. It describes a framework for motivating computer science (CS) students by adding the context of social good to introductory computing assignments. Adding the context in this manner also goes some way to addressing the need for graduates to have skills, attributes and behaviours appropriate to contributing to social good outcomes such as those described by Paterson[66].

There are many factors that influence students' perception of computing. Some of the misconceptions are due to a lack of correct knowledge of what computing practitioners actually do coupled with the reinforcing by the popular media of longstanding myths and stereotypes. One often cited study of STEM oriented high school students describes students' perception of the computing discipline(s) as boring, tedious and irrelevant[40, 85].

Some of these factors are outside the control of CS educators, while others are not. To what degree does a given introductory curriculum[1] either reinforce student held myths and misconceptions or work to dismantle them? Buckley, in a 2009 CACM Viewpoint column[12], complained that introductory computing students, based on the unscientific examination of the textbooks in his office, are seemingly obsessed with animals (e.g. counting ducks, separating cows from horses), games (e.g. Tetris, Checkers), and food (e.g. donut counting, lemonade stands). He concludes with empathy for the student exposed to such motivating examples who quits CS and goes on to study something important.

CS educational activities for the social good (CSG-Ed), a refinement of the acronym CSG[38], is an umbrella term meant to incorporate any *educational* activity, from small to

[1]We utilize the term *introductory curriculum* or *introductory courses* to represent those courses which first expose a student to computer science and lay the foundation for continuing study in the discipline. These courses, sometimes referred to as CS0, CS1, CS2, AP-CS, Pre-AP CS, etc., may occur at the University level or pre-University level (e.g. high school, or college).

large, that endeavors to convey and reinforce computing's social relevance and potential for positive societal impact. Besides the obvious benefit to society, CSG-Ed endeavors to exploit the finding that students' desire to have a positive societal impact is a strong determinant regarding their selection of a major[14]. A side effect of incorporating CSG-Ed activities, particularly in the introductory curriculum, is that it could potentially broaden participation in computing. It is worth noting that this "positive societal impact" is considered an inclusive term: CSG-Ed therefore includes sustainability [57], ICT4D [42], ICT4Peace [44], HFOSS [84], value sensitive design [61] and so on.

1.1 Motivation

It is reasonable to hypothesize that incorporating CSG-Ed activities into the introductory curriculum will address motivational issues on many levels. Students wanting to work in a field in which they can make a purposeful or meaningful social contribution have little difficulty seeing such connections for disciplines like political science, art, education, nursing, archaeology, or STEM fields such as biology, environmental science, or civil engineering. Currently, no such connections exist for CS.

Furthermore, there is evidence that students' inability to see CS's potential social contributions impacts female enrollments. Although girls enroll in secondary math and science classes at or near gender parity and perform as well as or better than boys[45], many avoid taking computer science classes because they do not perceive a computing career as having "the power to do good and make a difference"[1].

A survey of American high school students who had the aptitude and academic preparation (calculus, pre-calculus) for CS "had no concept of what a Computer Science major entails." This same study found that "students choose not to major in CS because they have an incorrect or no perception of what the field is... The vast majority of students could not provide a description of what Computer Science majors learn"[17]. Although there is a growing number of outreach programs such as CS-Unplugged[10], CS4FN[24], LEGO League(s)[47], Digital Divas[23], Project Impact[50, 31], the Computer Science Inside Project[25], and the Bebras contest[26] that work to correct the myths and misconceptions associated with the discipline, those best positioned to influence and alter computing's image are CS educators themselves.

1.1.1 Introductory courses

Introductory courses that consistently fail to showcase in an integrated way the social value of CS across a broad spectrum of fields are wasted opportunities to recruit those who may be academically prepared, but reside outside the traditional CS student demographic. Furthermore, it may be that such courses reinforce student misconceptions instead of working to dismantle them.

Introductory computing courses provide what may be the only opportunity for introducing examples of the social value of CS to students with no prior exposure to CS. Pedagogical approaches need not, however, be limited to solving problems in external domains at the application level. Bioinformatics algorithms, developed specifically to process genomic and proteomic data, are introductory-level topics for which the social value is both apparent and implicit. A course that interweaves the teaching of both introductory biology and CS showed not only that participants gained the same level of CS1 knowledge and skills as their standard CS1 counterparts, but they also enrolled in CS2 at equal or greater numbers[29]. Courses in computational journalism have utilized CS algorithms and social science principles to mine data for patterns that uncover stories that would otherwise go unreported[70]. In fact, the emergence of Big Data in an ever growing list of disciplines presents an opportunity for students to grasp computing's potential for social impact in a large number of (non-traditional) ways[56].

Students come with a great and varied amount of background knowledge that can serve as the starting points of problems for which CS can indicate solutions. Pedagogically, the more connections students can make for new concepts, especially to prior knowledge, the better the retention[49].

United States enrollment data consistently shows near equal or over-representation of women in biology, chemistry, environmental science, calculus, statistics, history, and the humanities[2]. Making clear connections of CS to a wide variety of topics and disciplines – especially to ones that students already perceive as meaningful as indicated by current participation rates – is one strategy to motivate students, particularly those traditionally underrepresented, to enroll in subsequent CS courses.

1.1.2 Making the Projects Work

Regardless of how attractive CS educators may believe game programming, mobile app development, or duck counting may be for introductory students, we have a professional obligation to introduce CS as a discipline in a widely framed and significant manner. While this perspective does not necessarily banish game programming from the introductory curriculum, it does argue for the inclusion of CSG-Ed projects.

CSG-Ed projects need not necessarily be based on external domains. The current CS Principles pilot courses scattered across the U.S.A. have been incubators for innovative, and sometimes impressive, curricular ideas [6]. A steganography lesson, developed at the Univ. of Washington, showcases the use of bit-shifting operations to hide one image inside another[81]. The "wow" factor cannot be overemphasized when one sees a color photograph of a "subversive" Tahrir Square protest emerge from a conventional B&W photo of a foggy fishing scene.

In the following sections we describe previous work in this area and then ask why CSG-Ed is not more prevalent. The barriers identified are then used as the basis for a rubric to describe example CSG-Ed assignments.

2. RELATED WORK

The inclusion of social good in CS education is not a new notion. Schneiderman, back in 1971, argued that students should be equally stimulated to study sociological modeling as faster algorithms for eigenvalue calculations[79]. Along with an increasing recognition of the socially valuable contributions of computing [46, 83], there is a consequent call for inclusion of CSG-Ed activities in the computing curricula [22, 38, 51, 58].

Such approaches have clear direct benefit in the involvement in social good, but also there is some evidence to suggest that success in broadening participation may be improved when computing is shown to connect with students' values rather than their more superficial interests[7, 16, 28, 36]. There is a growing body of experience reports describing CSG-Ed experiences in upper division courses such as software engineering (e.g. [13, 32, 67]), the senior capstone project (e.g. [5, 53, 80]), service learning experiences (e.g. [74, 78]), sustainability (e.g. [15, 33]), and within fields such as HCI (e.g. [60, 65]). Hence only those students who have survived one to two years of duck counting and Checkers will finally get exposed to the potential social value contributions of computing.

Unfortunately, the literature describing CSG-Ed activities for introductory computing students is thin[29, 37, 64, 68].

Delaying the inclusion of CSG-Ed projects until the third or final year is at best problematic. In addition to the obvious problem of potentially losing interested students to more "meaningful" majors in the first (or second) year, there is evidence which suggests coverage of such material is more effective when it is not segregated into a separate course. The inclusion of both ethics in CS curricula[39] and writing across the curriculum[76] have been shown to be more effective when spread or "integrated" throughout the curriculum.

One recent study endeavored to test the effect that CSG-Ed programming projects have on CS students[72]. After incorporating a CSG-Ed project into both their CS1 and Software Engineering courses, first year students and CS seniors were surveyed regarding their interest in humanitarian projects. Both cohorts indicated that they liked humanitarian projects (79% & 92%), with the seniors reporting the higher approval ratings. Only 11% of the first year students were CS majors, with the majority being declared engineering majors.

When comparing CSG-Ed projects to more traditional assignments, the CSG-Ed projects did not fare so well; especially when compared to game-based projects. While women, in comparing CSG-Ed projects to other projects still ranked CSG-Ed projects positively, none of the CSG-Ed projects received a positive ranking from the male students. The survey's results are of limited scope due to the lack of female participation in the study and that the CS1 cohort contained a majority of students (engineering majors) who admittedly disdain programming. The study did reveal that while it is possible to successfully deploy CSG-Ed projects into introductory courses, the perceived increased difficulty and open-ended nature of their particular CSG-Ed project (search and rescue) affected student appeal. The study concludes by observing that care must be exercised to ensure that CSG-Ed programming projects for introductory courses require "the same level of difficulty and require the same level of programming skills as the other problems in the course"[72], a sentiment echoed in an Australian study of engineering curricula infused with topics of social and environmental justice[63].

Finally we observe that academic service learning experiences have the potential for introductory student participation in CSG-Ed projects. Given the extent of introductory students' disciplinary knowledge, it is not surprising that the academic service learning projects for these students fall back on the expertise of their instructor and center around social good through providing educational resources (e.g. [4, 19, 30, 62, 71, 75]). The canonical example, which one of the authors is involved with, is to have introductory students assist at an after school Scratch/Alice/Kodu/Lego club at a local (disadvantaged?) middle school, or to help run a computer recycling program.

3. BARRIERS TO INTEGRATION OF CSG-ED

Given that there are many advantages to CSG-Ed, the obvious question is why so few CSG-Ed projects exist for the introductory level. There is, of course, no simple answer, but rather many interrelated factors that have delayed CSG-Ed development. Here we subjectively comment on a few of the most significant elements of this problem.

Historical reasons. Concurrent with the dot-com bust, enrollments in CS courses dropped precipitously [86]. Instructors desperate to fill courses tried making introductory courses more "attractive" to students. Unfortunately, "attractive" was usually defined as graphics and game projects which appealed to the contemporary CS student.

What is the role of a teacher? In *Teaching as a Subversive Activity*, Postman and Weingartner argue that teachers do not inject social values into their teaching because they do not conceive themselves as having this role [69]. Instead, they see themselves as information providers, or job trainers. We see this in the current culture of computer science: CS instructors see themselves as information providers and trainers of future computing practitioners (or graduate students), rather than deep educators.

Extraneous load. Cognitive load theory posits that we should reduce the *extraneous load* of our assignments[27]; as such, many instructors have done what they can to strip the context of their assignments. By omitting context, the idea is that students will focus on the CS materials and not be distracted. However, context provides motivation to students [48], and the evidence for cognitive load theory is mixed [43].

Overworked teachers. With little time for curriculum design, most instructors will turn to existing assignments. Many will reuse previous assignments, their colleagues' assignments, or textbook-provided assignments/resources. Overworked teachers are less likely to navigate domain knowledge, and hence are more likely to produce context-free assignments.[2]

Examples in textbooks. Prior to the wide-spread adoption of Java and Python in the introductory curriculum, CS textbooks had examples and exercises with scientific contexts. Indeed, there was a rich body of work providing Pascal projects for instructors to use, such as [20]. However, in the switch to Java, the new generation of textbooks provided examples and exercises with no context, such as bank accounts. Two reasons may be behind this move: first, the switch to Java represented a cultural shift towards producing professional practitioners, rather than educating scientists. The second lies in the notion that the medium is the message [59]: Pascal and similar languages were set up for line output, allowing for easy abstractions and a primary focus on algorithm development; the shift to Java also represents a shift to simple graphics – and from there, a shift to assignments such as to move a circle across a screen.

Scoping issues for CS1. While some work has already been done in adding social context to software engineering courses, databases, and other higher level courses (see Section 2 for more details), introductory courses present more difficulty. Typically the "meaty" problems that are interest-

[2]We do not differentiate between a context-free assignment (e.g. create a stack class) and a universal-context assignment (e.g. bank account or movie rental service).

ing to tackle require a level of depth, expertise, or commitment that we cannot expect at the introductory level. Further, if an instructor consults a colleague in, say, chemistry, for advice on CS assignments with a chemistry context, the chemist is likely to provide an example that is too complex for the introductory level.

Instructors' lack of domain knowledge. Our educational model produces specialists: computer scientists who know computer science, and little of the disciplines which use it. Without a rich understanding of, for example, physics, it can often be difficult for a CS instructor to find and motivate a physics-influenced introductory assignment.

Fears of instructors. In talking to CS instructors, we've encountered two fears. First, instructors' lack of domain knowledge leads them to fear looking incompetent in front of their students. Should students ask domain-based questions when doing the assignment, the instructors worry about being unable to answer the questions. Second is the issue of student feedback. While there is evidence that students appreciate context-based assignments in CS [34], instructors may fear shifting away from popular assignments on games or mobile development. In a system which focuses on student evaluations, the reward system is set up to select for popular assignments rather than valuable assignments.

4. APPROACH

The potential scope of CSG-Ed can be considered as the set produced from the interaction of computer science domains, social good domains and approaches to educational engagement. Clearly this is a very large set that we cannot hope to describe completely. Therefore, we propose that for any non-theoretical computer science domain, topic, or even task, one should be able to articulate a useful CSG-Ed assignment. The key here is *useful*. We argue that utility in this case is to provide a vehicle for teaching an aspect of the introductory curriculum, engage in social good, and overcome the barriers described in Section 3 above.

It is not our intention to attempt to provide a comprehensive list of these possible projects, however. Rather, we provide a set of framing examples that provide the basis for a discussion of the characteristics of CSG-Ed.

The case studies are presented in the Appendix and summarized below. It should be noted that the collection of case studies is not intended as a complete course in CS.

When addressing domains covered in the projects it became difficult to conceive of a specific ontology that might be suitable, as the scope of domains addressed was simply too broad. The Software Engineering curriculum[52, 54] outlined a set of systems and application specialities which addressed some application domains, but these again were framed somewhat restrictively at a system level (e.g. network-centric systems, financial and e-commerce systems, highly secure systems, bio-medical systems). Furthermore, even when addressing CS as a single domain, the CS ontology project has found itself facing huge challenges[18]. Therefore we have opted for a categorisation scheme which captures the six elements which we have considered essential, and augmented those with a narrative description applying the practice bundle pattern proposed by Fincher, Petre & Clark[35]. Each case study, then, is described according to a computer science project work practice bundle. In addition to this meta-data and basic description, a rubric is used to elaborate specific CSG-Ed aspects; See Table 1.

Case studies have in common that all contribute to introductory computing. All are designed to be completed as an assignment, series of assignments, or programming project, each taking approximately two weeks to a month.

4.1 CSG-Ed Rubric Table Explained

- **Student directed:** Describes the extent to which the assignment is determined by the student. Such learner control aids engagement and is considered a fundamental aspect of values based education[82]. Reeves [73], cited in Clear[21], describes the pedagogical dimensions of interactive learning from which these categories are derived.

- **Scaffolding - Instructor resourced:** Reflects the amount of instructor prepared material required to support the learning. We observe that a project can be student directed and still require support from the instructor. Prepared material can take two forms: frameworks, libraries or classes upon which a solution program is built and code that can be used directly in a solution. This category addresses only the latter.

- **External domain knowledge:** Reflects the amount of non-computer science material that is needed to support a student's engagement in the assignment. The delivery mechanism of the material is not distinguished in this category–rather each case study describes whether this is instructor delivered, student delivered, or a combination.

- **Social Good contribution:** An adaptation of a description of student engagement in sustainability projects [11]. A self-referenced project (i.e. without external context) or a traditional not-social good assignment (such as a Tower of Hanoi game) would be classified here as "none." Projects with a non-traditional context, such as biological systems would be classified as "some." A project that explicitly addresses a social good problem such as a sorting system for a humanitarian mission would be medium, and high indicates a real world problem brought by stakeholders and with real world benefits rather than just an exercise. Note that the intention here is to consider the social good contribution of each specific class exercise and assignment *per se* rather than the social good of CS education itself.

- **Coolness:** Attempts to describe elements of ease of student engagement. Most studies of engagement count process (e.g. class attendance) or changed attitudes towards a subject but few attempt to predict the sexiness of assignments. Maybe we are deluded.

- **Explicit reflection:** Describes the extent to which students are encouraged to reflect on the social good aspects of the assignment. The metric here is derived from Hatton and Smith [41]. Reflecting on the technical aspects without reference to the social good is considered "none." The degree of reflection increases from the technical reflection through reflection on action (including descriptive) to dialogic and critical reflection. The last class includes consideration of multiple viewpoints and consideration on the role of the profession.

	None	Some	Medium	High	Basis for scale
Student directed	Students carry out work as directed. Outcome predetermined. Reductionist.	Students select topics from predetermined list.	Students negotiate tasks.	Learners select goals, inquiry based learning. Constructivist pedagogy.	[73] in [21]
Scaffolding: Instructor resourced	No resourcing required.	Some scaffolding provides overview for solution sets. Tight sequencing.	Skeleton frameworks or code.	Extensive code structures supplied.	[73] in [21]
External domain knowledge	None – entirely in CS traditional suite, or universally known context.	Brief introduction required. (Some Wikipedia may be needed.)	Requires reading.	Requires subject expertise.	External domain knowledge (fear factor)
Social Good Contribution	None.	Something done in external context (e.g. biological humanities).	Explicitly social good.	Actual good contribution. (i.e. Has real world outcomes.)	[11]
Coolness	Difficult to engage. Not tractable. What's this got to do with computing?	Fosters interest	Retains interest. Is considered relevant.	"Sexy," immediate engagement. Relevant. Inspirational.	
Explicit reflection on social good aspects	None.	Technical reflection. (What I did, related to personal experience) *social good aspect.	Reflection on action. (Descriptive – best practice)	Dialogic – viewpoints, alternative solutions * about social good aspects. Critical – role of profession, impact on other and wider forces.	[41]

Table 1: The CSG-Ed Rubric Table

5. THE CASE STUDIES

The Appendix includes a variety of ready-to-use *introductory* CSG-Ed projects. These projects reflect the varied interests of the authors in both domains and approaches. As a result, the topics covered range from largely scientific issues (e.g. astronomy) to disaster management to voting systems. Furthermore, the approaches utilized a range from typical first year programming projects transformed into a CSG-Ed project (e.g. a shortest path implementation becomes a Red Cross disaster response program.) to projects that are centered on particular social issues. Despite this range, the projects presented in the Appendix demonstrate certain characterizations for each of the four categories used in their classification. In the following paragraphs we list the projects, discuss the computer science topics they cover, and summarize the project characterizations.

5.1 The CSG-Ed projects

- **Radioactive mice** - Simulation of a robot seeking out and catching contaminated mice: Section A.
- **Red Cross disaster relief** - Application of the shortest path algorithm to determine which red cross office should respond to a disaster: Section B.
- **The use of nuclear power** - GIS application examining safety of nuclear plant sites: Section C.
- **Water pollution** - GIS application examining point-source water pollution: Section D.
- **Ad hoc emergency Wi-Fi networking** - Simulation of an emergency ad hoc WiFi-based network: Section E.
- **A mini package manager for software distribution**: Section F.
- **Voting simulation** - An examination of how different voting systems affect democratic elections: Section G.
- **STI modeling** - Using graph models to examine disease propagation through a population of sexual partners: Section H.
- **Banana republic** - Modeling population growth, crop growth, boat migrations for a set of island populations: Section I.
- **Kiwi population** - Modeling of various aspects of food production, population growth and food distribution: Section J.
- **Molecular modeling (MM) and DNA** - Building a 2-D MM program to study hydrogen bonding between bases in the DNA double helix: Section K.
- **Around the world** - Simulation of a rotating Earth to examine the "Around the World in 80 Days" phenomenon: Section L.
- **Social good website** - Web development for community organization or development initiative: Section M.
- **Social good scholarly work** - Formally analyze an aspect of communication technology: Section N.

5.2 Computer Science Topics

The computer science topics covered in the projects presented here fall mainly in the ACM knowledge areas[3] *Fundamental programming structures* (PF1), *Algorithms and problem solving* (PF2), and *Fundamental data structures* (PF3). Not all topics in these knowledge areas are covered, but a significant number of the topics typically covered in an introductory sequence are addressed. Table 2 relates the CS topics to individual projects. The breadth of topics covered

	Radioactive Mice	Red Cross Disaster Relief	The use of Nuclear Power	Water Pollution	Ad Hoc Emergency Wi-Fi	A mini package manager	Voting Simulation
‘ Random # Gen.	X						X
Encapsulation/Classes	X						
Inheritance/Polymorphism							
Iteration				X			X
Selection			X	X			X
1-D arrays			X				
2-D arrays				X			
ArrayLists	X						X
File I/O			X	X			
Dictionaries							X
Searching	X					X	
Sorting	X					X	
Graphs/Algs		X			X	X	
Software Eng							
GUI/Event driven							
Other							

	STI modeling	Banana Republic	Kiwi Population	Molecular Modeling and DNA	Around the world	Social good website	Social good scholarly work
Random # Gen.	X		X				
Encapsulation/Classes		X	X	X	X		
Inheritance/Polymorphism				X			
Iteration		X	X	X	X		
Selection		X	X	X			
1-D arrays		X	X	X	X		
2-D arrays							
ArrayLists			X	X			
File I/O	X						
Dictionaries							
Searching							
Sorting							
Graphs/Algs	X			X	X		
Software Eng				X	X		
GUI/Event driven				X	X		
Other						X	X

Table 2: CS concepts covered

indicates how easily socially relevant issues can be incorporated into introductory courses.

Although the topics in the projects range from the very introductory level to a relatively sophisticated level, most projects cover topics that require students to have mastered some simple programming skills. Examining Table 2, it is clear that most of the projects use aggregate data structures (arrays, lists, dictionaries) and at least a third of the projects cover relatively sophisticated graph or searching algorithms.

5.3 Summary of project set categorizations

The projects presented in this paper vary greatly in the emphasis given to each of the categories. In the following paragraphs we summarize the projects in terms of their category emphasis to give interested instructors a feel for the breadth and types of projects offered in this paper. Table 3 provides complete categorization for all projects.

Student Directed: Most of the projects in this paper require at least some student self-initiative, though only one allows students complete freedom to determine the direction of the project. In the projects that do allow some student participation in the project direction, the involvement usually takes the form of choosing among a list of topics or uses guided discovery (such as the “Molecular Modeling and DNA” project). Two projects, the nuclear power project and the water pollution project, require students to determine the criteria that they use in their analysis and several projects (such as the “Ad Hoc Wi-Fi” project and “the mini package manager”) allow students to extend the project or add features for credit. Several projects (“Radioactive Mice,” “Red Cross Disaster Response”) also have flexible assessment rubrics that allow students some flexibility in how they complete assignments.

Scaffolding: Most of the projects presented here provide minimal scaffolding for the student. Several (including the “Nuclear Power” and “Water Pollution” projects) provide frameworks or classes that can be used by students in their solution. In several projects (such as the “Banana Republic” and “Kiwi Population” projects) code similar to the solution is provided to give students an understanding of the program structure. Three of the projects provide extensive scaffolding. The “DNA,” “Around the World,” and “Voting” projects provide the outline of at least some of the solution code to students. The former two projects use a guided discovery approach and much of the code used by students is created in the classroom.

External Domain Knowledge: Only two of the projects presented here require extensive knowledge of an external field, the “Molecular Modeling and DNA” and “Around the World” projects. These projects both use the guided discovery approach. The former requires instructors to present geometry, trigonometry, and molecular biology principles while the latter requires the instructor to have a good grasp on geography. Four of the projects require neither student nor instructor to have knowledge of an external domain; the projects are motivated by the external domain but the problem has been reduced to pure computer science concepts. The remaining projects are split between requiring “some” or “medium” external domain knowledge. In the projects that require “some” knowledge the knowledge is usually well known to the target audience (“Banana Republic,” “Kiwi Population,” “Voting,” “Ad Hoc Emergency Wi-Fi,” “Social Good Website”). In the projects rated “medium” the knowledge is discovered by the students themselves (“Nuclear Power,” “Water Pollution,” “Social Good Scholarly Work,” “STI Modeling”).

Social Good: Given the purpose of this paper, all of the projects provided here contribute in some way to the social good. The “high” rating requires real community engagement and only a few of the projects reach that goal

	Radioactive Mice	**Red Cross Disaster Relief**	**The use of Nuclear Power**	**Water Pollution**	**Ad Hoc Emergency Wi-Fi**	**A mini package manager**	**Voting Simulation**
Student directed	some	some	some	some	none	none	none
Scaffolding. Instructor resourced	none	none	some	some	none	none	high
External domain knowledge	none	none	medium	medium	some	none	some
Good contribution	some	medium	medium	medium	medium	medium	medium
Coolness	some	medium	medium	high	medium	high	medium
Explicit reflection on social good aspects	medium	medium	high	high	high	medium	medium

	STI modeling	**Banana Republic**	**Kiwi Population**	**Molecular Modeling and DNA**	**Around the world**	**Social good website**	**Social good scholarly work**
Student directed	none	some	some	medium	medium	medium	high
Scaffolding. Instructor resourced	some	medium	medium	high	high	some	some
External domain knowledge	none	some	some	high	medium	some	none to medium
Good contribution	some	some or medium	some or medium	high	high	high	medium
Coolness	high	medium	medium	high	high	some	medium
Explicit reflection on social good aspects	some	some	some	medium	medium	medium	high

Table 3: Project Categorization

("Molecular Modeling and DNA," "Around the World," "Social Good Website"). Several of the projects are framed by social issues ("Radioactive Mice," "Red Cross Disaster Relief," "Ad Hoc Emergency Wi-Fi," "Mini Package Manager") but don't require students to investigate or reflect on those issues. The remaining projects require students to either research, reflect on, or in some way confront social good issues.

Coolness: The projects included in this paper tend to be more highly rated in this category. The idea of social relevancy often captures student attention and these projects focus, for the most part, on high-profile issues. In some of the projects, such as "Nuclear Power" or the role of computer science in emergency situations ("Red Cross Disaster Relief" or "Ad Hoc Emergency Wi-Fi"), the topic itself is engaging. In other exercises, simulations ("Banana Republic" or "Radioactive Mice") are used for engagement. Although these latter projects are less realistic or less directly related to social good, they provide an environment similar to those of virtual world games and encourage students to think more broadly about social issues.

Reflection: All of the projects presented in this paper require some reflection on social issues, though this reflection is, for many of the projects, indirect. Through their focus on socially relevant issues and through the process of developing a program that deals with some aspect of a socially relevant issue, students are forced to reflect, to some degree, on the issue. There are some projects, however, that make this reflection more purposeful. The "Nuclear Power," "Water Pollution," and "Social good scholarly work" projects all require explicit reflection in the form of a paper or technical report.

5.4 Limitations of our work

While the CSG-Ed approach can capture the interest of students who did not realize the social applications of computer science, more *techie* students may not appreciate these projects. In fact, their main interests are more related to methods and technical details rather than to motivations and means.

The assignments presented here are class-tested to varying degrees. While it is hoped that instructors could use them "as is," we make no promises as to their robustness. Rather the intention of including them here is to provide material to support the exploration of the proposition that for any non-theoretical computer science domain topic, one should be able to articulate a useful CSG-Ed assignment. A significant number of the topics typically covered in the first year are addressed.

5.5 Allaying the barriers

The barriers described in Section 3 can be daunting, but are by no means insurmountable. CS instructors are well aware of the need to appeal to a larger audience to increase enrollments, especially of minorities and women so the historical patterns are already changing. This paper presents projects that should appeal to a much wider range of students and also addresses the needs of overworked instructors.

The projects presented here will also ameliorate the problem of domain knowledge and the resulting feeling of inadequacy. Most of the projects supply sufficient domain information for both student and instructor and only two projects ("Molecular Modeling and DNA" and "Around the World") require deep domain knowledge on the part of the instructor. Some of the projects (e.g. "Red Cross Disaster Relief" and

"Radioactive Mice") illustrate how traditional introductory assignments can, with a little creativity, be repurposed into a CSG-Ed assignment.

- "Radioactive Mice" is a reworked version of a Greenfoot-based game called *Catch Me If You Can*.
- "Red Cross Disaster Relief" is a reworked example of a shortest-path algorithm assignment, often presented at the introductory level as the *Travel Agent Problem*.

The projects also address the problem of extraneous load on students. Many of the projects (such as the "Radioactive Mice" and the "Mini Package Manager") place minimum extra overhead on the students. Others (such as "Nuclear Power" or "Molecular Modeling and DNA") place a much higher burden on the students, yet even these assignments are constructed in such a manner that the instructor can minimize the time spent pursuing outside domain knowledge.

All of the assignments integrate many CS concepts and in most of them the external domain assessment can, if necessary, be reduced. For example, the technical report in both the "Nuclear Power" and "Water Pollution" projects can be omitted. As a result, the overhead problem can, in most cases, be minimized.

The only serious barrier, then, is the availability of introductory CS textbooks that address social good issues. This is, of course, the horse and cart problem; textbooks with problems that deal with social good issues would raise instructor awareness, but will not be written until there is sufficient demand from instructors. The projects in this paper, which can be used with any textbook, are a first step in raising both awareness and demand.

6. CONCLUSIONS

In this report we have made a case for the merits of Computer Science Education for Social Good (CSG-Ed) projects from an early stage in the curriculum. We considered the background of computing for the social good, motivated the work and proposed a categorisation and an illustrative set of exemplar case study projects intended for CS educators to adopt in their own institutions.

We believe a CSG-Ed approach will better motivate students by providing them with a more comprehensive view of the discipline and its scope for meaningful societal contribution from the very beginning of their CS education. This larger viewpoint often does not appear until the senior capstone project in a traditional CS curriculum by which time their perspective on computer science has already been formulated.

While this report includes a number of CSG-Ed projects, no attempt has been made to formally evaluate their impact. Though such evaluation must be accomplished in the future, we believe that CS educators have a professional obligation to introduce CS as a discipline in a widely framed and significant manner regardless of its potential impact on enrollments. The pervasiveness of computing and its impact on all aspects of our lives suggests that it be seen as the transformational discipline that it truly is, rather than as a frustrating struggle with obscure syntax and problems or as the handmaiden to some other scientific or business discipline. In addition to providing current majors with a more accurate understanding of their profession, this insight will also make the field more appealing to students seeking a major through which they can make a real impact on the world. The projects presented in this paper give instructors a concrete starting point to incorporate a CSG-Ed approach in their teaching.

7. FUTURE WORK

The projects included in the appendix are just a first collection of CSG-Ed ideas. They do not cover all computer science topics nor do they include all students' knowledge levels. However, the heterogeneity of the projects make them a viable proof-of-concept to show the effectiveness of the approach.

Much work also remains to be done field testing the projects. Some of them have been effectively tested in courses while others are new projects designed for this paper. Our intent is that these projects form just the first set of a larger collection that will grow with the contributions from interested lecturers who seek to share their projects and experiences.

In addition to testing the projects, an assessment of the approach itself remains to be done, especially in light of the very preliminary assessment work by Rader et.al.[72]. How does the inclusion of social good concepts affect student perceptions of computer science? Are students more motivated by these problems than traditional CS problems? Are non-traditional CS students attract to CS through these problems? Does this approach enhance the learning of CS concepts? The answers to these questions, and more, will determine the effectiveness of the approach.

8. REFERENCES

[1] Dot diva. `http://dotdiva.org/educators/problem.html`.

[2] College board AP program summary report. `http://professionals.collegeboard.com/data-reports-research/ap/data`, 2011.

[3] ACM/IEEE-CS Joint Interim Review Task Force. Computer science curriculum 2008: An interim revision of CS 2001, report from the interim review task force, 2008.

[4] J. B. Adams and E. Runkles. "May we have class outside?": implementing service learning in a CS1 curriculum. *J. Comput. Sci. Coll.*, 19(5):25–34, May 2004.

[5] R. J. Anderson, R. E. Anderson, G. Borriello, and J. Pal. An approach to integrating ICTD projects into an undergraduate curriculum. In *Proceedings of the 41st ACM Technical Symposium on Computer Science Education*, SIGCSE '10, pages 529–533, New York, NY, USA, 2010. ACM.

[6] O. Astrachan, T. Barnes, D. D. Garcia, J. Paul, B. Simon, and L. Snyder. CS principles: piloting a new course at national scale. In *Proceedings of the 42nd ACM Technical Symposium on Computer Science Education*, SIGCSE '11, pages 397–398, New York, NY, USA, 2011. ACM.

[7] L. J. Barker, C. McDowell, and K. Kalahar. Exploring factors that influence computer science introductory course students to persist in the major. In *Proceedings of the 40th ACM Technical Symposium on Computer Science Education*, SIGCSE '09, pages 153–157, New York, NY, USA, 2009. ACM.

[8] D. Barnes and M. Kölling. *Objects First with Java: A Practical Introduction Using BlueJ*. Pearson: Prentice Hall, 4 edition, 2009.

[9] K. Becker. Grading programming assignments using rubrics. In *Proceedings of the 8th Annual Conference on Innovation and Technology in Computer Science Education*, ITiCSE '03, pages 253–253, New York, NY, USA, 2003. ACM.

[10] T. Bell. Computer science unplugged. `http://csunplugged.org/`.

[11] K. Brundiers, A. Wiek, and L. Redman, Charles. Real-world learning opportunities in sustainability: from classroom into the real world. *International Journal of Sustainability in Higher Education*, 11(4):308–324, 2010.

[12] M. Buckley. Viewpoint: Computing as social science. *Commun. ACM*, 52(4):29–30, Apr. 2009.

[13] M. Buckley, H. Kershner, K. Schindler, C. Alphonce, and J. Braswell. Benefits of using socially-relevant projects in computer science and engineering education. In *Proceedings of the 35th SIGCSE Technical Symposium on Computer Science Education*, SIGCSE '04, pages 482–486, New York, NY, USA, 2004. ACM.

[14] M. Buckley, J. Nordlinger, and D. Subramanian. Socially relevant computing. In *Proceedings of the 39th SIGCSE Technical Symposium on Computer Science Education*, SIGCSE '08, pages 347–351, New York, NY, USA, 2008. ACM.

[15] Y. Cai. Integrating sustainability into undergraduate computing education. In *Proceedings of the 41st ACM Technical Symposium on Computer Science Education*, SIGCSE '10, pages 524–528, New York, NY, USA, 2010. ACM.

[16] J. Carter, D. Bouvier, R. Cardell-Oliver, M. Hamilton, S. Kurkovsky, S. Markham, O. W. McClung, R. McDermott, C. Riedesel, J. Shi, and S. White. Motivating all our students? In *Working Group Report from the 16th Annual Conference on Innovation and Technology in Computer Science Education*, ITiCSE-WGR '11, pages 1–18, New York, NY, USA, 2011. ACM.

[17] L. Carter. Why students with an apparent aptitude for computer science don't choose to major in computer science. *SIGCSE Bull.*, 38(1):27–31, Mar. 2006.

[18] L. N. Cassel, G. Davies, W. Fone, A. Hacquebard, J. Impagliazzo, R. LeBlanc, J. C. Little, A. McGettrick, and M. Pedrona. The computing ontology: application in education. *SIGCSE Bull.*, 39(4):171–183, Dec. 2007.

[19] K. Christensen, D. Rundus, G. Perera, and S. Zulli. CSE volunteers: a service learning program to provide IT support to the Hillsborough County school district. *SIGCSE Bull.*, 38(1):229–233, Mar. 2006.

[20] M. Clancy and M. Linn. *Designing Pascal Solutions: A Case Study Approach*. Principles of Computer Science Series. Computer Science Press, 1992.

[21] T. Clear. Diagnosing your teaching style: how interactive are you? *ACM Inroads*, 1(2):34–41, June 2010.

[22] R. W. Connolly. Beyond good and evil impacts: rethinking the social issues components in our computing curricula. In *Proceedings of the 16th Annual Conference on Innovation and Technology in Computer Science Education*, ITiCSE '11, pages 228–232, New York, NY, USA, 2011. ACM.

[23] A. Craig and J. Fisher. Digital divas club. `http://digitaldivasclub.org/vic/`.

[24] P. Curzon, P. McOwan, and J. Black. Computer science for fun. `http://www.cs4fn.org/`.

[25] Q. Cutts, M. Calder, and P. Dickman. Computer science inside... bring computer science alive. `http://csi.dcs.gla.ac.uk/`.

[26] V. Dagiene. Bebras contest. `http://www.bebras.org/en/welcome`.

[27] T. de Jong. Cognitive load theory, educational research, and instructional design: some food for thought. *Instructional Science*, 38:105–134, 2010. 10.1007/s11251-009-9110-0.

[28] B. DiSalvo and A. Bruckman. From interests to values. *Commun. ACM*, 54(8), 2011.

[29] Z. Dodds and R. Libeskind-Hadas. Bio1 as CS1: Evaluating a crossdisciplinary CS context. In *Proceedings of the 17th Annual Conference on Innovation and Technology in Computer Science Education*, ITiCSE '12, 2012.

[30] M. A. L. Egan and M. Johnson. Service learning in introductory computer science. In *Proceedings of the 15th Annual Conference on Innovation and Technology in Computer Science Education*, ITiCSE '10, pages 8–12, New York, NY, USA, 2010. ACM.

[31] M. A. L. Egan and T. Lederman. The impact of IMPACT: assessing students' perceptions after a day of computer exploration. In *Proceedings of the 16th Annual Conference on Innovation and Technology in Computer Science Education*, ITiCSE '11.

[32] H. J. C. Ellis, R. A. Morelli, T. R. de Lanerolle, J. Damon, and J. Raye. Can humanitarian open-source software development draw new students to CS? In *Proceedings of the 38th SIGCSE Technical Symposium on Computer Science Education*, SIGCSE '07, pages 551–555, New York, NY, USA, 2007. ACM.

[33] A. Erkan, T. Pfaff, J. Hamilton, and M. Rogers. Sustainability themed problem solving in data structures and algorithms. In *Proceedings of the 43rd ACM Technical Symposium on Computer Science Education*, SIGCSE '12, pages 9–14, New York, NY, USA, 2012. ACM.

[34] A. Erkan, T. Pfaff, J. Hamilton, and M. Rogers. Sustainability themed problem solving in data structures and algorithms. In *Proceedings of the 43rd ACM Technical Symposium on Computer Science Education*, SIGCSE '12, pages 9–14, New York, NY, USA, 2012. ACM.

[35] S. Fincher, M. Petre, and M. Clark, editors. *Computer science project work: principles and pragmatics*. Springer-Verlag, London, UK, 2001.

[36] A. Fisher and J. Margolis. Unlocking the clubhouse: the Carnegie Mellon experience. *SIGCSE Bull.*, 34(2):79–83, June 2002.

[37] M. Goldweber. A day one computing for the social good activity. *ACM Inroads*, 3(3), 2012.

[38] M. Goldweber, R. Davoli, J. C. Little, C. Riedesel,

H. Walker, G. Cross, and B. R. Von Konsky. Enhancing the social issues components in our computing curriculum: computing for the social good. *ACM Inroads*, 2:64–82, February 2011.

[39] D. Gotterbarn. Integration of computer ethics into the CS curriculum: attachment or synthesis. In *Working group Report from the 4th Annual Conference on Innovation and Technology in Computer Science Education*, ITiCSE-WGR '99, pages 13–14, New York, NY, USA, 1999. ACM.

[40] M. Guzdial. Teaching computing to everyone. *Commun. ACM*, 52(5), 2009.

[41] N. Hatton and D. Smith. Reflection in teacher education: towards definition and implementation. *Teaching and Teacher Education*, 11:33–49, 1995.

[42] R. Heeks. ICT4D 2.0: The next phase of applying ICT for international development. *Computer*, 41(6):26–33, June 2008.

[43] D. Holton. Cognitive load theory: Failure?, 2009. `http://edtechdev.wordpress.com/2009/11/16/cognitive-load-theory-failure/`.

[44] J. P. Hourcade. Give peace a chance: a call to design technologies for peace. In *Proceedings of the 27th International Conference Extended Abstracts on Human Factors in Computing Systems*, CHI EA '09, pages 2499–2508, New York, NY, USA, 2009. ACM.

[45] J. H. Janet and J. Mertz. Gender, culture and mathematics performance. *Proceedings of the National Academy of Sciences*, 106(22).

[46] L. C. Kaczmarczyk. *Computers and Society: Computing for Good*. Chapman and Hall, 2011.

[47] D. Kamen and K. K. Kristiansen. First lego league. `http://www.firstlegoleague.org/`.

[48] J. S. Kay. Contextualized approaches to introductory computer science: the key to making computer science relevant or simply bait and switch? In *Proceedings of the 42nd ACM Technical Symposium on Computer Science Education*.

[49] J. Kirkpatrick, J. Swafford, and B. Findell. *Adding it up: Helping children learn mathematics*. National Academy Press, 2001.

[50] C. Lang, A. Craig, J. Prey, M. A. L. Egan, and R. Ayfer. Outreach programs to promote computer science and ICT to high school and middle school students. In *Proceedings of the 16th Annual Conference on Innovation and Technology in Computer Science Education*, ITiCSE '11.

[51] L. Layman, L. Williams, and K. Slaten. Note to self: make assignments meaningful. In *Proceedings of the 38th SIGCSE Technical Symposium on Computer Science Education*, SIGCSE '07, pages 459–463, New York, NY, USA, 2007. ACM.

[52] R. LeBlanc, M. Ben-Menachem, T. B. Hilburn, S. Mengel, T. C. Lethbridge, B. Thompson, A. Sobel, and J. L. Díaz-Herrera. IEEE-CS/ACM computing curriculum software engineering volume project. In *Proceedings of the 16th Annual Conference on Software Engineering Education and Training*, CSEET '03, 2003.

[53] P. M. Leidig, R. Ferguson, and J. Leidig. The use of community-based non-profit organizations in information systems capstone projects. In *Proceedings of the 11th Annual Conference on Innovation and Technology in Computer Science Education*, ITiCSE '06, pages 148–152, New York, NY, USA, 2006. ACM.

[54] T. C. Lethbridge, R. J. LeBlanc Jr, A. E. K. Sobel, T. B. Hilburn, and J. L. Diaz-Herrera. SE2004: Recommendations for undergraduate software engineering curricula. *IEEE Softw.*, 23(6):19–25, Nov. 2006.

[55] R. Lister, A. Berglund, T. Clear, J. Bergin, K. Garvin-Doxas, B. Hanks, L. Hitchner, A. Luxton-Reilly, K. Sanders, C. Schulte, and J. L. Whalley. Research perspectives on the objects-early debate. In *Working Group Report from the 11th Annual Conference on Innovation and Technology in Computer Science Education*, ITiCSE-WGR '06, pages 146–165, New York, NY, USA, 2006. ACM.

[56] S. Lohr. The age of big data. New York Times, February 2012.

[57] S. Mann, L. Muller, J. Davis, C. Roda, and A. Young. Computing and sustainability: evaluating resources for educators. *SIGCSE Bull.*, 41(4):144–155, Jan. 2010.

[58] S. Mann, L. Smith, and L. Muller. Computing education for sustainability. *SIGCSE Bull.*, 40(4):183–193, Nov. 2008.

[59] M. McLuhan and Q. Fiore. *The Medium Is the Message: An Inventory of Effects - Centennial Facsimile Edition*. Gingko Press, 2011.

[60] L. P. Nathan, E. Blevis, B. Friedman, J. Hasbrouck, and P. Sengers. Beyond the hype: sustainability & HCI. In *CHI '08 extended abstracts on Human factors in computing systems*, CHI EA '08, pages 2273–2276, New York, NY, USA, 2008. ACM.

[61] L. P. Nathan, P. V. Klasnja, and B. Friedman. Value scenarios: a technique for envisioning systemic effects of new technologies. In *CHI '07 extended abstracts on Human factors in computing systems*, CHI EA '07, pages 2585–2590, New York, NY, USA, 2007. ACM.

[62] R. B. Osborne, A. J. Thomas, and J. R. Forbes. Teaching with robots: a service-learning approach to mentor training. In *Proceedings of the 41st ACM Technical Symposium on Computer Science Education*, SIGCSE '10, pages 172–176, New York, NY, USA, 2010. ACM.

[63] J. O'Shea and C. Baillie. Engineering education towards social and environmental justice. In *Proceedings of the 22nd Annual Conference for the Australasian Association for Engineering Education*, 2011.

[64] J. Owens and J. Matthews. Cybercivics: a novel approach to reaching K-12 students with the social relevance of computer science. In *Proceedings of the 39th SIGCSE Technical Symposium on Computer Science Education*, SIGCSE '08, pages 372–376, New York, NY, USA, 2008. ACM.

[65] L. Patricia. Service learning: an HCI experiment. In *Proceedings of the 16th Western Canadian Conference on Computing Education*, WCCCE '11, pages 12–16, New York, NY, USA, 2011. ACM.

[66] D. A. Patterson. Rescuing our families, our neighbors, and ourselves. *Commun. ACM*, 48(11):29–31, Nov. 2005.

[67] V. P. Pauca and R. T. Guy. Mobile apps for the

greater good: a socially relevant approach to software engineering. In *Proceedings of the 43rd ACM Technical Symposium on Computer Science Education*, SIGCSE '12, pages 535–540, New York, NY, USA, 2012. ACM.

[68] S. R. Portnoff. Teaching HS computer science as if the rest of the world existed: rationale for a HS Pre-APCS curriculum of interdisciplinary central-problem-based units that model real-world applications. In *Proceedings of the 43rd ACM Technical Symposium on Computer Science Education*, SIGCSE '12, pages 243–244, New York, NY, USA, 2012. ACM.

[69] N. Postman and C. Weingartner. *Teaching as a Subversive Activity*. Delacorte Press, 1969.

[70] S. Pulimood, D. Shaw, and E. Lounsberry. Gumshoe: A model for undergraduate computational journalism education. In *Proceedings of the 42nd ACM Technical Symposium on Computer Science Education*, SIGCSE '11, 2011.

[71] T. S. Purewal, Jr., C. Bennett, and F. Maier. Embracing the social relevance: computing, ethics and the community. In *Proceedings of the 38th SIGCSE Technical Symposium on Computer Science Education*, SIGCSE '07, pages 556–560, New York, NY, USA, 2007. ACM.

[72] C. Rader, D. Hakkarinen, B. M. Moskal, and K. Hellman. Exploring the appeal of socially relevant computing: are students interested in socially relevant problems? In *Proceedings of the 42nd ACM Technical Symposium on Computer Science Education*, SIGCSE '11, pages 423–428, New York, NY, USA, 2011. ACM.

[73] T. Reeves. Effective dimensions of interactive learning systems. In *Information Technology for Training and Education*, Brisbane, Australia, 1992. Australia Information Technology for Training and Education.

[74] S. Reiser and R. Bruce. Service learning meets mobile computing. In *Proceedings of the 46th Annual Southeast Regional Conference*, ACM-SE 46, pages 108–113, New York, NY, USA, 2008. ACM.

[75] B. J. Rosmaita. Making service learning accessible to computer scientists. In *Proceedings of the 38th SIGCSE Technical Symposium on Computer Science Education*, SIGCSE '07, pages 541–545, New York, NY, USA, 2007. ACM.

[76] D. Russell. *Landmark Essays on Writing Across the Curriculum*, chapter American Origins of the Writing-across-the-Curriculum Movement. 1994.

[77] M. Schuhmacher and S. Markham. Applying rubrics assessment in undergraduate computer science education. In *Proceedings of the 15th Annual NACCQ*, 2002.

[78] R. A. Scorce. Perspectives concerning the utilization of service learning projects for a computer science course. *J. Comput. Sci. Coll.*, 25(3):75–81, Jan. 2010.

[79] B. Shneiderman. Computer science education and social relevance. *SIGCSE Bull.*, 3(1):21–24, Mar. 1971.

[80] L. Smith and S. Mann. Sustainable software engineering. In S. Mann and M. Verhaart, editors, *1st Annual Conference of Computing and Information Technology, Education and Research in New Zealand (incorporating 23rd Annual NACCQ)*, pages 366–367. CITRENZ/NACCQ, 2010.

[81] L. Snyder. Steganography. University Lecture, 2011.

[82] S. Sterling. *Higher Education, Sustainability, and the Role of Systemic Learning*, pages 50–70. Kluwer Academic, NY, 2004.

[83] B. Tomlinson. *Greening through IT: Information Technology for Environmental Sustainability*. The MIT Press, 2010.

[84] A. Tucker, R. Morelli, and T. d. Lanerolle. The humanitarian FOSS project: Goals, activities, and outcomes. In *Proceedings of the 2011 IEEE Global Humanitarian Technology Conference*, GHTC '11, pages 98–101, Washington, DC, USA, 2011. IEEE Computer Society.

[85] S. Yardi and A. Bruckman. What is computing?: Bridging the gap between teenagers' perceptions and graduate students' experiences. In *Proceedings of the 3rd International Workshop on Computing Education Research*.

[86] S. Zweben and B. Bizot. CRA-Taulbee survey. `http://cra.org/resources/taulbee/`.

APPENDIX

For more complete project descriptions either contact the project's author directly or visit the "CSG-Ed community" which resides as part of the Computing Portal (`www.computingportal.org`).

A. RADIOACTIVE MICE

MICHAEL GOLDWEBER

The dispersal of radioactive contamination can be caused by many factors; weather, water, human intervention as well as animal vectors. One famous case lampooned by a popular folk song was the radioactive frogs found in the early "1990's" around the Oak Ridge National Laboratory (USA). The frogs had been living in the mud of a half-acre holding basin for waste water from the lab's nuclear research in the "1940's" and "1950's." In 2010 radioactive mice and rabbits were found in the area around the plutonium production facility in Hanford, Washington, USA.

More recently rats contaminated with cesium have been found as far away as 30 kilometers from the Fukushima nuclear power plant. Containment of radioactive contaminated animal vectors is a particularly vexing problem. Contaminated animals can get eaten by predators, which typically have a larger habitat range (e.g. consider a mouse or rat that is eaten by a fox). Feces of contaminated animals can pollute water sources. Even if the contaminated animal dies "naturally," there is still the question of how much contamination is left in the soil.

Autonomous robotic devices have been developed for a variety of purposes; vacuuming a room or mowing a lawn. Not surprisingly, autonomous robotic devices to catch (and kill?) rodents for both interior and exterior applications are being experimented with. Consider a robotic "cat" whose job it is to keep a nuclear energy facility rodent free.

A.1 Categorization

Student Directed: Some, based on an open ended assessment rubric.
Scaffolding: None.
External Domain Knowledge: None.
Social Good: Some.
Coolness: Some.
Reflection: Medium.

A.2 CS Concepts

Encapsulation: 2d locations as objects; ArrayLists; The Examine-All (i.e. Find Closest) array processing algorithm.

A.3 Implementation Strategies

A.3.1 The way it works

For this project, students are to use Greenfoot to create a variation of the robotic rodent catching cat (r^2c^2). Consider an $n \times n$ grid world where there is one instance of the r^2c^2, whose starting location is selected randomly, and m stationary radioactive mice (possibly already dead from radiation overdose), whose starting positions are also random. Furthermore, there are obstacles in the world through which neither the r^2c^2 nor any of the mice can pass. Students need to endow the r^2c^2 with an algorithm to successfully "catch" each radioactive mouse.

r^2c^2 must employ something more sophisticated than brute force search. Using the Greenfoot API, the r^2c^2 can learn the location of one mouse (or all of the mice) and move purposefully towards the closest mouse, one at a time. The r^2c^2 can only move one square at a time (i.e. per invocation of "act"). Diagonal moves are permitted. Initially, students should consider the world a "closed" world; hence the exterior grid boundaries are hard. When r^2c^2 occupies the same cell as a mouse, the mouse is considered "caught" and is removed from the world. When all the mice have been caught, the r^2c^2 should recognize this case and shut down the program, though a celebratory dance may first be in order.

A.3.2 It works better if

No suggestions.

A.3.3 Assessment strategies

This project lends itself to being open-ended. In keeping with the proposal that project assignments should also convey the evaluation rubric[9, 77], one such rubric, implementing both the design paradigm of *iterative improvement* and open-ended creativity might be:

1. C: Students must have a correctly working program for m stationary mice in a world bounded with hard boundaries.
2. B: The requirements for a C in addition to excellent documentation. Furthermore, the mice are not dead and can move. Like the r^2c^2, the mice can only move to one adjacent square at a time (including diagonals). The mice should move at a rate slower than the r^2c^2, selecting the next cell to occupy randomly. The r^2c^2 should always take one purposefully selected step towards the mouse closest to it.
3. A: The requirements for a B in addition to some other open-ended improvements. Here is the opportunity for students to think creatively. Some examples of such improvements are:
    - Consider the world a torus (i.e. allow full wrap-around). This complicates r^2c^2's nearest mouse algorithm.
    - Allow the mice to reproduce. Individual mice can sense other mice, up to some distance limit (e.g. two squares away) and move purposefully toward each other. When two mice occupy the same square for an iteration, they spawn a pinkie (i.e. baby mouse).
    - Allow some obstacles to be such that a mouse can hide in them. So while a mouse can co-occupy a square with an obstacle, the r^2c^2 cannot. Mice can only stay hidden in an obstacle for a fixed number of iterations.

A.3.4 It doesn't work unless

This project requires an event-driven grid world environment such as Greenfoot.

A.4 Extensions

In addition to the above, students could experiment with multiple r^2c^2 cats.

A.5 Deliverables

The code for the r^2c^2, and possibly for the mice as well. A short reflection paper on the possible other uses of such technologies is recommended.

B. RED CROSS DISASTER RESPONSE

MICHAEL GOLDWEBER

In most countries there is a branch of the International Red Cross (`icrc.org`); Red Crescent Societies, Mogen David Adom, and national Red Cross societies. In each case, the organization is tasked with responding to emergencies; both natural and man-made. For a given disaster, it is the responsibility of the closest Red Cross office with the appropriate resources/supplies to respond. The question is, given a disaster location along with the locations of all Red Cross offices, which one should respond?

This problem is an example of what is called a "shortest path scheduling algorithm." For this problem one needs a graph. Each node in the graph represents a *city*. The edges (or links) represent highways connecting the cities. Each edge has a distance or mileage value associated with it. Furthermore, some, but not all, of the cities house a Red Cross office. Finally, one city is designated as the disaster site. Given the above program inputs, the program output is the name (or identifier) of the city housing a Red Cross office that is closest (shortest total mileage) to the disaster city.

B.1 Categorization

Student Directed: Some, based on an open ended assessment rubric.
Scaffolding: None to some.
External Domain Knowledge: None.
Social Good: Medium.
Coolness: Small to medium.
Reflection: Medium.

B.2 CS Concepts

Graph representations, graph algorithms; Shortest path algorithms.

B.3 Implementation Strategies

B.3.1 The way it works

This is a traditional text-driven program. There are many ways to represent the inputs. One such straightforward method involves a simple text file input methodology:

- line 1: integer n representing the number of cities. Each city is represented by an integer in $[0..n-1]$
- line 2: integer m representing the number of edges in the graph.
- lines 3-$(m+2)$: three-tuples (i, j, k) representing an edge, one edge per input file line. i and j $(i \neq j)$ are integers in $[0..n-1]$ representing the edge's two endpoints. k is a positive integer representing the distance or mileage between the two *adjacent* cities.
- line $m+3$: an integer r representing the number of cities with a Red Cross office.
- The following r lines each contain a single integer i in $[0..n-1]$, indicating that city i houses a Red Cross office.

After processing the input graph and Red Cross office data, the program should interactively prompt the user for the location of the disaster; an integer in $[0..n-1]$. The output of the program should be the city identifier of the closest Red Cross office and the mileage between the disaster city and the closest Red Cross office.

B.3.2 It works better if

Students have a very firm understanding of a shortest path algorithm, say through interaction with an algorithm visualization tool or an in-class kinesthetic learning activity.

B.3.3 Assessment strategies

This project lends itself to being open-ended. In keeping with the proposal that project assignments should also convey the evaluation rubric[9, 77], one such rubric, implementing both the design paradigm of *iterative improvement* and open-ended creativity might be:

In order to earn a:

1. C: Students must have a correctly working program.
2. B: The requirements for a C in addition to excellent documentation. Furthermore, so as not to overwhelm any single Red Cross office, the program should output the two closest Red Cross offices to the disaster site and their respective distances to the disaster city.
3. A: The requirements for a B in addition to some other open-ended improvements. Here is the opportunity for students to think creatively. Some examples of such improvements are:
   - Instead of simply outputting the two closest Red Cross offices, the program should output the *c* closest Red Cross offices, where *c* is a positive integer interactively input by the user.
   - Encode (using letters) various Red Cross resources. e.g. "B" for beds, "T" for trailers, "W" for water, "F" for food, etc. For each Red Cross office, encode in the input file which resources that office stocks. Finally, for a given disaster, the user not only inputs the disaster location, but which resources are needed for that disaster. For each resource, the program outputs the closest Red Cross office (or two) which stocks that resource.

B.4 Extensions

See above.

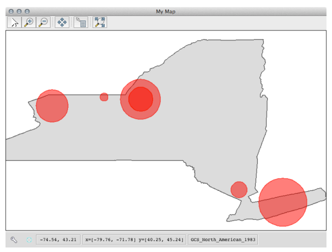
B.5 Deliverables

The program code plus a reflection paper on the following "location" problem: Given an input graph of cities and edges and a constraint of having only x Red Cross offices, where should they be located so that no city is *too far* from any Red Cross office; i.e. how to distributed limited resources so as to maximize coverage.

C. NUCLEAR POWER

JOHN BARR

This project considers the issues with the use of nuclear power. Students are required to analyze nuclear power plants in New York State and to write a technical report with their findings. To perform this analysis students must determine several factors that could be used to evaluate nuclear power, quantify these factors, write a program to produce a map that illustrates the factors, and then write a technical report with their findings.



C.1 Categorization

Student Directed: This project is completely specified. Students do have the freedom, however, to choose which factors to use in their analysis of nuclear power and how to quantify the factors. They can also determine how to design the map. For example, they could use a single map layer that shows a single risk factor number or they could create multiple layers, each a different color and each showing a different risk factor.

Scaffolding: A java class, `geoMap.java`, is supplied to allow students to create *.shp* files. A *.shp* file is a standard file type used in geographic information systems (GIS). It can be read by most major commercial and open source GIS programs. A shapefile for New York state is provided, but the `geoMap` class will read any shapefile so the instructor can easily use her own state/region in the project. The geoTools framework is used by the `geoMap.java` class to produce the *.shp* file. Instructors will have to prepare a project in either the Eclipse or Netbeans IDE with the geoTools framework. There are instructions for doing this at `http://www.geotools.org/` Students can follow the instructions to set up their IDE to use geoTools, but the project will flow more smoothly if the instructor sets up a project to work correctly and provides the project to the students.

External Domain Knowledge: Two areas of domain knowledge are covered in this project, geographic information systems (GIS) and nuclear power. The GIS knowledge required is minimal; if you like students can complete the project using only the supplied `geoMap.java` class with no knowledge of GIS. Some minimal knowledge of nuclear power risks is necessary. Students are required to obtain this knowledge independently and the required knowledge is minimal. All necessary information can be obtained, for example, from Wikipedia. In particular, students must obtain the location of all nuclear power plants in New York (available on Wikipedia), identify at least one risk factor (seismic activity for each nuclear power plant in New York is available on Wikipedia), and quantify this factor.

Social Good: The safety of Nuclear power, especially in the wake of the 2011 tsunami in Japan, is a critical national issue.

Coolness: Because of the publicity of the Fukushima nuclear accident in the wake of 2011, students both understand and are concerned with nuclear power plant safety.

Reflection: Students are required to submit a technical report justifying their choice of risk factors and analyzing the results. The report also requires students to reflect on the applicability of both the program and the map for the analysis of nuclear power.

C.2 CS Concepts

Single dimension arrays (including parallel arrays), reading values from a file, selection (if-then-else).

C.3 Implementation Strategies

C.3.1 The way it works

The solution strategy will employ the following steps:

1. Student locates power plant locations in New York and records their latitude and longitude in a text file. These locations are available on Wikipedia.
2. Student researches issues with nuclear power and chooses several. These might include such factors as distance to major population centers, amount of radiation that might be released in an accident, and the possibility of seismic activity at the plant.
3. Student quantifies each issue for each nuclear power plant. For example, Wikipedia articles on a particular New York power plants give the probability of an earthquake at that plant. A single number representing each site must be stored in the text file.
4. Once the student has the latitude/longitude and risk number (from the previous step) recorded in a text file, they can write a program to display the information. Their program will read the text file and store the latitude, longitude, and risk numbers in parallel 1D arrays, create a geoMap object, call the `makeMap()` method, and then call the `addBufferLayer` method (passing in the arrays).
5. Finally the student must write a technical report that 1. analyzes the use of nuclear power in New York and 2. discusses whether the program and map are effective analysis tools.

C.3.2 It works better if

This project uses the geoTools framework to access, create, and display GIS shapefiles. This is a very large framework that can be accessed dynamically using the *Maven* build tool. Though *Maven* is not hard to integrate with popular IDEs such as Eclipse or Netbeans, it can be confusing for students. The best approach to the project, then, is for the instructor to create a solution for the project, extract the main method, and then provide students with the Eclipse or Netbeans project. The complete project description provides details on using the *Maven* build tool within Eclipse or Netbeans or you can visit `http://www.geotools.org/` for instructions. Students can follow the instructions to set up their IDE to use geoTools, but this project will work better if the instructor sets up the initial project.

C.3.3 Assessment strategies

There are three deliverables for this project. The instructor can choose to weigh each of these differently depending on what they would like to emphasize. For example, the program can be heavily weighed to emphasize the computer science used or the technical report can be heavily weighed to develop critical thinking skills.

C.3.4 It doesn't work unless

Requires the `geoTools` frameworks together with either the Eclipse or Netbeans IDE.

C.4 Extensions

This project can easily be extended to include multiple layers of map information. A student could, for example, plot major population areas, areas of farm use, etc. The project is easily adapted to other domains. The project as written requires a technical report. The program, however, can be written independently, so the technical report can be omitted by the instructor.

C.5 Deliverables

Students must turn in a documented program, a screen dump of the map they create and a two-page technical report that analyzes the risk posed by the power plants in New York.

D. WATER POLLUTION

JOHN BARR

This project analyzes water pollution in three large New York rivers. Students must identify sources of pollution and amounts of pollution and also identify areas (such as towns, forests or habitats) that might be sensitive to pollution.

Students first locate three large rivers in New York and plot an approximation of their location on a map (as line segments). They then research the different types of pollution that commonly occur in New York, their sources and their effects.

After students have determined relevant factors, they quantify the factors and plot them on a map by creating "buffers" for each river (line). The greater the pollution on a particular segment of the river, the larger the buffer. For example, if there is a factory at one point that dumps pollutants in a river and downriver from that source there is another source of pollution, the buffer after the second source must be larger (or perhaps in a different color) to indicate that there are more pollutants. Students also use buffers to indicate sensitive areas that could be affected by pollution.

D.1 Categorization

Student Directed: This project is completely specified. Students do have the freedom, however, to choose which factors to use in their analysis of water pollution and how to quantify the factors. They can also determine how to design the map. For example, they could use a single map layer that shows a single risk factor number or they could create multiple layers, each a different color and each showing a different risk factor.

Scaffolding: A java class, `geoMap`, is supplied to allow students to create *.shp* files. A *.shp* file is a standard file type used in geographic information systems (GIS). It can be read by most major commercial and open source GIS projects. A shapefile for New York state is provided, but the `geoMap` class will read any shapefile so the instructor can easily use her own state/region in the project.

The geoTools framework is used by the geoMap class to produce the *.shp* file. Instructors will have to prepare a project in either the Eclipse or Netbeans IDE with the geoTools framework.

External Domain Knowledge: Two areas of domain knowledge are covered in this project, geographic information systems (GIS) and river pollution. The GIS knowledge required is minimal; if you like students can complete the project using only the supplied `geoMap` class. Some minimal knowledge of water pollution is necessary. Students are required to obtain this knowledge independently, however. In particular, students must identify at least one type of water pollution, and quantify the amount of this factor present in each of the chosen rivers.

Social Good: Pollution has long been a critical social issue in all parts of the world and its importance has increased over the past few decades with the industrialization of the world.

Coolness: Pollution is part of sustainability, an area of great interest to students.

Reflection: Students are required to submit a technical report justifying their choice of risk factors and analyzing the results. The report also requires students to reflect on the applicability of the program and map to the analysis of water pollution.

D.2 CS Concepts

Two dimension arrays, reading values from a file, selection (if-then-else), iteration (for or while).

D.3 Implementation Strategies

D.3.1 The way it works

The solution strategy will employ the following steps:

1. Students identify the locations of three large rivers in New York State and record their latitude and longitude in a text file. Each river consists of multiple straight-line line segments. The coordinate of the beginning point (latitude, longitude) of each of these segments must be recorded (the geoMap method assumes that the end point of each segment is the beginning point of the next segment). These locations can be found, for example, on google maps.
2. Students research sources of water pollution along the chosen rivers. These might include such factors as factory outflow, agricultural run off, and medical dumping.
3. Students quantify each issue for each river segment. A single number representing the amount of pollution for the segment must be stored in the text file.
4. Once the latitude/longitude and pollution number (from the previous step) have been recorded in a text file, students write a program to display the information. Their program will read the text file and store the latitude, longitude, and pollution numbers in a 2D array, create a geoMap object, call the `makeMap()` method, and then call the `addBufferLineLayer` method (passing in the array).
5. Finally the student must write a technical report that 1. analyzes the significance of pollution on the rivers and 2. discusses whether the program and map are effective analysis tools.

D.3.2 It works better if

This project uses the geoTools framework to access, create, and display GIS shapefiles. This is a very large framework that can be accessed dynamically using the *Maven* build tool. Though *Maven* is not hard to integrate with popular IDEs such as Eclipse or Netbeans, it can be confusing for students. The best approach to the project, then, is for the instructor to create a solution for the project, extract the main method, and then provide students with the Eclipse or Netbeans project. The complete project description provides details on using the Maven build tool within Eclipse or Netbeans or you can visit `http://www.geotools.org/` for instructions. Students can follow the instructions to set up their IDE to use geoTools, but this project will work better if the instructor sets up the initial project.

D.3.3 Assessment strategies

There are three deliverables for this project. The instructor can choose to weigh each of these differently depending on what they would like to emphasize. For example, the program can be heavily weighed to emphasize the computer science used or the technical report can be heavily weighed to develop critical thinking skills.

D.3.4 It doesn't work unless

Requires the `geoTools` frameworks together with either the Eclipse or Netbeans IDE.

D.4 Extensions

This project can easily be extended to include multiple layers of map information. A student could, for example, plot major population areas, areas of farm use, etc. The project is easily adapted to other domains. The project as written requires a technical report. The program, however, can be written independently, so the technical report can be omitted by the instructor.

D.5 Deliverables

Students must turn in a documented program, a screen dump of the map they create and a two-page technical report that analyzes the risk posed to the New York rivers by pollution.

E. AD HOC EMERGENCY WI-FI NETWORKING

RENZO DAVOLI

In case of catastrophic events (e.g. earthquakes, floods, volcanic eruptions) communication lines can be destroyed. The ability to communicate can save human lives.

Communication in emergency situations is both a need for the rescue teams and for the population. The cellphone services need a working infrastructure made of base radio stations to be operational. If the infrastructure collapses cell phones are useless. Earthquakes or floods can put base radio stations out of service: they are mounted on the top of high structures or buildings and they need an operational electricity grid. Even when the base radio stations remain operational the abnormal traffic can lead to an overload and block service.

The Internet can help to solve this situation. Whilst the cellphone service needs an infrastructure, modern smart phones have wi-fi interfaces onboard It is possible to use these interfaces to communicate by providing a very light infrastructure or even no infrastructure at all, in a peer-to-peer way.

The idea of this exercise comes from a project to create light inexpensive wi-fi radio stations. A number of these radio stations could be kept in storage, ready to be installed in disaster areas in case of need. These units should also be able to operate using solar panels and, most important, they should not need any configuration to be operative.

The users of this emergency network will not receive a full connection to the Internet, that could be misused, but just some low bandwidth services, e.g. text mail messages.

These services are powerful enough to provide the population with updated news from the emergency teams, and provide the population living in the damaged area with a means of communication to inform relatives and friends about the real situation. Obviously a lack of direct information would induce relatives and friends to reach the emergency area to check the situation. This extra traffic of vehicles on the roads can interfere with and delay emergency teams.

E.1 Categorization

Student Directed: None to some. The exercise is guided but open to extensions.
Scaffolding: None.
External Domain Knowledge: Some. The students should have an intuitive knowledge of the structure of some radio networks: e.g. wi-fi, cell phone and ham radio networks.
Social Good: Medium. These services can be really useful.
Coolness: Small to medium.
Reflection: High? Cellphones are perceived as a commodity. Students can have a more comprehensive view on radio services and their integration.

E.2 CS Concepts

Graph representations, graph algorithms; Shortest path algorithms.

E.3 Implementation Strategies

E.3.1 The way it works

This exercise can be implemented in many ways. The simplest one is to start an instance of a shortest path algorithm (e.g. Dijkstra's) for each pair of nodes and once each shortest path has been computed (or each set of shortest paths have been computed) it is possible to compute the maximum number of hops for that pair of nodes. It is also possible to rewrite the algorithm in order to compute for each node the set of first_hops of the minimum paths towards all the other nodes and the maximum number of hops in a single pass.

E.3.2 It works better if

Graph algorithms are sensitive to graph representations. Hence, properly designed data structures are vital to student success.

E.3.3 Assessment strategies

A working implementation should be granted a sufficient evaluation. Higher grades should take into account the coding style or efficiency concerns. Some extensions to the basic problem should be granted top grades.

E.4 Extensions

It is possible to add to the computation of one or more spanning trees for the delivery of broadcast packets. Obviously this exercise can be completed within a Networking course for a better understanding of bridging and routing, more specifically Ethernet packet switching and the link state routing algorithms.

E.5 Deliverables

Each student should provide two programs. The first program takes as input the network topology graph file. This file is an ASCII text file where each line represents a link and has three fields: the names of the connected nodes at the ends of the link and the weight of the link. e.g.

```
N1 N2 2
N2 N3 1
N1 N4 2
N4 N3 1
```

Note that the graph is undirected, so the first line of the file is a bidirectional link connecting N1 and N2 whose weight is 2. Furthermore, the file does not specify the set of nodes in a separate way, the set of nodes is the set of all the nodes whose names appear as an endpoint of a link.

The first program gives as output for each pair of nodes (A,B):

- The list of the next hops along the paths from A to B
- The maximum number of hops in the minimum paths.

e.g.

```
N1 to N2 nexthops N2 maxhops 1
N1 to N3 nexthops N2,N4 maxhops 2
N1 to N4 nexthops N4 maxhops 1
N2 to N1 nexthops N1 maxhops 1
N2 to N3 nexthops N3 machops 1
N2 to N4 nexthops N3 maxhops 2
...
```

The second program takes the output of the first program as its input. This program should provide an interactive interface where a user can enter source and destination nodes. The program should show the whole path a packet would follow. e.g.

```
input: N1 to N3
N1 sends the packet to N2 (maxhops 2)
N2 sends the packet to N3 (maxhops 1)
N3 was reached.
```

F. A MINI PACKAGE MANAGER FOR SOFTWARE DISTRIBUTION

RENZO DAVOLI

A software distribution is a collection of integrated, harmonic, continuously updated software tools.

Often this concept is not clear to students (at least at my longitude/latitude). Many students use libre software distributions but they do not realize the difference between operating systems and distributions.

Windows and MacOS are operating systems. Debian, Ubuntu OpenWRT, Mint, Fedora (and many others, see `distrowatch.org`) are distributions. These distributions are based on Operating Systems like GNU/Linux, GNU/Hurd, FreeBSD, NetBSD or others.

Many distributions have been built in a social way, and "for social good." Debian is a good example. Debian is the base on which Ubuntu and Mint have been constructed.

Libre, or free software can be freely used for any purpose, be studied, modified, or redistributed. Using libre software independent institutions and organizations together can create a collection of software tools, and can work to keep the whole collection updated, secure and consistent. In an analogy with literature, a distribution is an anthology.

F.1 Categorization

Student Directed: None to some. The exercise is guided but open to extensions.
Scaffolding: None.
External Domain Knowledge: None. The knowledge about software distribution and licenses should be part of the CS domain.
Social Good: Medium. This exercise teaches students how to be part of the development community.
Coolness: High? A distribution is a social way to provide cool code. Students can be empowered to be actors and not just spectators of the Software development world.
Reflection: Medium. From this example students can understand the real meaning of a distribution and decide to join the community of a software distribution: A social way to be a computer scientist.

F.2 CS Concepts

Graph representations, graph traversal algorithms and topological sorting.

F.3 Implementation Strategies

F.3.1 The way it works

Apart from the standard graph visiting/traversal algorithm (for package install/deinstall) and topological sorting (for distribution update) students need to add a reference count to each packet to delete dependent packages when they are no longer needed. During a distribution update, students must pay attention to update these values correctly as the list of dependencies may have changed.

F.3.2 It works better if

Graph algorithms are sensitive to graph representations. Hence, properly designed data structures are vital to student success, particularly when students elect a recursive implementation.

F.3.3 Assessment strategies

Students submissions should be considered sufficient if the problem has been solved, good if the solution is well structured and the data structures have been properly designed, and excellent when the students add some extra features.

F.4 Extensions

It is possible to show students that the real problems related to package management are very complex. There are NP-complete problems concerned with package management. There is a wonderful blog entry[3] which shows how it is possible to create a set of package dependencies that are the problem equivalent of solving a Sudoku puzzle. (Problem reduction!) Students may study more complete implementations of package managers and add further features to their exercise.

[3] http://algebraicthunk.net/~dburrows/blog/entry/package-management-sudoku/

F.5 Deliverables

The goal of this project is to write a mini-package manager. The package database, for the scope of this exercise, is an ASCII file. Each line corresponds to a package, the first two fields are the name of the package and its version. The following fields are pairwise the name of each dependent packages and the minimum version required. e.g.

```
P1 10
P2 3 P1 9
...
```

The version 10 of package P1 is available, as well as the version 3 of P2. P2 requires P1 to be installed, and the minimum version of P1 required by P2 to work properly is 9. So in this case P2 could be installed as P1 version 10 (> 9) is available.

The database of installed packages has the same structure of the available package database.

phase1: packages install. Input: a distribution database, a database of installed packages, and a set of packages names. Output: the updated database of installed packages, and a log containing the list of newly installed packages as well as the list of the packages that could not be installed as some of their dependencies were missing. It is possible to re-install already installed packages: if a newer version is available the package gets updated otherwise the package manager must add a warning message on the log file.

phase2: packages deinstall. Input: a distribution database, a database of installed packages, and a set of packages names. Output: the updated database of installed packages, and a log containing the packages deleted. Note that the deletion of a package can cause the deletion of its dependent packages if they are not needed by other packages. The log should report errors such as requests to delete an uninstalled package.

phase3: distribution update. Input: the new distribution database, a database of installed packages. Output: the updated database of installed packages, the list of the packages that can be updated. A package cannot be updated if it depends upon another package which does not exist or whose version number is older than the minimum version.

G. VOTING SIMULATION

ELIZABETH PATITSAS

In this assignment, we simulate Canadian elections under different voting systems. The assignment gives students practice with lists and dictionaries in Python, and in the process, exposure to different voting systems and how they would impact a democracy.

Canada uses a plurality-based voting scheme; there have been recent efforts for voting reform. A motive of this assignment is to inform our students as citizens in our referenda on voting reform. Another motive is for students to realize that algorithms play a role in democracy.

We provide the students with randomly-generated ballots based on recent polling data. Students complete the ballot-tallying methods for six methods, and for a helper method (finding the argmax of a list).

G.1 Categorization

Student Directed: None.
Scaffolding: High.
External Domain Knowledge: Some. We provide the ballots to the students, and handle the output and test cases for the students. Some of the methods at the beginning contain scaffolding, such as the setup of the loop over the list, so that students need only fill in the body of the loop. We provide the method for plurality as a worked example for the students.
Social Good: Medium. For Canadians, there is very little external domain knowledge necessary. We provide information on how each of the different voting systems works. We also provide

a brief description of Canadian politics for the international students. The details one needs to know are the names of the four major federal parties (Conservative, Liberal, New Democratic, and Green), that constituencies are known as "ridings" in Canadian English, and there are 308 ridings, each of which elects a single Member of Parliament (MP). The party with the most MPs forms the government.
Coolness: Medium.
Reflection: Medium.

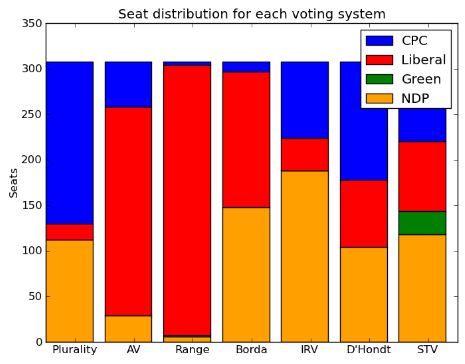


Figure 1: Simulated election results under different voting results; this shows the students how the choice of algorithm can affect the outcome!

G.2 CS concepts covered

Lists, dictionaries (hashing), iteration, selection, and seeding random number generation.

G.3 Implementation strategies

G.3.1 The way it works

We give students code that runs the elections; they fill in the code that gets the ballots counted. Students implement one voting method at a time. We provide test cases for each, so students can test their work as they go. We have organized the voting methods in order of complexity.

G.3.2 It works better if

Students have access to matplotlib. We provide a visualization of the results using matplotlib; without it there is only line output for how many MPs each party has elected. Implementing the proportional representation methods is much easier with a decent debugger. Going over an example of voting systems in a tutorial would also be helpful for students.

G.3.3 Assessment strategies

In the assignment we provide test cases; we plan to run student code with an auto-marker and then have TAs look at code style. The assignment also contains a short essay question for TAs to mark; we plan to mark liberally on completion, reasoning and clarity. The essay is to have students reflect on the assignment, not to be a definitive written work.

G.3.4 It doesn't work unless

Students need to be comfortable with lists (or dictionaries) and iteration. Currently this project is implemented in Python.

G.4 Extensions

The assignment can be easily extended to have students implement other voting systems. At present, it includes plurality, approval voting, range voting, the Borda Count, Instant Run-Off Voting, the D'Hondt Method for proportional representation, and the Single Transferable Vote.

Other voting systems that would be suitable include the Sainte-Laguë method for proportional representation, Mixed-Member Representation, Condorcet-Schulze, majority judgment, and Dodgson's method (created by Charles Dodgson, aka CS Lewis.)

With more effort, this could be redone for elections in other countries. It would be easiest for other parliamentary systems like the UK or Australia.

G.5 Deliverables

Students hand in a single file of code containing their methods. We plan to set this up to allow for auto-marking.

G.6 Acknowledgements

Michelle Craig trialed out the assignment and provided detailed feedback. A number of anonymous Redditors on `r/CanadaPolitics` assisted in designing the ballot generation code which is provided to the students.

H. STI MODELLING

ELIZABETH PATITSAS

In this assignment students model STI through sexual network graphs. The assignment covers graph traversal and modification, and guided refinement of disease modeling. It also provides an example of how computer modeling is useful in public health, and hopefully motivates students to use safer practices when having sex.

First, students model chlamydia – a disease where people are either susceptible to the disease, or infected. We then add complexity by modeling the common HPV strains, for which there are vaccines. Next, we add complexity by adding death: modeling HIV/AIDS.

Along the way we examine matters such as how many people in a population need to be vaccinated to contain a disease, reducing rates of transmission, and the epidemiological effects of delaying death to HIV/AIDS.

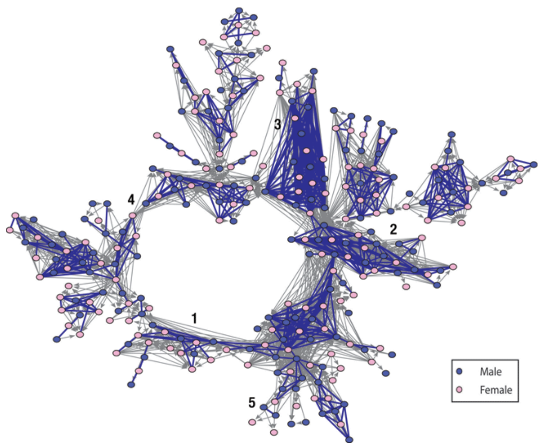


Figure 2: An example of a sexual network – a high school's sexual network of the course of 18 months, per Bearman

H.1 Categorization

Student Directed: None.
Scaffolding: Some. Students are directed what to complete, and given one example of expected output. Students are not provided with starter code.
External Domain Knowledge: None to some. All needed external domain knowledge for the students is provided in the handout. Instructors may want to spend some time reading about SIS and SIR models on Wikipedia (or elsewhere).
Social Good: Some.

Coolness: High.
Reflection: Some.

H.2 CS concepts covered

Graph theory, object-oriented programming, file I/O.

H.3 Implementation strategies

H.3.1 The way it works

Three files are provided to the students, containing adjacency matrices of three different sexual networks. The files vary in size – a small population of size four, a medium one of size 40, and a larger one of size 400.

In the instructions, the expected output for the small population is provided and visualized. Students are expected to test their code on the other two populations.

H.3.2 It works better if

Students have a good way of visualizing graphs. Printing out adjacency matrices works, but tend to require you drawing out the visual graph.

The assignments requires the students to write or use a graph class. Having them do this beforehand in lab will reduce the overhead on the assignment.

H.3.3 Assessment strategies

Students should be marked on their ability to generate graphs, traverse them, and modify them. They should also be marked on their report: its clarity, whether the open-ended portion demonstrates technical knowledge of graph theory or OO design, whether the open-ended portion has scientific merit, and whether the students have reflected on the use of modeling.

H.3.4 It doesn't work unless

This assignment is doable in any OO language, although it was designed for either Python or C++. Students will struggle with the open-ended nature of the project, so providing examples, scaffolding, and reassurance at the beginning can go a long way.

H.4 Extensions

A possible extension would be to have multiple diseases propagating through a network, while various treatments are also underway. This could be used to bridge into a discussion of the merits of the SIR model of disease transmission (which uses differential equations to describe population-level changes, rather than simulating the whole population at a given time.)

Adding transmission through needle sharing – for the purposes of modeling HIV/AIDS – would also be possible – as would having the students simulate the effect of having infected people die in the network. For HIV/AIDS (and Herpes), simulating viral load would also be interesting.

Incorporating inheritance and polymorphism can also be done by having multiple classes of people in the network, particularly for simulating high risk populations (eg. class FemaleProstitute inherits from Woman inherits from Person).

H.5 Deliverables

Students hand in a report of their modeling, and their code.

H.6 Acknowledgements

Cathy Meyer verified the epidemiology in this assignment. The assignment is based on Mark Maclean's first-year calculus project on SIR/SIS modelling with differential equations.

I. BANANA REPUBLIC

TONY CLEAR

This staged set of programming exercises is geared for an objects early CS1 course in Java, supported by the Barnes & Kölling text "Objects First with Java: A Practical Introduction Using BlueJ"[8]. The three exercises move from the introductory stage (week 2) addressing variables, assignment, operators, simple methods and sequence, to the more advanced exercises (week 9) where students create classes, simulate population movements, crop growth, harvesting and fiesta cycles, immigration and boat movements between islands. It requires the use of further techniques such as iteration, selection, arrays, string handling, object creation and removal, parameter handling, printing. It is intended as a motivating set of activities in a context in which students can experience success in programming in a staged manner, while observing the movements of their citizens, the cycles of the seasons and the dynamics of their own Banana Republic evolve.

I.1 Categorization

Student Directed: Some. Students are required to actively engage in completion of the programming exercises. Exercise A and B have optional challenge exercises, and exercise C allows students the freedom to add a method and variables to the BananaRepublic class "that allows the republic to do something interesting of your own invention."

Scaffolding: Medium. The exercises are carefully staged, and supported by the textbook, lecture sessions and closed labs in which exercises of a similar style and complexity are completed on a formative basis. Pre-written clean code with some predefined classes is provided for each exercise set, or students may continue with their own code developed from the earlier exercises.

External Domain Knowledge: Some. Domain knowledge, while specific to the setting is not overwhelming or especially complex, and has been tailored to a culturally diverse student body with variable English language proficiency. For instance Classes and concepts involved include - Banana Republic, El Presidente, revolution, currency, farmers, bananas, harvest, crops, crop reserve, GDP, currency, satisfactionRating, islands, boats, cargo, and fiesta.

Social Good: Some or Medium. The assignment sets computing in a context where relatively straightforward computational activities and object oriented development with a small set of classes, enable the creation of a simulated island state wherein a set of normal and chaotic events take place. While being intended as a slightly tongue-in-cheek set of exercises to maintain student engagement, it also has the potential to raise wider questions about the environmental, political and social challenges facing small island states such as those found in the Pacific or the Caribbean.

Coolness: Medium. The chosen scenario appeared to be engaging. Students seemed to find the assignments interesting, and enjoyed creating the required conditions on their respective islands.

Reflection: Some. The assignments as framed require no specific reflection on the part of students, although students have been expected to maintain a reflective journal and programmer's log of their activities during the CS1 and CS2 courses. These in some iterations have been required to be handed in with their assignment submissions, and help give instructors some indication of where students are spending their efforts, having difficulties, and can be used as evidence of own contributions when a plagiarism issue might arise.

I.2 CS Concepts

A range of basic concepts such as variables, assignment, operators, simple methods and sequence, to more advanced CS1 concepts such as classes, random simulations, and movement and cyclical scenarios requiring further techniques such as iteration, selection, arrays, string handling, object creation and removal, parameter handling, printing.

I.3 Implementation Strategies

I.3.1 How it works

The sets of BananaRepublic exercises have been designed around the availability of a good supporting text book, and an "objects early" pedagogy [55]. The staged exercises address the concepts progressively exposed in the text and covered in the accompanying lecture program. The challenge is to identify an engaging context and develop a set of exercises which progressively develop the core CS concepts while building a more meaningful world within the simulation, which demonstrates the power of programming as a way to innovate and create new conceptual and meaningful worlds. Thereby even an introductory programming course can demonstrate the potential of programming to relate to significant challenges in the students' world.

I.3.2 It works better if

The current closed lab formative exercises have tended to have a collaborative element, which sometimes leaks across into the individual assignments. Emphasising which activities are collaborative in nature and which are individual needs some care. If there is a strong desire to emphasise the social good element of the assignment, then a useful addition would be a more extended reflective assessment in which students were required to investigate a social challenge facing the context or how an extension component might be implemented to represent the chosen challenge scenario.

I.3.3 Assessment strategies

These three assignments comprised 50 per cent of the summative assessment for the course (10% for the first assignment, 20% each for the latter two). Complementing these are a set of formative closed laboratory exercises. The remainder of the summative assessment comprises 10% allocated to fortnightly in class quizzes and 40% to a final exam. The assessments are framed as individual but could equally be expanded to accommodate pair programming or small group project work.

I.3.4 It doesn't work unless

The concept needs to have been well thought through and the progression of the exercises concurrent with student skill building needs some care. For instance the first exercise has the support of pre-built classes, which students can use, amend and extend in well defined steps. As a progressive assignment, the provision of a clean starting point (a fresh set of pre-built classes) for each set of exercises enables those students, who have been struggling with parts of each exercise set, to avoid stalling. At the same time those who have done well can continue to build on their own code bases at each stage, which builds confidence and retains ownership.

I.4 Extensions

These assignments inherently have some extension features, but they could be expanded in many ways. For instance students could be asked to add extension classes to expand the range of potential scenarios for the Banana Republic (e.g. natural disaster strikes - hurricane or tsunami; more gradual impacts on quality of life - global warming and sealevel rise; the impact of urbanisation as populations move in from the countryside; famine, fire, epidemics, health challenges - diabetes, obesity or civil unrest etc.). Alternatively from a more technical perspective students could be asked to develop unit tests for their code.

I.5 Deliverables

The output of these assignments are zip files of working Java code projects, including some incorporated javadoc comments, code printouts and accompanying MS Word documents to evidence output screen shots.

I.6 Acknowledgements

We gratefully acknowledge the contributions of colleagues Dr Jacqueline Whalley and Dr Yun Sing Koh of Auckland University of Technology in their conception and design of the above exercises, and for their agreement to make them available for the working group.

J. KIWIS

TONY CLEAR

This staged set of programming exercises is geared for an objects early CS1 course in Java, supported by the Barnes & Kölling text "Objects First with Java: A Practical Introduction Using BlueJ"[8]. The five exercises move from the introductory stage (week 2) addressing object creation and method calling, to the more advanced exercises (week 10) by which time students may create classes and methods, simulate egg laying, food consumption and exercise activities, burrowing, population expansion and feed distribution strategies. It requires the use of further techniques such as iteration, selection, arrays and arraylists, string handling, object creation and removal, parameter handling, printing. It is intended as a motivating set of activities in a context in which students can experience success in programming in a staged manner, while observing the activities of their kiwis while tending to their food, shelter, procreation and safety needs and observing the dynamics of their own kiwi population evolve.

J.1 Categorization

Student Directed: Some. Students are required to actively engage in completion of the programming exercises. Exercise C and D have optional challenge exercises, and exercise D allows students the freedom to modify their simulation "so that kiwis are fed using a central feeding box in the kiwi house."
Scaffolding: Medium. The exercises are carefully staged, and supported by the textbook, lecture sessions and closed labs in which exercises of a similar style and complexity are completed on a formative basis. Pre written clean code with some predefined classes is provided for each exercise set, or students may continue with their own code developed from the earlier exercises.
External Domain Knowledge: Some. Domain knowledge, while specific to the setting is not overwhelming or especially complex, and while geared to a New Zealand context has also been tailored to a culturally diverse student body with variable English language proficiency. For instance Classes and concepts involved include - Kiwis, Kiwi House, burrow, age, weight, eggs, hatching, food, insects, exercise, escape routes, and feeding box.
Social Good: Some or Medium. The assignment sets computing in a context where relatively straightforward computational activities and object oriented development with a small set of classes, enable the creation of a simulated kiwi colony wherein a set of typical kiwi life events take place. While being intended as a fun set of exercises to maintain student engagement, it also has the potential to raise wider questions about the environmental challenges facing endangered species, and issues related to diet, habitat and predation.
Coolness: Medium. The chosen scenario appeared to be engaging. Students related to the conservation theme and the iconic nature of the kiwi as both New Zealand national bird and endangered species. They seemed to find the assignments interesting, and enjoyed simulating the activities and natural cycles of kiwi life in their own kiwi house.
Reflection: Some. The assignments as framed require no specific reflection on the part of students, although students have been expected to maintain a reflective journal and programmer's log of their activities during the CS1 and CS2 courses. These in some iterations have been required to be handed in with their assignment submissions, and help give instructors some indication of where students are spending their efforts, having difficulties, and can be used as evidence of own contributions when a plagiarism issue might arise.

J.2 CS Concepts

A range of basic concepts such as object creation and method calling, variables, sequence, operators, to more advanced CS1 concepts such as classes, random simulations, and movement and cyclical scenarios requiring further techniques such as iteration, selection, arrays and arraylists, string handling, object creation and removal, parameter handling, printing.

J.3 Implementation Strategies

J.3.1 How it works

The sets of Kiwi exercises have been designed around the availability of a good supporting text book, and an "objects early" pedagogy [55]. The staged exercises address the concepts progressively exposed in the text and covered in the accompanying lecture program. The challenge is to identify an engaging context and develop a set of exercises which progressively develop the core CS concepts while building a more meaningful world within the simulation, which demonstrates the power of programming as a way to innovate and create new conceptual and meaningful worlds. Thereby even an introductory programming course can demonstrate the potential of programming to relate to significant challenges in the students' world.

J.3.2 It works better if

The current closed lab formative exercises have tended to have a collaborative element, which sometimes leaks across into the individual assignments. Emphasising which activities are collaborative in nature and which are individual needs some care. If there is a strong desire to emphasise the social good element of the assignment, then a useful addition would be a more extended reflective assessment in which students were required to investigate a social challenge facing the context or how an extension component might be implemented to represent the chosen challenge scenario.

J.3.3 Assessment strategies

These five assignments comprised 50 per cent of the summative assessment for the course (10% for each assignment). Complementing these are a set of formative closed laboratory exercises. The remainder of the summative assessment comprises 10% allocated to fortnightly in class quizzes and 40% to a final exam. The assessments are framed as individual but could equally be expanded to accommodate pair programming or small group project work.

J.3.4 It doesn't work unless

The concept needs to have been well thought through and the progression of the exercises concurrent with student skill building needs some care. For instance the first exercise has the support of pre-built classes, which students can use, amend and extend in well defined steps. As a progressive assignment, the provision of a clean starting point is required for each set of exercises. In this version of the assignment set, students are expected to have their prior code exercises signed off by a Teaching Assistant before they can continue to build on these code bases at each stage, which is intended to build confidence and retain ownership.

J.4 Extensions

These assignments inherently have some extension features, but they could be expanded in many ways. For instance students could be asked to add extension classes to expand the range of potential scenarios for the Kiwis. For example, students might model predator attacks by dogs, rodents or mustelids; forest fire; the impact of habitat loss as urban populations encroach; pesticide impacts on insect populations and food supply, epidemics, etc. While the assignments focus on the "Kiwi" as the New Zealand native bird, it would be simple to replace the kiwis by other endangered species, golden eagles, polar bears, orang-utans, etc. Alternatively from a more technical perspective students could be asked to develop unit tests for their code.

J.5 Deliverables

The output of these assignments are zip files of working Java code projects, including some incorporated javadoc comments, code printouts and accompanying MS Word documents to evidence output screen shots.

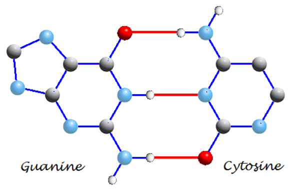
J.6 Acknowledgements

We gratefully acknowledge the contributions of colleague Dr Jacqueline Whalley of Auckland University of Technology in her conception and design of the above exercises, and for her agreement to make them available for the working group.

K. MOLECULAR MODELING AND DNA

SCOTT PORTNOFF

Students build a 2-D molecular modeling program to examine the hydrogen bonding between purine and pyrimidine bases that holds the two anti-parallel strands of the DNA double helix together.



K.1 Categorization

Student Directed: Medium. Guided Discovery. Instructor leads whole class activity in the building of the program. Students write/complete methods needed to solve particular sub-problems regarding CS concepts or program logic.
Scaffolding: High. Guided Discovery.
External Domain Knowledge: High. Software Engineering pedagogy. Students use: regular polygon geometry to calculate coordinates of hexagon vertices; sin/cos to calculate relative polar coordinates to position pentagon and functional group atoms; the additive sin/cos formulas for computer graphics rotation; the chemistry of polar covalent bonds and hydrogen bonds.
Social Good: High. Students use CS to study the most important feature of the structure of DNA and view the film **Double Helix** to consider the complex interpersonal dynamics of the four biophysicists who contributed to the solution. The discovery of the double helix revolutionized the field of genetics, made possible the **Human Genome Project** and a new CS discipline **Bioinformatics**, and led to advancements in medicine and public health.
Coolness: High. The unit encourages student engagement at several programming "obstacle" points.
Reflection: Medium. Students are assessed on their understanding of programming concepts, and a reflection of the interpersonal dynamics between the film's characters.

K.2 CS Concepts

Software Engineering, 2-D Graphics Transformations (Translation, Rotation, Mirroring), Inheritance, Polymorphism, GUI, Nested Loops.

K.3 Implementation Strategies

K.3.1 The way it works

The unit begins with students familiarizing themselves with the freely available 3-D molecular modeling program **MolSoft ICM-Browser**, and exploring ways to configure the four DNA bases Adenosine, Guanine, Cytosine and Thymine within the program. The molecule files will be downloaded from the **NYU Library of 3-D Molecular Structures**. Students then proceed to calculate the angles of the pyrimidine (hexagon) and imidazole (pentagon) rings and use **BYOB**[4] to correctly position each base's ring and functional group atoms. During this process, students design their program to reflect the biochemical nomenclature of the molecule's major features. They also abstract shared features of the molecules into common methods for building pyrimidine rings, imidazole rings, and adding functional groups at any of the 6 pyrimidine atoms.

Once they are familiar with the structure of the 4 bases, students go about building the 2-D molecular modeling program in **Processing**. They use the sine and cosine functions to create a method to position atoms using polar coordinates. They use getter methods to encapsulate the x- and y-coordinates of each atom. These methods become the central repository for calculating translated, rotated, and mirrored coordinates for each atom. Students use a geometry proof to find the additive angle formulas for sine and cosine. These are used to derive the rotation formulas for x- and y-coordinates, which are then implemented in the program.

Students then study the chemistry of polar bonds and hydrogen bonds. They write methods for deciding which hydrogen atoms are **electropositive** and which nitrogen and oxygen atoms are

[4]Build Your Own Blocks: `http://byob.berkeley.edu/`

electronegative. The optimal distance for intermolecular hydrogen bonds is indicated using color and line thickness.

Students program the GUI for object selection with mouse, control key and lassoing. Move, rotate, and mirror-image actions are driven by mouse events.

At unit's end students *use their programs* to display normal A-T and G-C pairings. They also perform **predictive** tasks, i.e. find configurations for rare A-C and G-T pairings, which represent point mutation situations.

To anchor this project in a social setting, students study the **BBC** film **Double Helix**, which relates the little known story of the discovery of the DNA double helix by Watson and Crick, who used the x-ray diffraction data of the biophysicist Rosalind Franklin for building their model. Although her data was crucial to their calculations, which won them the Nobel Prize, they did not acknowledge her contribution until long after her death.

K.3.2 It works better if

The unit has been taught twice, and would work better if small exercises were written to give students practice in applying the major programming concepts.

K.3.3 Assessment strategies

The unit grade is based on (1) the final program code, (2) the film essay, and (3) a multiple choice exam that covers the major programming concepts.

K.3.4 It doesn't work unless

This project has a warmup exercise that uses Scratch/BYOB. The main project is based on the `Processing` language. Use of version 2.6a or greater is recommended.

K.4 Extensions

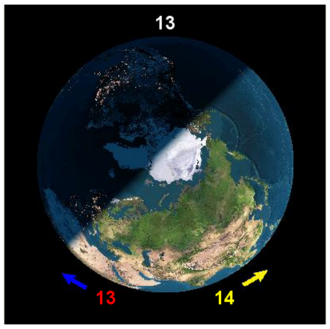
The program can be automated to find all possible purine-pyrimidine pairings with at least 2 H-bonds. Criteria to handle the best of duplicate pairings is an open-ended task. Automation should test pairings with regards to rotation angles, mirror images, and intermolecular distances.

K.5 Deliverables

Student must (1) turn in a complete working program, and (2) write an essay on the film in response to a prompt.

L. AROUND THE WORLD IN 24 DAYS

SCOTT PORTNOFF



Students build a simulation of a rotating Earth in order to model the phenomenon described in Jules Verne's **Around the World in 80 Days** of an east-bound traveler who circumnavigates the world and experiences one day more than an observer remaining at the starting point.

Three observers are placed on the surface, two of whom circumnavigate the globe in opposite directions: (a) an East-bound traveler (yellow); (b) a West-bound traveler (red); and (c) a stationary observer (white). After 24 days, the two travelers return to the starting point to rejoin the stationary observer. The east-bound traveler will have seen 25 sunrises, and the west-bound traveler will have seen 23 sunrises. The simulation illustrates the reason for the establishment of the *International Date Line*.

L.1 Categorization

Student Directed: Medium. Guided Discovery. Instructor leads whole class activity in the building of the program. Students write/complete methods needed to solve particular sub-problems re: CS concepts or program logic.
Scaffolding: High. Guided Discovery.
External Domain Knowledge: Medium. Software Engineering pedagogy. Students use sin/cos to calculate polar coordinate positions on the circle representing the Earth's surface. History: Students study the political and religious considerations of the country-by-country adoption of the Gregorian calendar and the later establishment of the International Date Line.
Social Good: High. Human societies have always used natural events to mark time, e.g. the new moon to mark months, the sun's highest position to measure a solar year. The leap year calculation based on the Gregorian calendar has been a traditional staple of CS1 courses. Students with family roots in Asia are well-familiar with having to adjust their watches forwards or backwards an entire day when traveling across the Pacific. This unit simulates the full-day discrepancies that arise when travelers circumnavigate the globe, a problem resolved by the adoption of the International Date Line time standard.
Coolness: High. The unit encourages student engagement at several programming "obstacle" points.
Reflection: Medium. Students are assessed on their understanding of programming concepts as utilized to implement the logic for this program.

L.2 CS Concepts

Software Engineering, Classes, Event Handling, Graphics, Mod Function, Conditional Expressions, Discrete vs. Continuous models.

L.3 Implementation Strategies

L.3.1 The way it works

Students download 96 satellite images of Earth using the **View from Earth** website (`http://www.fourmilab.ch/cgi-bin/Earth`). These represent snapshots taken over a 24-hour period spaced at 15-minute intervals. So that the surface of the Earth is half in shadow, the date chosen is either the Spring or Autumnal Equinox with Latitude = 90°N as if the satellite is positioned over the North Pole. Longitude is arbitrary, but we choose 72°E so that Los Angeles is at the top of the simulation.

Students load the images into an array and implement the animation using a circular queue, which displays a stationary Earth with a moving *terminator* (the boundary line separating day and night). Following an investigation of Processing's 2-D transformation operations, students implement rotation by translating the coordinate system origin to the center of the window, perform the rotation, then translate the origin back to the top left corner. Students investigate the use of bracketing transformations between **pushMatrix** and **popMatrix** to independently rotate several objects simultaneously. The final rotation effect is that the Earth rotates and the **terminator** is stationary. A toggle variable controls whether the animation rotates.

Earth, **Sunrise** and **Traveler** classes are implemented. A conditional expression to enable a stationary traveler to detect a sunrise begins with normalization of degree measurements to keep traveler and sunrise angles within the same range. The conditional expression is modified as more cases are accommodated, culminating with solving the edge condition at 0°/360°. The final expression implements a **sector-point** intersection model.

Movement for travelers is implemented using a **speed** instance variable which is either positive for traveling West, negative for traveling East, or 0 for no movement. Students discover that sunrise detection breaks down for moving travelers: at some point during their circumnavigations - depending upon starting values for sunrise and traveler - the East traveler misses a sunrise and the West traveler clocks a double sunrise. The analogy is to an escaping prisoner avoiding detection by a moving flashing searchlight. The problem is solved by narrowing or expanding the **sector** by the traveler's speed, and students consider the issue of representing a **continuous** system by using a **discrete** model.

L.3.2 It works better if

The unit has been taught twice, and would work better if small exercises were written to give students practice in applying the major programming concepts.

L.3.3 Assessment strategies

The unit grade is based on the final program code and a multiple choice exam on major programming concepts.

L.3.4 It doesn't work unless

This project is based on the `Processing` language. Use of version 2.6a or greater is recommended.

L.4 Extensions

None.

L.5 Deliverables

Student must turn in a complete working program.

M. SOCIAL GOOD WEBSITE

SAMUEL MANN

A Web 1 course aims to acquaint students with the range of available web-based tools for productivity, entertainment, and communication. It is intended to guide students toward consideration of the social, academic, economic and cultural issues surrounding web-based interaction. There is also an introduction to the technologies available for development of web-based functionality. This course is designed for students with only a single semester of programming experience.

In groups of three, students take on the task of developing an integrated web presence for a community organisation or development initiative. This web presence must address functional requirements, experience design and aesthetic design requirements. It must be dynamic for the user experience (but not require server-side processing), and incorporate multiple channels. The project must address an area of social need. There are several sub-assignments, primarily the delivery of an website written in static HTML, a porting to a content management system, integrating social media, and (for bonus marks) system deployment.

Example social good topics are given such as "A website celebrating the heritage of your suburb/hometown" or "A website where you document every bit of rubbish you pick on up your walk home." The groups workshop an area that interests them, consider the driver, problem, and likely impact of the solution. It is not required that students have an actual person as client for this project. We do need at least a conceptual "client" i.e. who is the sponsor for this project?

M.1 Categorization

Student Directed: Medium. Given the task of creating a website for social good using a content management system integrating social media, the students determine what they learn
Scaffolding: A development process is provided.
External Domain Knowledge: Some. The students sometimes select an area of social good about which the instructor knows little. There are some areas where the instructor does need to ensure student safety - for example when they choose to develop a website on depression or sexual abuse.
Social Good: High. The projects are all explicitly social good. Some projects reach actual deployment. As an example in 2011, students developed EducatingCambodia.com to support a school and community building initiative in the poorest region of Cambodia. In 2012 students developed PortWireless.co.nz to provide free wifi to a port town in New Zealand. This project included extensive community consultation and development of business models. Other projects have included coastal erosion education, pet adoption, community storytelling, and resource sharing.
Coolness: Some. Students enjoy the ability to work on a project of their choosing. Some struggle to find an area of tractable social good getting initially stuck on "cure cancer" or "solve world hunger" but the class workshops potential impacts stemming from their project and students get quite excited about the potential to make a real change.
Reflection: Medium. Students are required keep a development log, to present progress weekly and to reflect on their learning – including on the role of computing in their selected area of social need. This is all done on a public wiki and students are required to comment on colleague's work (at least weekly). Furthermore, they must submit a technical report justifying their choice of risk factors and an analysis of the results.

M.2 CS Concepts

HTML, social media

M.3 Implementation Strategies

M.3.1 The way it works

Students work in small groups to develop social good websites.

M.3.2 It works better if

Nominally at least one class per week is dedicated to the project with the other class providing skills required for the project. The learning objectives and schedule for paper are presented on the wiki with clear indication that they are flexible and negotiable according to the direction of their project.

The main trick is helping students find a sizable yet tractable project. Some students, given the option first select a small or joke project. Such projects are difficult to maintain interest. At the other end of the scale, projects are avoided that require significant server side processing.

M.3.3 Assessment strategies

Assessment is by ongoing peer and self assessment via a wiki. The marking schedule is negotiated according to the direction taken by the group but with a minimum of 20% for reflection.

M.4 Extensions

In the last week students are asked "what extra thing could be done that would really add value to this project?" This is then used as an extra learning outcome for each group (i.e. they have to do that extra thing). The EducatingCambodia group, for example, needed to integrate a payment system and integration into charity accreditation.

Students can carry this initial project through to their capstone project. Some groups have double dipped, using their project to contribute credits in other courses. The PortWireless group, for example, set up a company for their project, getting credit for this in a business class.

M.5 Deliverables

Students must turn in a working website and evidence portfolio of their development process.

N. SOCIAL GOOD SCHOLARLY WORK

SAMUEL MANN

Students complete a formal work in the area of the social implications of an aspect of communications technology.

N.1 Categorization

Student Directed: High. Both the question and the form of this work are individually negotiated between the student and instructor.
Scaffolding: Some. Some students require guidance in ensuring the scholarly aspects of non-traditional formats.
External Domain Knowledge: None to medium.
Social Good: Medium. The scholarly works are all explicitly social good. Some have "real" benefits. Examples from 2011 include:

- A report on an experiment into how modern media affects on the importance of physical beauty in self image.
- A video documentary on the online personas of body builders.
- A radio documentary on internet addiction.
- A radio documentary on community building through wireless networks.
- A video documentary on the relationship between animal welfare and computing in the dairy industry.

- A class seminar on the social implications of online gaming.
- A class seminar on the negative effects of social networking.
- A book aimed at engaging 12 year old girls in computing.

Coolness: Medium. Once students get over the shock, the coolness factor is very high. Students enjoy the freedoms to work on an area of their choosing and the braver ones relish the challenge of exploring this in song or dance.
Reflection: High. The purpose of the assignment is to encourage students to deeply engage in the nature of computing as a profession.

N.2 CS Concepts

Social implications of computing.

N.3 Implementation Strategies

N.3.1 The way it works

Students write a scholarly work. This may take the form of a formal essay, research article, Wikipedia contribution, radio feature or any form that the student can justify as being scholarly.

The task meets the requirement for critical thinking and to meet an explicit requirement for consideration of social implications of computing.

Here's the brief:
You will write a short proposal that includes 200 words on your thoughts on each of four peer reviewed papers related to your topic area. The proposal must finish with an outline of your essay. Students can present the work in any format that:

1. Supports the development of an argument with justification and evidence based examples. In the proposal, students have to justify how this form provides a vehicle for creative and evidence based development of an argument.
2. Supports citation of at least four peer reviewed articles. APA6th referencing must be used (although appropriate to the form should be used, deviations from APA must be agreed beforehand).
3. Can be submitted via your wiki, webpage or blog (i.e. if you perform a song, you'll need to video it).

N.3.2 It works better if

Students can get quite a shock that not only the question but also the form of the work is undefined. Sample topics are given, chosen deliberately to drive students to readings even if to find out what they mean. Sample questions include:

- How might Web2 principles promote democracy?
- How can computing make the invisible visible?
- How did the internet affect the Arab Spring?
- What is the impact of participatory media on local government?
- What are the employment implications of time spent playing online games?
- How might radical transparency affect business?
- How can we harness humanity's cognitive surplus for social good?

N.3.3 Assessment strategies

Assessment is by supported self assessment, with the final grade being negotiated between the instructor and student. For scholarly works with performance aspects (e.g. taught class), peer feedback was sought. The marking schedule varies according to the form of the submission, based upon the following: development of argument, justified 12 marks; content and style appropriate for agreed format (includes language, clarity, spelling). 5 marks; Appropriated referenced 3 marks (APA6th unless otherwise agreed); There is also a bonus 5 marks available for innovation in approach aiming to give credit for people pushing the boundaries.

N.4 Extensions

None.

N.5 Deliverables

Students must turn in a proposal and their completed scholarly work. About half write what could be considered a traditional essay. Others take a variety of approaches.