Overview

• What is Computing Education & Learning Technology research?
• Why is it an interesting area of research?
• A few examples
  • student projects (including COMPSCI 747)
  • leveraging existing expertise in Computer Science
• Overarching research questions
• Two examples
  • Specific research questions
What is it?

• Distinct from “teaching”
  • Teaching is helping others acquire knowledge and develop skills in a discipline
  • Research is creating new knowledge and exploring new ideas
What is it?

- Distinct from “teaching”
  - Teaching is helping others acquire knowledge and develop skills in a discipline
  - Research is creating new knowledge and exploring new ideas

- Computing Education Research:
  - the study of how people learn and teach computing
  - the goal is to help students learn, and teachers teach, more effectively
What is it?

• Distinct from “teaching”
  • Teaching is helping others acquire knowledge and develop skills in a discipline
  • Research is creating new knowledge and exploring new ideas

• Computing Education Research:
  • the study of how people learn and teach computing
  • the goal is to help students learn, and teachers teach, more effectively

• Learning Technology Research:
  • designing and evaluating tools for learning
  • covers the broader use of technology in teaching, learning and education across disciplines
Why research education?

• To have a positive impact in the world
  • Better outcomes for learners
Why research education?

• To have a positive impact in the world
  • Better outcomes for learners

• Practical application of technology
  • A clear need and an enormous user base
Why research education?

• To have a positive impact in the world
  • Better outcomes for learners

• Practical application of technology
  • A clear need and an enormous user base

• Big business
  • “Computational Thinking” is an essential 21st century skill, yet there are few people who can help others to develop those skills.
    • Constant need to train staff in technology
    • Increasing need for non-CS people to program
    • Increasing integration of programming skills into school curriculum
    • Increasing number of companies involved in technology to support education, and education about CS.

International education contributes $5.1 billion to New Zealand economy

HON CHRIS HIPKINS
A few student projects

• Researchers apply their expertise from many areas of CS
A few student projects

- Researchers apply their expertise from many areas of CS
  - **graph theory** (e.g. bibliometric analysis - James)
A few student projects

- Researchers apply their expertise from many areas of CS
  - graph theory (e.g. bibliometric analysis - James)
  - **program analysis**
    - static analysis (e.g. generating ASTs for visualising control flow - Lucy and Robert)
    - dynamic analysis (e.g. classification of array access errors - Liam)

```java
while (i < 10) {
    if (shouldIncrement()) {
        i++;
    }
}
```
A few student projects

- Researchers apply their expertise from many areas of CS
  - graph theory (e.g. bibliometric analysis - James)
  - program analysis
    - static analysis (e.g. generating ASTs for visualising control flow - Lucy and Robert)
    - dynamic analysis (e.g. classification of array access errors - Liam)
  - **machine learning/AI** (e.g. NLP for question answering - Bill)
A few student projects

- Researchers apply their expertise from many areas of CS
  - graph theory (e.g. bibliometric analysis - James)
  - program analysis
    - static analysis (e.g. generating ASTs for visualising control flow - Lucy and Robert)
    - dynamic analysis (e.g. classification of array access errors - Liam)
  - machine learning/AI (e.g. NLP for question answering - Bill)
  - game design (e.g. teaching version control, data structures – Kevin, Simon)
A few student projects

• Researchers apply their expertise from many areas of CS
  • graph theory (e.g. bibliometric analysis - James)
  • program analysis
    • static analysis (e.g. generating ASTs for visualising control flow - Lucy and Robert)
    • dynamic analysis (e.g. classification of array access errors - Liam)
  • machine learning/AI (e.g. NLP for question answering - Bill)
  • game design (e.g. teaching version control, data structures – Kevin, Simon)
  • VR/AR (e.g. visualising object transformations for computer graphics - Thomas)
A few student projects

• Researchers apply their expertise from many areas of CS
  • graph theory (e.g. bibliometric analysis - James)
  • program analysis
    • static analysis (e.g. generating ASTs for visualising control flow - Lucy and Robert)
    • dynamic analysis (e.g. classification of array access errors - Liam)
  • machine learning/AI (e.g. NLP for question answering - Bill)
  • game design (e.g. teaching version control, data structures – Kevin, Simon)
  • VR/AR (e.g. visualising object transformations for computer graphics - Thomas)
  • **software design** (e.g. using metaphors for teaching design patterns - Zain)
A few student projects

- Researchers apply their expertise from many areas of CS
  - graph theory (e.g. bibliometric analysis - James)
  - program analysis
    - static analysis (e.g. generating ASTs for visualising control flow - Lucy and Robert)
    - dynamic analysis (e.g. classification of array access errors - Liam)
  - machine learning/AI (e.g. NLP for question answering - Bill)
  - game design (e.g. teaching version control, data structures – Kevin, Simon)
  - VR/AR (e.g. visualising object transformations for computer graphics - Thomas)
  - software design (e.g. using metaphors for teaching design patterns - Zain)
  - **programming** (e.g. interactive tool for teaching debugging skills - Emma and Liz)
A few student projects

- Researchers apply their expertise from many areas of CS
  - graph theory (e.g. bibliometric analysis - James)
  - program analysis
    - static analysis (e.g. generating ASTs for visualising control flow - Lucy and Robert)
    - dynamic analysis (e.g. classification of array access errors - Liam)
  - machine learning/Al (e.g. NLP for question answering - Bill)
  - game design (e.g. teaching version control, data structures – Kevin, Simon)
  - VR/AR (e.g. visualising object transformations for computer graphics - Thomas)
  - software design (e.g. using metaphors for teaching design patterns - Zain)
  - programming (e.g. interactive tool for teaching debugging skills - Emma and Liz)
  - **computer systems** (e.g. compiler error messages - Dave)

```java
if (a < 0) || (a > 100)
error = true;
```

```plaintext
if (a < 0) || (a > 100)
  ^^
Syntax error on token "||", if expected
  1 error
```
Graduate course in Computing Education

James Garner
University of Auckland
New Zealand
jgar@cs.auckland.ac.nz

Paul Denny
University of Auckland
New Zealand
paul.c.auckland.ac.nz

Andrew Luxton-Reilly
University of Auckland
New Zealand
andrews.c.auckland.ac.nz

ABSTRACT

Mastery learning is a pedagogical approach in which students must demonstrate mastery of the currently assessed unit of material before being permitted to progress to the next unit. Recent work has suggested that mastery learning may provide a solution to the disengaged outcomes observed in introductory computer science (CS) courses. While mastery learning has shown benefits outside of CS, it has received less attention in CS education, and there is no extant overview of the methods and tools used in the field. This paper presents a case study of a student group to gain an early concept which will be magnified in use needs involving later concepts. This problem should not arise in mastery learning approach, since mastery learning does not demand students to attempt a later assessment without first mastering an earlier concept. Several of the papers included in this review are the learning edge “Mentorship” paper that supports for adoption mastery learning approach.

Despite the relevance of mastery learning to computer science education today, the literature on it is scarce, which influenced the paper. We provide a discussion of the strengths, weaknesses, and remaining questions related to the use of mastery learning in CS education, as well as recommendations for future research.

Transitions from Block-based to Text-based Programming Languages

Lake Moore
Department of Computer Science
University of Auckland
New Zealand
lmoore@cs.auckland.ac.nz

Andrew Luxton-Reilly
University of Auckland
New Zealand
andrews.c.auckland.ac.nz

Paul Denny
Department of Computer Science
University of Auckland
New Zealand
paul.c.auckland.ac.nz

ABSTRACT

Block-based programming environments are becoming increasingly popular as introductory tools for teaching programming to children. These environments differ significantly from their text-based counterparts and have proven to be successful in motivating children and making it easier to learn programming. This paper presents the findings of a study comparing the effectiveness of two block-based environments: Scratch and Alice. The study involved 120 participants who were randomly assigned to one of the two environments. The results showed that the participants in the Scratch group performed significantly better than those in the Alice group on measures of programming ability and engagement. The study also found that Scratch was more accessible to novice programmers, allowing them to create media-rich content in relation to their own interests. Storytelling Alice, in particular, implements a storytelling approach found to be appealing to female students. On the other hand, Michael Pehaya, a researcher at Oxford University, has shown that Scratch is more effective in teaching problem-solving skills compared to Alice. Future research could explore the potential of Scratch-based programming environments in teaching higher-level programming concepts.

Keywords: block-based programming, text-based programming, Alice, Scratch, programming ability, engagement, storyline, student motivation.

A Review of Peer Code Review in Higher Education

Theresia Devi Indrasisari
School of Computer Science: The University of Auckland, New Zealand

Andrew Luxton-Reilly and Paul Denny
School of Computer Science: The University of Auckland, New Zealand

ABSTRACT

Peer review is the standard process within academia for maintaining publication quality, but it is also widely employed in other settings, such as education and industry, for improving work quality and generating actionable feedback to content authors. For example, in the software industry peer review of program source code—or peer code review—is a key technique for detecting bugs and maintaining coding standards. In a programming education context, however, peer code review offers potential benefits to both code reviewers and code authors, individuals are typically less experienced, which presents a number of challenges. Some common logic errors made by novice programmers include:

1. Common Logic Errors Made by Novice Programmers

<table>
<thead>
<tr>
<th>Author</th>
<th>University</th>
<th>Auckland, New Zealand</th>
<th>Contact</th>
</tr>
</thead>
<tbody>
<tr>
<td>Andrew</td>
<td>University</td>
<td>Auckland, New Zealand</td>
<td>Contact</td>
</tr>
<tr>
<td>Luxton-Reilly</td>
<td>University</td>
<td>Auckland, New Zealand</td>
<td>Contact</td>
</tr>
<tr>
<td>Paul</td>
<td>University</td>
<td>Auckland, New Zealand</td>
<td>Contact</td>
</tr>
</tbody>
</table>

ABSTRACT

The logic of a program (sometimes referred to as semantic or logical errors) can be very frustrating for novice programmers to detect and correct. Developing a better understanding of the kinds of logic errors that are most common and problematic for students, and finding strategies for teaching them, may help to improve teaching practice and reduce student frustration.

In this paper, we analyze 1,286 code snippets created by novice program reviewers, who indicated errors, and classify the errors into five categories: syntax errors, semantic errors, run-time errors, and logical errors. Our findings indicate that one of the most difficult types of errors for students to identify is that of logical errors. The most common error type was logical errors, followed by syntax errors.

Keywords: logical errors, novice programmers

2 RELATED WORK

The term “logic error” refers to an error in a program that is not detected by the compiler or interpreter, but is detected during runtime. Logic errors are often the result of incorrect assumptions about how the program will behave. Logical errors can be difficult to detect, as they may not cause the program to crash, but can still produce incorrect results.


discussion on the importance of engaging students in the learning process and the role of peer code review in supporting this engagement. The authors present empirical evidence that peer code review can improve student learning outcomes and promote a more collaborative and inclusive learning environment. Case studies are used to illustrate the effectiveness of peer code review in real-world settings, highlighting the potential benefits for both students and instructors. The paper concludes by outlining future research directions and recommendations for implementing peer code review in higher education institutions.
Example

- Liam Rigby’s paper
  - COMPSCI747 project
  - Australasian Computing Education Conference (ACE) 2020
  - Dynamic analysis of large existing code dataset

---

A Miss is as Good as a Mile: Off-By-One Errors and Arrays in an Introductory Programming Course

Liam Rigby  
The University of Auckland  
New Zealand  
lrig521@aucklanduni.ac.nz

Paul Denny  
The University of Auckland  
New Zealand  
paul@cs.auckland.ac.nz

Andrew Luxton-Reilly  
The University of Auckland  
New Zealand  
a.luxton-reilly@auckland.ac.nz

ABSTRACT

Loops and arrays are fundamental CS1 concepts, but ones that can be problematic for novice programmers. In this research, we investigate off-by-one errors — logic errors where loops perform one too few or one too many iterations — in code using an indexed loop over an array. We classify off-by-one errors, and explore the prevalence of each type, by analyzing a large set of code submissions from students in a first year programming course as they tackle a sequence of exercises. We describe an approach to reliably identify off-by-one errors through dynamic analysis, and find that off-by-one errors are both common and persist across exercises. We also show that students infrequently choose to iterate over an array in reverse, but when they do they more commonly encounter off-by-one errors. We conclude that teaching material should explicitly focus student attention on boundary cases, and should provide more examples that iterate through arrays in reverse.
Example

- Liam Rigby’s paper
  - COMPSCI747 project
  - Australasian Computing Education Conference (ACE) 2020
  - Dynamic analysis of large existing code dataset

was reviewing recently...

report arrays and indexing problems, noting off-by-one errors and difficulties setting up the appropriate range. Rigby et al [25] examine off-by-one-errors in which students make logic errors resulting in loops performing too few or too many iterations, and find that such errors are both common and persist across multiple types of exercises. These difficulties have led some computing education researchers to argue for the use of collection objects and their
Example

• Liam’s paper
  • CS747 project
  • Published in ACE 2020
  • Analysed an existing dataset

was reviewing the
report arrays and
difficulties setting
off-by-one-errors
loops performing
such errors are big
exercises. These
researchers to a

SO I GUESS YOU CAN SAY

I'M PRETTY FAMOUS NOW
An exciting time

- Rapid growth in adoption of learning tools
- Vast amounts of data collected on how people learn

**Anant Agarwal (MIT / EdX)**

“Data collection for educational research is one of the key goals of EdX.... we gather huge amounts of data.... all this rich data will be available to researchers.... to understand how people really learn and we can help synthesize a better educational experience”

**Daphne Koller (Stanford / Coursera)**

“Tremendous opportunities.... every click, every homework submission, every forum post, from tens of thousands of students.... turn the study of human learning from the hypothesis driven mode to the data driven mode”
Example: keystroke vs submission data

Courtesy: Juho Leinonen (University of Helsinki)
Overarching research questions:
- How do people learn computing?
- How do teachers teach and assess computing?
- How can people learn computing more effectively?
- How can teachers teach computing more effectively?
- How can access to computing education be improved?
- How can computing education be delivered equitably to all?
- How can learning technologies teach computing?
- How does computing education affect people's lives?
- What are the societal costs of computing illiteracy?
- What does it mean to know computing?
- What can be taught about computing to learners of different ages?
Two examples

• Two recent studies
• Feedback and learner behavior
  • Example 1
    • Computing education: Compiler error messages
  • Example 2
    • Learning technology: Influencing (positive) behaviours
Example 1: Compiler error messages
II. Background and Objectives

Manufacturer-supplied Fortran compilers normally provide rather efficient object code, provide flexible interaction with the operating systems, and have many sophisticated programming features. However, they are inadequate for the needs presented in the area of finding and correcting errors as quickly as possible. In many instances, the description of an error condition lacks resolution and offers the user little assistance in removing the error other than indicating the statement in which the error occurs. A more serious inadequacy is that many error descriptions are given in terms not understandable to a Fortran programmer.

DITRAN—a compiler emphasizing diagnostics

Moulton and Muller

Communications of the ACM

January 1967

Denny, Prather & Becker (2020)
Example 1: Compiler error messages

Yet, ask any experienced programmer about the quality of error messages in their programming environments, and you will often get an embarrassed laugh. In every environment, a mature programmer can usually point to at least a handful of favourite bad error responses. When they find out that the same environment is being used by novices, their laugh often hardens.
Example 1: Compiler error messages

Error Message Readability and Novice Debugging Performance
Paul Denny
University of Auckland
Auckland, New Zealand
paul.denny@auckland.ac.nz

James Prather
Abilene Christian University
Abilene, Texas, USA
james.prather@acu.edu

Brett A. Becker
University College Dublin
Dublin, Ireland
brett.becker@ucd.ie

ABSTRACT
It is well known that programming error messages can be notoriously difficult for novices to understand, hampering progress and leading to frustration. In response, researchers have explored various approaches for enhancing such messages, yet results from this active strain of research are currently mixed. Direct comparisons of results between studies is challenging as these typically investigate different kinds of message enhancements and report results using different metrics. In addition, many prior studies have involved code writing tasks. In such cases, not all students encounter the same errors and messages, and it is difficult to isolate the time spent interpreting messages and resolving errors from the time spent writing code. In this research, we explore the effects of presenting novices with compiler error messages designed using the most recent collection of published guidelines – specifically, more easily readable, short, positive messages containing resolution hints. To accurately determine the time and effort required to read and respond to the messages, we utilized a debugging task where all students are presented the same code and therefore encounter the same errors. We present results of a randomized controlled experiment (n > 700) which shows that, compared to standard error messages, the messages tested resulted in significantly shorter debugging times and higher self-reported scores of message usefulness for students in the early stages of learning a new language.

CCS CONCEPTS
- Social and professional topics → CS1: Computing education; Computer science education; - Human-centered computing → Human-computer interaction

KEYWORDS
compiler error messages; CSS debugging; error message enhancement; novice programmers; programming error messages; readability

ACM Reference Format:

https://clang.llvm.org/diagnostics.html

Denny, Prather & Becker (2020)
#include <stdio.h>

#define CENTIMETERS_TO_FEET 0.0328
#define CENTIMETERS_TO_INCH 0.3937

int main(void)
{
    // Variables for converting metric to imperial
    int centimeters feet;
    double inches;

    // Read value into the variable centimeters
    scanf("%d", centimeters);

    feet = centimeters * CENTIMETERS_TO_FEET;
    inches = (centimeters - feet / CENTIMETERS_TO_FEET) * CENTIMETERS_TO_INCH;
    printf("%d centimeters is %d feet and %.2f inches\n", centimeters, feet, inches);

    return 0;
}
Example 1: Compiler error messages

```c
#include <studio.h>
#define CENTIMETERS_TO_FEET 0.0328
#define CENTIMETERS_TO_INCH 0.3937

int main(void)
{
    // Variables for converting metric to imperial
    int centimeters, feet;
    double inches;

    // Read value into the variable centimeters
    scanf("%d", centimeters);

    feet = centimeters * CENTIMETERS_TO_FEET;
    inches = (centimeters - feet / CENTIMETERS_TO_FEET) * CENTIMETERS_TO_INCH;
    printf("%d centimeters is %d feet and %.2f inches\n", centimeters, feet, inches);

    return 0;
}
```

1:10: fatal error: studio.h: No such file or directory
Example 1: Compiler error messages

```c
#include <studio.h>
#define CENTIMETERS_TO_FEET 0.0328
#define CENTIMETERS_TO_INCH 0.3937

int main(void)
{
    // Variables for converting metric to imperial
    int centimeters, feet;
    double inches;

    // Read value into the variable centimeters
    scanf("%d", &centimeters);

    feet = centimeters * CENTIMETERS_TO_FEET;
    inches = (centimeters - feet / CENTIMETERS_TO_FEET) * CENTIMETERS_TO_INCH;
    printf("%d centimeters is %d feet and %.2f inches\n", centimeters, feet, inches);

    return 0;
}
```

1:10: fatal error: studio.h: No such file or directory
Example 1: Compiler error messages

```c
#include <studio.h>

#define CENTIMETERS_TO_FEET 0.0328
#define CENTIMETERS_TO_INCH 0.3937

int main(void)
{
    // Variables for converting metric to imperial
    int centimeters feet;
    double inches;

    // Read value into the variable centimeters
    scanf("%d", centimeters);

    feet = centimeters * CENTIMETERS_TO_FEET;
    inches = (centimeters - feet / CENTIMETERS_TO_FEET) * CENTIMETERS_TO_INCH;
    printf("%d centimeters is %d feet and %.2f inches\n", centimeters, feet, inches);
    return 0;
}
```

Example 1: Compiler error messages

```
#include <studio.h>
#define CENTIMETERS_TO_FEET 0.0328
#define CENTIMETERS_TO_INCH 0.3937

int main(void)
{
    // Variables for converting metric to imperial
    int centimeters feet;
double inches;

    // Read value into the variable centimeters
    scanf("%d", centimeters);

    feet = centimeters * CENTIMETERS_TO_FEET;
inches = (centimeters - feet / CENTIMETERS_TO_FEET) * CENTIMETERS_TO_INCH;
printf("%d centimeters is %d feet and %.2f inches\n", centimeters, feet, inches);

    return 0;
}
```

Example 1: Compiler error messages

```
#include <studio.h>
#define CENTIMETERS_TO_FEET 0.0328
#define CENTIMETERS_TO_INCH 0.3937

int main(void)
{
    // Variables for converting metric to imperial
    int centimeters feet;
    double inches;

    // Read value into the variable centimeters
    scanf("%d", centimeters);

    feet = centimeters * CENTIMETERS_TO_FEET;
    inches = (centimeters - feet / CENTIMETERS_TO_FEET) * CENTIMETERS_TO_INCH;
    printf("%d centimeters is %d feet and %.2f inches\n", centimeters, feet, inches);
    return 0;
}
```

13:5: error: expected declaration specifiers before 'scanf'
Example 1: Compiler error messages

```
#include <stdio.h>
#define CENTIMETERS_TO_FEET 0.0328
#define CENTIMETERS_TO_INCH 0.3937

int main(void)
{
    // Variables for converting metric to imperial
    int centimeters, feet;
    double inches;

    // Read value into the variable centimeters
    scanf("%d", centimeters);

    feet = centimeters * CENTIMETERS_TO_FEET;
    inches = (centimeters - feet / CENTIMETERS_TO_FEET) * CENTIMETERS_TO_INCH;
    printf("%d centimeters is %d feet and %.2f inches\n", centimeters, feet, inches);

    return 0;
}
```

13:5: error: expected declaration specifiers before ‘scanf’
Example 1: Compiler error messages

```c
#include <studio.h>
#define CENTIMETERS_TO_FEET 0.0328
#define CENTIMETERS_TO_INCH 0.3937

int main(void)
{
    // Variables for converting metric to imperial
    int centimeters feet;
    double inches;

    // Read value into the variable centimeters
    scanf("%d", centimeters);

    feet = centimeters * CENTIMETERS_TO_FEET;
    inches = (centimeters - feet / CENTIMETERS_TO_FEET) * CENTIMETERS_TO_INCH;
    printf("%d centimeters is %d feet and %.2f inches\n", centimeters, feet, inches);
    return 0;
}
```

13:13: error: format ‘%d’ expects argument of type ‘int *’, but argument 2 has type ‘int’.
Example 1: Compiler error messages

```c
#include <stdio.h>
#define CENTIMETERS_TO_FEET 0.0328
#define CENTIMETERS_TO_INCH 0.3937

int main(void)
{
    // Variables for converting metric to imperial
    int centimeters, feet;
    double inches;

    // Read value into the variable centimeters
    scanf("%d", &centimeters);

    feet = centimeters * CENTIMETERS_TO_FEET;
    inches = (centimeters - feet / CENTIMETERS_TO_FEET) * CENTIMETERS_TO_INCH;
    printf("%d centimeters is %d feet and %.2f inches\n", centimeters, feet, inches);

    return 0;
}
```

13:13: error: format ‘%d’ expects argument of type ‘int *’, but argument 2 has type ‘int’.
Example 1: Compiler error messages

#include <studio.h>

#define CENTIMETERS_TO_FEET 0.0328
#define CENTIMETERS_TO_INCH 0.3937

int main(void)
{
    // Variables for converting metric to imperial
    int centimeters feet;
double inches;

    // Read value into the variable centimeters
    scanf("%d", centimeters);

    feet = centimeters * CENTIMETERS_TO_FEET;
inches = (centimeters - feet / CENTIMETERS_TO_FEET) * CENTIMETERS_TO_INCH;
printf("%d centimeters is %d feet and %.2f inches\n", centimeters, feet, inches);

    return 0;
}
Example 1: Compiler error messages

Could we provide more useful error messages?
What sort of difference would that make?
Example 2: Influencing (positive) behaviours

Empirical Support for a Causal Relationship Between Gamification and Learning Outcomes

Paul Denny, Fiona McDonald, Ruth Empson, Philip Kelly, Andrew Petersen

ABSTRACT
Preparing for exams is an important yet stressful task for many students. Self-testing is known to be an effective preparation strategy, yet some students lack motivation to engage or persist in self-testing activities. Adding game elements to a platform supporting self-testing may increase engagement and, by extension, exam performance. We conducted a randomized controlled experiment (n=701) comparing the effect of two game elements – a points system and a badge system – used individually and in combination.

We find that the badge system elicits significantly higher levels of voluntary self-testing activity and this effect is particularly pronounced amongst a relatively small cohort. Importantly, this increased activity translates to a significant improvement in exam scores. Our data supports a causal relationship between gamification and learning outcomes, mediated by self-testing behaviour. This provides empirical support for Landers’ theory of gamified learning when the gamified activity is conducted prior to measuring learning outcomes.

This raises the question, “Can gamification positively impact student behavior and learning outcomes?”

In a comprehensive review of the literature, Hamari et al. report that three particular elements: points, leaderboards and badges are the most commonly used in empirical studies of gamification [28]. Their review concluded that most published gamification studies reported some positive effects, but they identified a number of methodological limitations that may have contributed to varying results. These limitations included small sample sizes, lack of control groups and very short experimental timelines. A fourth limitation was that multiple game elements were often investigated in combination, but not individually, making it impossible to establish whether individual elements had measurable effects.

In this study we investigate two of the most common gamification elements, points and badges, as used in an online learning tool. Our context is a large first-year anatomy and physiology course (700 participants), where we investigate student engagement with the tool over an entire 15 week semester and relate engagement to subsequent exam performance. We examine the effects of the game elements both individually and in combination, relative to a control group.

We explore two related research questions. Our primary question tests the hypothesis that gamifying an online study tool will have a causal effect on subsequent exam performance. Landers’ theory of gamified learning provides strong theoretical support for this hypothesis [34]. Our secondary research question tests the hypothesis that a combination of game elements will have a greater effect on student behaviour than either element used on its own. We measure the individual effects of our implemented points and badge systems, and we determine if their simultaneous use is beneficial in one context.

BACKGROUND
Gamification is an increasingly common application area for gamification [2, 33, 51]. This has been driven by the potential for gamification to address challenges around student motivation and to positively impact learning [8, 36]. This latter outcome is of particular importance in educational contexts. The relationship between gamification and learning outcomes may be mediated by behaviors, such as time-on-task, that

Denny, McDonald, Empson, Kelly & Petersen (2018)
Example 2: Influencing (positive) behaviours

Denny, McDonald, Empson, Kelly & Petersen (2018)
Example 2: Influencing (positive) behaviours

• Or, avoiding (negative) behaviours

Can Mobile Gaming Psychology Be Used to Improve Time Management on Programming Assignments?

Michael S. Irwin and Stephen H. Edwards
Department of Computer Science
Virginia Tech
Blacksburg, VA, USA
mikesir@vt.edu, edwards@cs.vt.edu

Figure 1: Submission energy bar, showing countdown and the animated fading of the next available unit being regenerated.

Irwin & Edwards (2019)
Example 2: Influencing (positive) behaviours

The Programme

How do I receive points?

Every time you contribute to TripAdvisor, you receive TripCollective points. Here’s a list of what you can contribute, and how much it’s worth.

<table>
<thead>
<tr>
<th>Activity</th>
<th>Points</th>
</tr>
</thead>
<tbody>
<tr>
<td>Review</td>
<td>100</td>
</tr>
<tr>
<td>Photo</td>
<td>30</td>
</tr>
<tr>
<td>Video</td>
<td>30</td>
</tr>
<tr>
<td>Forum Post</td>
<td>20</td>
</tr>
<tr>
<td>Rating</td>
<td>5</td>
</tr>
</tbody>
</table>

TripCollective FAQs:

The Basics

What is TripCollective?

TripCollective is our enhanced contributor programme that recognises you each time you add to TripAdvisor. Think of it as your travel community’s way of saying thanks for helping us collectively travel better.

How do I become involved?

It’s easy – anybody who has contributed anything to TripAdvisor, be it a review, photo, forum post, rating or...
Example 2: Influencing (positive) behaviours

The Programme

How do I receive points?

Every time you contribute to TripCollective, you receive points. The more you contribute, the more points you earn.

- Review
- Photo
- Video
- Forum Post
- Rating

Do points have monetary value?

TripCollective points do not have monetary value and cannot be redeemed for anything.
“Empirical research on the *effectiveness* of incorporating game elements in learning environments is still scarce”

[Dicheva et al.; 2015]
Example 2: Influencing (positive) behaviours

Generation and Retrieval Practice Effects in the Classroom Using PeerWise

Matthew R. Kelley¹, Elizabeth K. Chapman-Orr², Susanna Calkins³, and Robert J. Lemke⁴

Abstract
The present study explored the generation and retrieval practice effects within a college classroom using a free, online tool called PeerWise (PW). PW allows students to create their own multiple-choice questions, share them with peers, and answer the shared questions written by their peers. Forty students from two sections of an upper level cognitive psychology course authored and answered multiple-choice questions as part of a semester-long assignment. Analyses showed reliable generation and retrieval practice effects following PW usage, along with a significant improvement in exam performance.

Keywords
generation effect, retrieval practice, PeerWise

## Example 2: Influencing (positive) behaviours

<table>
<thead>
<tr>
<th>Desired behavior</th>
<th>Example reward</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rate questions early and fairly</td>
<td>Reputation score (rating component)</td>
</tr>
<tr>
<td>Answer questions correctly</td>
<td>Answer score</td>
</tr>
<tr>
<td>Spaced practice sessions</td>
<td>Commitment badge</td>
</tr>
<tr>
<td>Create good questions</td>
<td>Good question author badge</td>
</tr>
<tr>
<td>.....</td>
<td>.....</td>
</tr>
</tbody>
</table>

![Diagram showing points and badges](image-url)
Example 2: Influencing (positive) behaviours

Example 2: Influencing (positive) behaviours

“Personally I tried really hard to get the ‘Leader’ badge, where I had to gain at least one follower! This was really motivating, and made me think more carefully and creatively when writing a question.

“I didn't think I was ‘badge’ type of person, but I did enjoy getting badges (I was the first one to get the ‘obsessed badge’ - yay!). It did help motivate me to do extra and in doing so, I believe I have learnt more effectively.”
Example 2: Influencing (positive) behaviours

Game elements caused:
- twice as many students to create questions (6.7% vs. 11.5%)
- nearly 40% increase in answering activity

37% increase in median number of answers
(p = .016)

4.5% increase in mean exam score
(p = .038)
Summary

• The goal of Computing Education research is to help students learn computer science more effectively
• The goal of Learning Technology research is the same, but applies more broadly across disciplines
• These are interesting areas of research, to which a range of computer science skills can be applied, and with the potential of large impact
• Our School’s graduate course is COMPSCI 747!