UNIVERSITY OF CAPE TOWN
IYUNIVESITHI YASEKAPA • UNIVERSITEIT VAN KAAPSTAD

Department of Computer Science

Computer Science Honours Final Paper 2016

Title: Playduino: Teaching Programming using Arduino, and Game-based Educational Frameworks

Author: Mitch Myburgh

Project abbreviation: ARDUINOED

Supervisor: Gary Stewart

Category	**Min**	**Max**	**Chosen**
Requirement Analysis and Design	0	20	0
Theoretical Analysis	0	25	0
Experiment Design and Execution	0	20	20
System Development and Implementation	0	15	10
Results, Findings and Conclusion	10	20	20
Aim Formulation and Background Work	10	15	10
Quality of Paper Writing and Presentation	10		10
Quality of Deliverables	10		10
Overall General Project Evaluation	0	10	0
Total marks			80

Playduino: Teaching Programming using Arduino, and Game-based Educational Frameworks

Final Project

Mitch Myburgh
MYBMIT001
University of Cape Town
Rondebosch
Cape Town, South Africa
mybmit001@myuct.ac.za

ABSTRACT

Computer Science is an important skill in the modern world, and engaging first year students in programming is essential to build the high tech workforce that South Africa needs. With the rise of connected and 'smart' devices, hardware design and programming is also becoming a necessity in a modern and well rounded Computer Science education. Teaching students to use and program for cheap, open source and easy to use microcontrollers such as the Arduino will equip them with skills essential in a number of disciplines in Computer Science. Despite this need for Computer Scientists, first year Computer Science is difficult for many students and sees flagging enrollment and high dropout rates. It is essential to engage students in new and exciting ways to combat this.

The Arduino is a low-cost, open-source microcontroller that includes a vibrant development community with a large number of examples and strong support. This makes it ideal for introducing students to both programming and hardware design. It is a small computing device capable of interfacing with a number of hardware and electronics components, this makes it ideal for prototypers, hackers, students and designers. It enables students to gain experience building hardware and coding interactivity into it.

Playduino is an educational framework designed for first-year, university-level Computer Science Courses. The framework consists of a game engine, game console, documentation and two sample assignments. Playduino aims to intergrate both games and hardware (Arduino) programming into first-year courses, in an attempt to increase student motivation in their assignments and course content.

Games and hardware have been used in a number of studies, courses and workshops and have been found to engage, interest and motivate students, however no papers currently exist that investigate the effect of integrating both games and hardware on student motivation. Playduino seeks to close this gap by providing a game console and set of programming assignments based on the Arduino microcontroller, as well as data on motivation levels associated with this framework.

The study involves an expert evaluation of the framework in the form of one-on-one conversations with experts, such as teacher's assistants and tutors, as well as a standard materials motivation survey.

The results were overwhelmingly positive with statistically significantly better results than traditional assignments. The experts were also entirely in favour of the integration of the assignments in class.

CCS Concepts

•**Applied computing** → **Interactive learning environments;** •**Hardware** → **Sensor applications and deployments; Sensor devices and platforms;** *Displays and imagers; Networking hardware;* •**Human-centered computing** → Handheld game consoles;

Keywords

Education; Microcontrollers; Arduino; Programming; Game Framework

1. INTRODUCTION

Computer Science and programming is a fast growing field, with computer programming being among the industries with the fastest growth in demand[31] from 1965 to 1994 in South Africa. Despite the large demand South Africa has a lack of qualified Computer Scientists, *Daniels and Reza* list computer-related professionals in their list of scarce skills[9] and the South African Government list *Information and Communication Technology* and *Information Technology* on their 2014 *List of Occupations in High Demand*[25].

Hardware devices such as microcontrollers and even computers have reduced in price in recent years, making developing hardware systems and infrastructure accessible to more people than ever before. Devices such as the Arduino, an open-source, low-cost, and easy to use microcontroller board, have enabled anyone to own and build microcontroller systems. With the rise of the Internet of Things[19], hardware and micrcontroller programming is becoming more and more important.

ACM ISBN 978-1-4503-2138-9.
DOI: 10.1145/1235

Having a robust knowledge of not only computer programming, but programming on dedicated hardware (such as microcontrollers) is an essential skill in the modern world, one that is needed in South Africa. Equipping students with these skills will allow them to compete in the knowledge-based economies of the 21^{st} Century[29]. Futhermore engaging often apathetic students in Computer Science is an important task.

Hardware programming and game programming are two options that, the literature has shown, have potential to engage, interest and motivate students, potentially stemming the tide of dropouts[2] and flagging enrollments[21][29][20][8][2] and creating a new generation of students eager to get involved in Computer Science, however no previous research has been done on integrating both into a single framework.

Playduino is an educational framework based around a C++ game engine and Arduino-based game console. It also includes detailed documentation, two sample assignments and exemplar code for the assignments. The game engine provides abstractions for dealing with the hardware, enabling students to focus on the coding. The assignments are available as skeleton code which the students fill in, as well as larger assignments with minimal scaffolding. The frameworks can be used as is, or new assignments can be created using the hardware and software to test specific skills.

This project is an investigation into the use of an educational framework for building games running on microcontrollers (namely the Arduino system) in order to teach first year students programming, and engage and interest them in expanding their knowledge. The project will investigate the potential for this framework to increase student motivation and engagement in the Computer Science curriculum, using a standard materials motivation survey. The framework will be evaluated by domain experts, including teacher's assistants and tutors.

The Arduino system is a low-cost, open-source platform for hardware design and prototyping. It is coded in a C-based language, which is procedural and will introduce students to concepts and syntax pervasive amoung the 'curly-brace' languages such as Java, Javascript, C++ and C#. This makes the Arduino an ideal candidate for introducing students to programming and hardware.

1.1 Research Question

The research question in this paper is:
Can Arduinos and games, both successful in raising student motivation, be combined to increase student motivation and interest in Computer Science?

The background work shows that game- and microcontroller-based assignments have both been successful in increasing student motivation, engagement and interest. No previous research has been done on integrating games and microcontrollers, this paper seeks to investigate this.

1.2 Aim

This paper hypothesises that using games and Arduino microcontrollers will increase the motivation of students after having completed the assignment. The aim is to investigate this claim by building and user-testing an educational framework in which students program a game using a custom game engine and handheld game console built with the Arduino.

The educational framework, Playduino, will be evaluated through an expert evaluation, in which experts attempt the assignment while giving feedback as well as by completing a motivation survey. The experts were also asked to complete the same survey on an assignment they had marked or done before, for comparison with the assignment presented in the educational framework.

2. BACKGROUND WORK

This section will present an overview of *Programming Education*, the problems it faces, possible reasons for these problems and potential solutions. It will then look at two methods to solve the problems in programming education namely *Games in Programming Education* and *Microcontrollers in Programming Education*, in which the use of these in programming classes is investigated and their success evaluated.

2.1 Programming Education

Learning programming is difficult[11][15][18]. From information technology courses in South African schools[18] to introductory Computer Science courses at universitys around the world[15] the literature agrees that teaching and learning programming skills are both difficult tasks that require refinement in order to equip more students with the basic coding and problem solving skills learned in introductory Computer Science courses.

Coupled with this difficulty Computer Science courses are seeing declining enrollment[21][29][20][8][2] and high dropout rates[2].

A report by the *ITiCSE 2001 Working Group*[22] in which the programming skills of first year computer science students were evaluated, found that many of the students do not have sufficient programming and problem solving skills. They were required to complete a test, in which many students performed poorly. Although this test was administered to only 216 students from 4 universities, it is indicative of a larger problem, that students are not coping with the difficult Computer Science curriculum[11][15].

Another paper by Jens Bennedsen and Michael E. Caspersen[3] investigates failure rates in first year computer science courses. In a survey of 63 institutions it was found that 67% of first year Computer Science students pass the course, which means that of the 2 million students who in enroll in such courses worldwide, 650,000 fail to pass the course[3].

Leutenegger and Edgington[21] report that from 2000 to 2005 the number of incoming computer science majors dropped by 60% to 70%, while Bayliss[2] reports a more conservative 50% drop from 2000 to 2007 and a dropout rate of 30% to 40%, which is in line with Bennedsen et al.'s [3] first year drop out rate of 33%.

The issues put forward by the *ITiCSE 2001 Working Group*[22] and Bennedsen et al.[3] regarding the inability of students to program and their failure rates has a number of reasons, chief among them the difficulty (and perceived difficulty) of introductory programming courses.

Papers by Anabela Gomes and António José Mendes [11], and Tony Jenkins[15] discuss this issue, suggesting reasons why the courses are considered difficult and possibly what steps can be taken to remedy the situation and more effectively teach students programming skills. They argue that students are not equipped with problem solving skills, mathematical ability and are often faced with complex programming languages in first year, as many courses favour 'real-

world' languages over ones designed for education. Another reason put forward is a lack of motivation, a problem that this paper seeks to address. This lack of motivation results in the student's interest waning throughout their Computer Science career, and are often linked to growing negative connotations about the field[11].

Gomes et al.[11] and Jenkins[15] put forward potential solutions to the problems highlighted above.

Among the potential solutions highlighted by Gomes et al.[11] two stand out as relevant, namely including games in the syllabus (This will be discussed in depth in the section on *Programming Education Through Games*), and using programming patterns to assist in development. Gomes et al.[11] suggest that incomplete patterns be presented to students that they then fill out with code this can assist students struggling to create the structure of the program as well as to provide examples of best practices and reinforce these in the student's mind.

2.2 Games in Programming Education

Games are probably the most obvious tools in increasing engagement and learning in the Computer Science classroom, as most students entering the field of Computer Science are familiar with games[20][29][21]. Games have captured the attention of young people and become a multibillion dollar[27] industry in the process, they hold the attention of players for often long stretches of time[27], during which they learn from the game world, it would be advantageous if this could be applied to the study of Computer Science.

Due to their interactive and self paced nature, computer games can be useful for teaching students new and difficult concepts. Games also capture players' attention and can also be used as a motivational tool in the classroom through gamification[26].

Games have been used to teach children and students programing concepts including *Toontalk*[16], a game to teach students about variables[32], real-time strategy games[24] and a UNIX scavenger hunt[2].

The use of games as programming assignments has also been discussed in the literature. Game assignments can provide students with additional motivation[21][8], can be implemented without sacrificing content[21][29], and the motivation from games can encourage students to look into more complex topics in Computer Science[20].

Students are often interested in making games, with many choosing a computer science major based on their love of games, and desire to make their own[20]. These students may become disillusioned by difficult, boring or seemingly irrelevant assignments[20][21]. However integrating game based assignments could increase motivation and keep students interested in the course.

Two papers by Cliburn[7][8] discusses experiments in which he provided students with a choice between game-based and traditional assignments. In both experiments the students overwhelmingly chose the game-based assignments with 79% of students choosing games in the first experiment and 71% of students choosing the game-based assignment in the second experiment. Despite this the game assignments had no impact on the average marks, as many of the stronger students chose the standard assignments that they perceived to be more difficult. The students did state that their motivation was increased when faced with game based assignments. Cliburn's study was based on a single class and may not be wholly representative of Computer Science students throughout the world.

In their paper Leutenegger et al.[21] discuss the use of games in an introductory Computer Science course. These games allowed students to get immediate visual feedback as to issues and bugs in their code, and Leutenegger et al. assert they were able to achieve this without sacrificing content.

Sung et al.[30] produced a series of game themed assignments that could be dropped into existing courses with minimal effort, taking the technical topics that needed to be tested and integrating them into games in order to provide additional motivation for students. Sung et al. made use of code scaffolding in their assignments.

In another article Sung[29] further discusses the use of games in programming assignments. Sung notes that in elective Computer Science courses, such as artificial intelligence, software engineering and computer graphics, game-based assignments fit particularly well given the close links these topics have to the game industry. This is backed up by Kurkovsky[20] who found students became engaged in more advanced topics such as databases, networking, artificial intelligence and software engineering in his mobile game assignments.

Bayliss's[2] game-based Compuer Science course showed a 93% retention rate throughout the degree, showing the power that integrating games into the Computer Science curriculum could have on retaining and engaging students in the content. She also makes use of game scaffolding.

2.3 Microcontrollers in Programming Education

Microcontrollers have become a pervasive part of the modern world, used in industry, cars, consumer electronics and others[13] to create connected and 'smart' devices. As a result many attempts have been made to engage students in the programming and creation of microcontroller enabled devices from using e-textiles[5][6], to designing graphical programming languages to help students to learn to use the devices[4][12][17].

The Arduino is a popular kit for many microcontroller based courses, due to its open-source nature, ease of use, low cost, and strong community[14].

Rubio et al.[28] made use of Arduinos in a course to teach traditional programming. The students were exposed to a traditional lecture, with integration of an Arduino example to illustrate concepts such as loops, conditional statements and arrays. The students were then given the opportunity to build their own Arduino based systems in lab sessions. 95% of students found the lab sessions interesting and 85% enjoyed the lecture demos. This increased interest translated directly into a 32% increase in the number of students achieving a good programming level, and a 21% increase in their confidence in programming over previous courses that did not use Arduinos. The investigation has only been applied to one course of one degree, however the results are promising for the usefulness and integration of Arduinos into other courses.

Jamieson[14] investigated the use of Arduinos in an Engineering course designed to teach programming and microcontrollers to students. The students had to present a midterm and final project based on a microcontroller with

90% of the class choosing Arduinos. In his study Jamieson found that the projects produced with Arduinos were far more complicated than those produced in previous years where Arduinos were not available. This shows that due to their ease of use, Arduinos are perfect for quickly protoyping advanced systems, and that students can learn electronics concepts faster with them.

The LilyPad Arduino[5][6] is a project that sought to engage the imagination of students through the design and implementation of wearable computing devices and e-textiles, based on a customised Arduino board. Buechley et al.[5][6] designed the board and a summer camp style programming and development course. They were able to not only engage students (including a large proportion of females), but were also able to stimulate interest in computing and programming, with some students expressing interest in taking further courses in wearable computing and 3 out of 8 students expressing interest in programming and using electronics at home.

Microcontroller robots have also been used to teach programming[1][10]. Additionally visual programming languages[4][12][17] such as Scratch have been modified to run on the Arduino.

2.4 Discussion

Programming skills are fast becoming a necessity in the modern workplace, from designers developing interactive prototypes to developers designing the software on which many businesses are now based, because of this those with programming experience are in high demand[25]. But despite the prospect of lucrative careers, many are turned off by the prospect of programming finding it difficult and lacking the motivation to pursue a career in Computer Science.

Two possible solutions which have been shown to capture the attention and interest of students are the use of games (whether to teach concepts or used as assignments) and the use of microcontrollers such as the Arduino.

Integrating games in the curriculum increases students' motivation, and when given a choice they overwhelmingly choose game-based assignments over traditional assignments, considering them to be more fun, and even less hard (despite examining the same content)[7][8]. It is clear that integrating games into Computer Science courses is an important step towards engaging modern students, many of whom play games regularly. Integrating games into the curriculum could have some problems, namely alienating of people who don't play games (and in particular women), however by providing multiple assignment choices[7][8] this problem can be mitigated. Furthermore, many women today play games[21] and so will not be alienated by game-based content in courses. Developing two (or more) types of assignments for each assignment can however be taxing on the lecturers, who often don't have much time.

Another development in motivating students towards Computer Science education is the use of hardware in the course. Hardware such as microcontrollers and in particular the Arduino have become highly popular in recent years, thanks to the demand for connected and smart devices comprising the Internet of Things[19]. Being able to see physical results motivates students when learning to program, and students find the use of Arduinos in lab sessions interesting[28]. Furthermore Arduinos can be used to introduce programming to students who have never programmed before, such as in the LilyPad projects[5][6]. Despite being extremely attractive as an educational aide and being low cost, buying a large number of microcontrollers can be an expensive exercise for Computer Science departments to undertake.

A new potential solution to providing motivation to students in Computer Science would be to integrate both games and microcontroller programming. A game framework, that provides a scaffold for students to fill in, that links to an Arduino controlled game console could provide the best features of game-based assignments and microcontroller programming. Giving students an opportunity to build their own games, as well as program on the Arduino, will help them learn skills that are valuable in the modern workplace.

3. SYSTEM DESIGN

The educational framework designed for this paper, Playduino, includes a number of components including a hardware game console and spec, a C++ game engine that abstracts away much of the complexity of working with the hardware, documentation for the game engine, 2 sets of assignment briefs for two example games and two complete games.

3.1 Hardware Game Console

The Playduino Game Console is a basic handheld game console that intergrates a number of hardware components and is powered by an Arduino Uno board. The Arduino board consists of a number of pins used for connecting to hardware components, including dedicated pins for power output and ground. The pins can either receive binary input, or output binary data in the form of High and Low voltages. While the Arduino can also accept analogue input it was found that binary input and output was sufficient for the Playduino engine.

The console was built on two breadboards to allow for the size of the screen, and all components were connected to the power supply with resistors to prevent burnout. The schematics for the game console can be seen in Figure 1[1]

The console consists of a screen in the form of an 8x8 LED matrix on which the game screen is presented and the game interacted with. The screen is a grid of 8 by 8 red LED's which can be turned on by aligning a row with a High voltage and the corresponding column with a Low voltage, the remaining rows should have a low voltage and the remaining columns should have a high voltage. The matrix is able to display complex patterns by quickly flashing pixels using this method, as the pixels appear to be continuously on. One of the earliest difficulties with designing the console was the matrix's lack of a clear indication of which pins corresponded to which rows and columns on the matrix, and after some experimentation an array was built which allows conversion of any length 16 array to display correctly on the matrix. The screen is powered by two 8-bit shift registers which enable the screen to be controlled by only 3 pins, despite requiring 16 input pins. This was necessary as it became clear initially that the number of pins available on the Arduino were two few to allow both the matrix and enough input devices. The 8-bit shift registers allow output from three pins to be written to sixteen pins by writing a 16-bit integer bitwise to the register, due to memory limitations on

[1]Also available at https://github.com/mitchmyburgh/playduino/blob/master/Documentation/circuitDiagram.png?raw=true

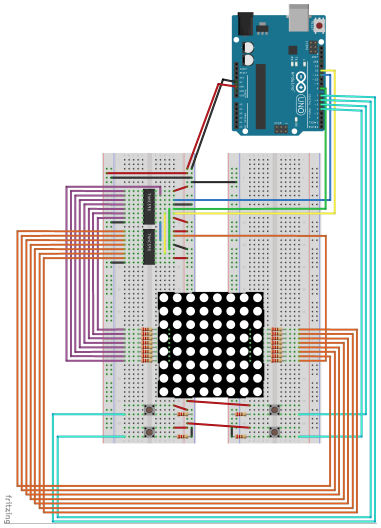

Figure 1: The circuit diagram for the Playduino Console

the Arduino, a simpler approach of representing the integer as an array was taken.

The Game Console was initially designed to be controlled by an infrared remote control, however after the initial feasibility demonstration it was pointed out that the remote sacrificed the handheld nature of the console, and it was decided to replace it with four buttons. The buttons were connected directly to the Arduino Board, however due to the grid-based nature of the breadboard were laid out in a slightly intuitive way: with up and down on the left and left and right on the right, however many of the testers found this arrangement usable after a short time. The layout of the buttons can be seen in Figure 1.

3.2 Game Engine

The Playduino game engine[2] was designed as a set of C++ classes that can be used piecemeal in games built for the Playduino Console. The engine sought to abstract away the hardware implementation by providing users with functions that they might expect from a game engine. The engine is designed for use with the Drop Dodge game, a game about dodging falling enemies and Snake, an implementation of the classic snake game, but can be used with a variety of games.

The game engine was written in C++ because the Arduino supports C and C++ code natively. The implementation was kept simple as the target of the educational framework was first year students, many of whom have no prior experience programming.

The two hardware classes, `Screen` and `Button`, provide methods for controlling the screen and the buttons. Both require that a `loop` method be called inside the main Arduino loop for taking input (in the case of the Buttons) and playing an animation when the game starts and ends (in the case of the screen).

The `Screen` class is by far the largest class and handles interfacing directly with the 8-bit shift registers controlling the LED matrix. It provides methods for drawing points and matrices to the screen, as well as for playing the built in animation at the start of the game and when the player "dies". The screen class handles converting an array of on and off values for the rows and columns on the matrix, to an array that represents the actual pin layout of the screen, additionally it handles writing this array to the shift registers. This allows students to write designs and games to the screen without needing to know how the screen or shift registers work, making it ideal for use in programming focused courses, while the class itself can be used as an example of interfacing with hardware for hardware focused courses.

The `Button` is the second hardware class and allows the user to interface with the buttons on the console. It abstracts the workings of the buttons and allows the user to listen for button presses and button holds.

The remaining classes represent Entities, Enemies, Powerups and Snakes; which can be used for building the game. Entities are objects which move around the game screen, and in the example games the class is used to represent the player, the entity is represented by a single point due to the size constraints of the screen and includes methods for movement and collision detection. The Powerup class represents a single point item which can be picked up, it cannot move, but it contains methods for collision detection.

The Enemy class extends the Entity class for creation of enemy objects. This includes a modified move method and constructors. C++ inheritance is different from Java inheritance that students are used to and as such the class can be used without a direct understanding of the implementation, however a C++ course may use this as an opportunity to introduce inheritance within the Playduino Educational Framework.

The Snake class represents a linked list of Entities which can be used to introduce linked lists. While the student is not directly exposed to the design of the linked list class, when building a snake game they will need to navigate the linked list which can be an opportunity to introduce linked lists. The snake provides movement methods which propagate the movement down the snake allowing the snake to move as it does in other snake games. Additionally it provides a single method to add a node to the tail of the snake. When drawing the snake it is necessary to navigate the linked list and draw each segment.

One issue encountered in the development of the Playduino game engine was the lack of memory on the Arduino board. Initial plans included writing the score to the screen by flashing individual letters and numbers. This was removed as after writing the matrices for the letter and number representations it was found that the Arduino had only a small amount of memory left and stability issues were encountered. This issue was mitigated through the use of a modular game engine, students can import only the classes they need, and with most classes below 100 lines of code, this leaves the student with space to get creative.

Another issue encountered was that it became necessary to use pointers, as issues with stability were encountered when using plan C++ objects. While this make be potentially confusing for students, understanding the complexity of pointers is unnecessary and the students can simply replace the point (`.`) with an arrow (`->`). Additionally this enabled students to make use of familiar object instantia-

[2]Available at the Author's GitHub profile at the following link https://github.com/mitchmyburgh/playduino/tree/master/game_engine

tion, such as using the `new` keyword which they are familiar with from Java.

3.3 Documentation

The documentation[3] provided with the assignments covers the hardware, each of the classes and a primer in C and Arduinos for students unfamiliar with them.

The classes are covered in detail with each class having their function definitions and well as a usage example for each function. A larger usage example for each class is included at the beginning of the class's documentation to allow students and users to quickly get up to speed with some of the functions of the classes and how they might be used together.

The primer describes the hardware in the Playduino game console, listing each component along with a short description. Additionally it covers the Arduino IDE, covering main interface elements and guiding the user through setting up the Arduino. It additionally includes a C primer which gives the format of the main control structures in C, as well as introducing pointers. Students with a background in some "curly brace" programming language, such as Java, should find the information concise and useful, although it should not be considered exhaustive and may not be suitable for absolute beginners.

3.4 Game Examples & Assignments

Two game examples[4] built using the Playduino Educational Framework are provided, namely Drop Dodge and Snake. They provide the base for the two assignments, however can be modified to produce new games, or assignments focusing on different learning areas. The games highlight basic control structures and illustrate an implementation of the Playduino game engine. The two sample assignments[5] included with the Playduino Educational Framework are based on the two example games. Both assignments come in two forms: an abridged assignment which is ideal for introducing Playduino in a class or lab session, or to be used as a practical test; and a longer assignment designed to be completed as a week long class assignment. The abridged assignments were used in the Playduino user testing due to time constraints, and involve users completing the two games with maximal scaffolding, only requiring students to fill in a number of lines of code. In the full assignments, minimal scaffolding is provided giving the students the flexibility to design their own solutions to the questions, and build a game almost from scratch.

3.4.1 Drop Dodge

Drop Dodge is a game about dodging falling enemies. It is considered the simpler of the two games, and is controlled by only two buttons. Of note is the use of the Enemy class which illustrates inheritance. In the game the player moves left and right to dodge falling enemies, while points are calculated each time the enemies move, and displayed in the console on death. The game is simple as an example of a basic game, and as such a number of modifications can be introduced, for example, the enemies move down one block every second (this allows the student to play the game for enough time to ensure their code is correct, without instantly dying), which could be modified to a scaled speed based on a level. Drop Dodge makes use of the Screen, Button, Entity and Enemy classes. The abridged assignment is a basic series of questions which test users on concepts such as object instantiation and basic control structures, as well as giving them the opportunity to work with an API. The abridged assignment provides maximal scaffolding and can be completed in a short amount of time. The full assignment contains minimal scaffolding and allows the student to demonstrate their ability to build a larger project, making use of objects, control structures and the Playduino game engine. The students have access to the comprehensive documentation which gives them the opportunity to gain skills in reading documentation and integrating it into a project on a small scale. Extensions to this assignment are possible, one suggestion is the use of the Enemy class to introduce or test students understanding of inheritance and related topics.

3.4.2 Snake

Snake is the more complex of the two games and makes use of the Snake class. It involves moving a snake around the screen, picking up powerups which grow the length of the snake. The snake is represented as a linked list, which can be used to teach students basic data structures. The game also includes a score which is the length of the snake and is displayed on the console when the snake dies by "eating" itself. The game can be modified to be harder by moving the snake faster based on its length or introducing walls or other collision points, or even basic AI snakes. Snake makes use of the Screen, Button, Powerup and Snake classes. The abridged assignment covers basic concepts such as object instantiation, control structures and using the Playduino game engine. In addition the linked list snake required students to write a simple linked list traversal algorithm. The full assignment provides minimal scaffolding and gives students an opportunity to build a larger game, expanding on the skills they would use from the abridged assignment. Extensions to the Snake assignment could involve requiring students to write their own Snake class, giving them the opportunity to learn not only linked list design but also class design.

4. EXPERIMENTAL DESIGN & EXECUTION

The experiment designed in this paper seeks to investigate the use of Arduinos and games as programming assignments to increase motivation of students completing these assignments.

The initial plan was to perform tests with first-year students, in which they completed the abridged assignments and fed back data in to form of a motivation survey as well as through informal interviews. However due to protest action occurring on the University of Cape Town campus in the fourth quarter of 2016 this became untenable. After consultation with the project supervisor, it was decided that the best course of action would be to perform an expert evaluation, in which experts would walk through the assignment and evaluate it. The experts were chosen from among the students in the Honours and Masters programs in the Com-

[3]Available at the Author's GitHub profile at the following link https://github.com/mitchmyburgh/playduino/wiki/Playduino-Documentation

[4]Available at the Author's GitHub profile at the following link https://github.com/mitchmyburgh/playduino/tree/master/examples

[5]Available at the Author's GitHub profile at the following link https://github.com/mitchmyburgh/playduino/wiki/Playduino-Documentation#assignment-briefs

puter Science Department at the University of Cape Town, and all the Honours students had tutoring experience, while the Masters students were teacher's assistants (TA's). This provided an opportunity to get feedback from experts who work with and understand the problems faced by first year Computer Science students, and as they marked and assisted with assignments also had expert knowledge of the current state of assignments available to students. In addition all the students had completed first year Computer Science at the University of Cape Town so were able to evaluate the Playduino Educational Framework from not only the perspective of an expert but also as a student.

Three rounds of testing were performed. A pre-pilot test completed with two University of Cape Town students, was performed when it became apparent that the university would be closed for an indefinite amount of time. This test assisted in ensuring everything worked together smoothly and that the test components were feasible, feedback from this stage was implemented into the Framework and test structure. Following this a Pilot test was completed with the expert users and after incorporating feedback from this test a final test was conducted with the same expert users. Detailed information about the individual tests is provided below.

The tests were laid out in such a way that allowed the participants to experience the Playduino Educational Framework and then provide comments on it. All participants were first asked to complete a Motivation Survey based on an assignment they had set, marked or done in the past. This would provide a point of reference for comparing the results with regard to the framework. Participants were then asked to read the primer and assignment and to complete the assignment. The abridged assignments were chosen due to time constraints, however these give an indication of the content on the larger assignments as well. Throughout this process the participants were asked to verbalise any feedback, comments, suggestions or difficulties they had while completing the assignment, and were additionally timed to ascertain if indeed this assignment could be set during the class or tutorial sessions suggested. The comments were noted down and was the main source for changes to the framework throughout the tests. Following this participants were asked to complete the motivation survey with regards to the Playduino Framework they had just used. These results are tabulated in the results section. Throughout the process the researcher recorded comments and provided assistance (as a tutor would) to the participants.

The Experts in the first and second round of testing were asked to complete the `Drop Dodge` game, while the third round focused on the `Snake` game. The main content of the primer and documentation was unchanged between the two assignments (aside from the changes indicated as necessary by the experts in the previous round) and the only difference was the assignment, this was in order to ensure that the game content of the initial assignment did not result in an anomalous result and instead that motivation survey results were indicative of the success of the framework as a whole and not a single assignment.

The survey was based on Keller's Instructional Materials Motivation Survey as suggested by Mcgill[23] which is designed to evaluate the motivation of students with regard to educational materials. The questionnaire consists of 36 questions, spanning 4 sections used to quantify motivation and makes use of a Likert scale. The four sections are Attention, Relevance, Confidence and Satisfaction. The survey was modified focusing questions on the assignments rather than general course material. The full 36 question survey was given to evaluate both a past assignment and the Playduino assignment, and in both cases the questions were identical to ensure a correct comparison could be made. In addition four questions were asked following the Playduino assignment in order to evaluate the participants thoughts regarding the use of Playduino, games and Arduinos in the course, as well as indicating whether they thought the larger assignment would also motivate students. The Likert scale goes from Strongly Disagree (1) to Strongly Agree (5), and the statements given are a mix of positive and negative statements. This ensures that the survey does not have a positive bias, however requires some questions to be reversed when calculating averages.

In terms of ethical issues surrounding the tests, all participants were given an ethical clearance form to sign in order to indicate their willingness to participate in the test and have their data recorded. The test does not involve anything that could potentially harm or inconvenience the participants, and all were willing to participate. In addition all participants were compensated for their time with a chocolate.

4.1 Pre Pilot Test

When it became clear that the University would be closing indefinitely, the author obtained two students from the Computer Science Department in order to do an initial evaluation of the educational framework and to ensure that, given the high probability of a condensed period of user testing, that the framework and assignments were doable and no major errors existed. The tests were completed on the 27 September 2016. The students completed the Drop Dodge game assignment. Changes indicated by the students were integrated into the framework prior to the next round of testing. A summary of the comments is available in the results section along with times and survey results.

4.2 Pilot Test

Following discussions with the project supervisor, Gary Stewart, it was decided to use expert users for the testing, as they would be more accessible during the ongoing protests, than first year students. Two Masters TA's, and three Honours tutors were recruited, and asked to complete an expert evaluation of the Drop Dodge game, as well as making use of the documentation as a student would. The testing was completed on the 10th and 11th of October 2016. The changes suggested by the experts were integrated into the framework prior to the next round of testing. A summary of the comments is available in the results section along with times and survey results.

4.3 Final Test

The final test was completed with the same experts as in the pilot test during the week of 17 October 2016. The experts were asked to complete the slightly more complex Snake game. Changes suggested by the experts were integrated into the final version of the educational framework available on the author's Github[6], and a summary of the comments, and survey results are available in the results section.

[6]https://github.com/mitchmyburgh/playduino

5. RESULTS

The results of the three rounds of testing are presented separately below. All results are reported with a 95% confidence interval. The results come from the motivation survey, which seeks to evaluate the experts perception of student motivation with regard to the Playduino framework and with regards to an existing assignment. These two facets were compared with a t-test to ascertain if there is a statistically significant difference between the results. Additionally results from specific questions, in which the experts were asked about the use of Playduino, in both its abridged and full forms, Arduinos and games in Computer Science courses, are presented. The time taken for experts to complete the assignment is also presented, as well as a sections detailing comments and suggestions made by the experts.

The surveys are presented with an average result for each of the major sections, Attention, Relevance, Confidence, and Satisfaction. In addition an Overall average is given. These results were calculated as averages, with some values flipped due to the presence of negative and positive questions, this results in 5 being the overall high score and 1 the lower bound. The results are presented for both the Playduino Educational Framework and a previous assignment and compared using a t-test with a significance level of 5%. The t-test column contains *true* if the results had a statistically significant difference and $false$ if not.

5.1 Pre Survey

The pre-survey was posed to all the participants in the survey in order to evaluate an assignment they had experienced in the past (whether they had marked, set or taken), in terms of their perception of student motivation. The results are presented below and in each round of testing was compared to the same survey given about the Playduino assignment.

Table 1: Motivation Survey Results for an existing assignment

Category	Previous Assignment
Overall	3.24 ± 0.16
Attention	3.27 ± 0.23
Relevance	3.32 ± 0.36
Confidence	2.90 ± 0.34
Satisfaction	3.57 ± 0.43

5.2 Pre Pilot Test

The pre-pilot test involved 2 subjects who were Computer Science students.

5.2.1 Time

The average time taken to complete the assignment by the two participants in the Pre Pilot Test was: 21.34 ± 8.79 minutes.

5.2.2 Motivation Survey

Table 2 contains the data from the motivation survey and Table 3 contains the specific question results.

Table 2: Motivation Survey Results for the Pre Pilot Test

Category	Framework	Previous Assignment	t-test
Overall	4.28 ± 0.28	3.24 ± 0.16	true
Attention	4.46 ± 0.39	3.27 ± 0.23	true
Relevance	3.78 ± 0.81	3.32 ± 0.36	false
Confidence	4.17 ± 0.57	2.90 ± 0.34	true
Satisfaction	4.83 ± 0.25	3.57 ± 0.43	true

Table 3: Specific Question Survey Results for the Pre Pilot Test

Question	Framework
Students would be motivated if this assignment was set in class.	4.50 ± 6.35
Students would be motivated if a larger version of this assignment was set in class in which they could build a complete game	5.00 ± 0.00
Students would be motivated if Arduinos were used in their assignments and classes	4.50 ± 6.35
Students would be motivated if they were able to program games as Computer Science assignments	4.00 ± 0.00

5.2.3 Comments

The comments received were positive and suggested that the difficulty of the assignment was not too high. In addition suggestions were made to indicate clearly the origin and coordinate system used on the screen, as well as minor layout changes in the assignment. It was also suggested to include information about how the use of Arduinos might impact students and other users' lives.

5.3 Pilot Test

The pilot test involved five experts, with experience in tutoring and being TA's, two coming from the Masters Computer Science class at UCT, and three from the Honours Computer Science class.

5.3.1 Time

The average time taken by the five experts in the Pilot Test was: 21.56 ± 16.27 minutes.

5.3.2 Motivation Survey

Table 4 contains the data from the motivation survey and Table 5 contains the specific question results.

Table 4: Motivation Survey Results for the Pilot Test

Category	Framework	Previous Assignment	t-test
Overall	3.93 ± 0.15	3.24 ± 0.16	true
Attention	3.77 ± 0.25	3.27 ± 0.23	true
Relevance	4.09 ± 0.30	3.32 ± 0.36	true
Confidence	3.67 ± 0.34	2.90 ± 0.34	true
Satisfaction	4.43 ± 0.23	3.57 ± 0.4	true

5.3.3 Comments

The comments were also positive with one expert commenting that it "feels awesome", and another saying the use

Table 5: Specific Question Survey Results for the Pilot Test

Question	Framework
Students would be motivated if this assignment was set in class.	3.60 ± 1.14
Students would be motivated if a larger version of this assignment was set in class in which they could build a complete game	4.40 ± 0.68
Students would be motivated if Arduinos were used in their assignments and classes	4.20 ± 0.56
Students would be motivated if they were able to program games as Computer Science assignments	4.8 ± 0.56

of pointers was not an issue due to the clear explanation of them in the Primer. Two participants felt that the abridged assignment would be good as an introduction and felt that the full assignment could then be used as a week long assignment. The feedback also indicated that it was necessary to include more usage examples in the documentation, which was implemented and the primer was moved to a separate file because it was common across all assignments and made the assignment text too long. Additionally it was suggested to insert an image of the Youtube video.

5.4 Final Test

The final test was conducted with the same five individuals as the pilot test.

5.4.1 Time

The average time taken by the five experts in the Pilot Test was: 16.63 ± 9.98 minutes.

5.4.2 Motivation Survey

Table 6 contains the data from the motivation survey and Table 7 contains the specific question results.

Table 6: Motivation Survey Results for the Final Test

Category	Framework	Previous Assignment	t-test
Overall	4.19 ± 0.13	3.24 ± 0.16	true
Attention	4.03 ± 0.26	3.27 ± 0.23	true
Relevance	4.24 ± 0.27	3.32 ± 0.36	true
Confidence	4.09 ± 0.25	2.90 ± 0.34	true
Satisfaction	4.60 ± 0.19	3.57 ± 0.43	true

5.4.3 Comments

The comments were also positive for the final test, with one expert commenting "Its good" and finding it easier after having completed the pilot. An expert also commented that the documentation was good and would be useful for students, while another praised the tangible nature of the assignment, commenting that students would enjoy the assignment. Experts also commented that the snake game was a clever way to introduce linked lists.

6. DISCUSSION

Table 7: Specific Question Survey Results for the Final Test

Question	Framework
Students would be motivated if this assignment was set in class.	4.20 ± 0.56
Students would be motivated if a larger version of this assignment was set in class in which they could build a complete game	4.60 ± 0.68
Students would be motivated if Arduinos were used in their assignments and classes	4.40 ± 0.68
Students would be motivated if they were able to program games as Computer Science assignments	4.60 ± 0.68

The results of the expert testing of the Playduino Educational Framework were overwhelmingly positive. When the motivation survey for the Playduino framework was compared with the same motivation survey for a past assignment the experts had experience in, the results showed a statistically significant difference in most cases, showing clearly that Playduino scored higher in the motivation survey than a traditional assignment, across all the categories in the survey. In addition the specific questions showed high scores with a minimum of 3.6 and a maximum of 5.

It is clear that students would have increased motivation if they were set an assignment based on the Playduino Educational Framework. Experts believe that the integration of Arduinos and games in the curriculum and in particular the integration of both in the form of the Playduino framework would increase students motivation and thus stimulate interest in Computer Science, while making it a more fun learning experience which may help to stop falling enrollments and high drop out rates, and encourage more people to enter into the Computer Science field.

Additionally the abridged assignments have been shown to take less than an hour to complete, making them ideal for use in tutorial sessions and practical tests.

The project was successful in its aim of building and testing an educational framework that, through the integration of Arduinos and games, has been shown to provide an improvement to student motivation over traditional assignments currently provided in the Computer Science Curriculum.

While the study was done with a small number of people, the participants in the pilot and final studies were experts who have experience in helping students with and marking assignments and as such are uniquely placed to evaluate the assignments, while the author of this paper believes this makes the results reliable, it would be advantageous to conduct the test again with a larger group of first year students to get an accurate account of their experience with the Playduino Educational Framework. In addition it is also an issue that the pre-pilot test only made use of two students, however it was a useful indication of problems with the assignment, allowing the pilot and final tests to be completed with less issues.

The results show that the integration of hardware and games into a course would be a good step towards increasing

students motivation. While it may be expensive to purchase an Arduino system for each student, or time consuming to construct the Playduino console for them, it may be worth the cost to integrate it into the curriculum. Many of the participants were excited about the opportunity to play with the console, and for many students the Playduino Framework will be their first opportunity to work with hardware, a fast growing domain of Computer Science. While most Computer Science courses are focused on the software side, hardware can be used to provide students with tangible results to the often abstract software they have designed, as well as sparking their imagination and interest in programming and hardware design.

The individual results for each section is discussed below:

6.1 Pre Pilot Results

The participants in the pre pilot study suggested a number of changes including formatting of the assignment and introducing the potential uses and impact that Arduinos could have on the student's life and community. The results were overwhelmingly positive and all the motivation survey results were statistically significantly better than those of traditional assignments, except for Relevance, which received a lower score due to the omission of information about the use of the framework in people's lives. The students both enjoyed the assignment and were enthusiastic about its use in the Computer Science curriculum, as well as the integration of games and Arduinos in the curriculum. While they did the Drop Dodge abridged assignment, they felt that the use of the larger assignment would also increase motivation.

6.2 Pilot Results

The pilot study results were positive with the motivation survey showing statistically significantly better results than traditional assignments in all categories. High scores also indicated the experts' feeling that the assignment, a bigger version of the assignment, Arduinos and games would increase student motivation.

6.3 Final Results

The experts in the final study were asked to complete the Snake abridged assignment. The scores were once again high on all accounts, with higher average scores for all categories than the pilot test (although the scores have overlapping confidence intervals). This indicates that the Playduino Framework can be used to increase student motivation and that the results do not speak for a single assignment, but are indicative of the framework as a whole.

7. CONCLUSIONS

With the high demand for Computer Science graduates in the modern workplace it is imperative that students become engaged and interested in Computer Science. The perceived and actual difficulty, high drop-out rates and lack of motivation of students need to be addressed in order to stem the tide of decreasing Computer Science enrollments.

A number of possible solutions exist, the two most promising being the integration of games into the curriculum, which has been shown to vastly increase student involvement and interest, even reducing the perceived difficulty of the course, as students consider game-based assignments to be easier than traditional assignments; and the use of microcontrollers, in particular the open-source, low cost, Arduino boards, which also have piqued the interest of students and allow them to see physical results from their 'virtual' code, while also giving them experience in the design of electronics.

The Playduino Educational Framework integrates both the Arduino microcontroller and games in the form of a game console and game engine that has been shown by expert evaluation to increase student motivation. The aim of this project was to investigate the link between the use of games, hardware and student motivation, and, as the results show there is an overwhelmingly positive increase in potential student motivation when using Playduino over traditional Computer Science assignments. While the study had to be completed with Honours and Masters students due to the protest action at the University of Cape Town, they were all tutors and TA's, able to provide expert evaluation due to their work with students.

Playduino provides a framework around which to build Computer Science assignments, as well as two assignments that have been developed through a number of usability and expert testing procedures. The game engine provides a number of classes that could be used to build a variety of games, and while the console may look complicated, it is relatively simple to put together with the diagrams and instructions provided. Playduino provides a range of uses and can be used in a "programming only" course, in which prebuilt consoles are provided, a hardware course in which students build their own consoles, or a mixed course. Additionally various levels of scaffolding are provided in the assignments, and can be modified by the instructors to focus on specific learning areas or to allow a more free form design process. Playduino would be an ideal component to any Computer Science course that wishes to introduce new topics, or evaluate old ones, in a way that is tactile, exciting and new.

8. FUTURE WORK

While the expert evaluation was sufficient and informative, future work could center around testing the Playduino Educational Framework with first year Computer Science students, as they are the target for the framework. In addition tests can be completed at other universities besides the University of Cape Town. Playduino is designed to be merely a framework, with which lecturers can design and run their own assignments and exercises, future work could involve extending the framework by adding more games and assignments to test different facets of Computer Science knowledge, as well as by extending the provided games into fuller and more complex projects. Finally Playduino is designed to be used in first year Computer Science courses and as such a wealth of data could be gleaned from its actual implementation in a curriculum.

9. ACKNOWLEDGMENTS

The author of this paper would like to thank Gary Stewart for his input and assistance with the project, as well as the research participants.

10. REFERENCES

[1] S. Alers and J. Hu. Admoveo: A robotic platform for teaching creative programming to designers. In *Learning by playing. Game-based education system*

design and development, pages 410–421. Springer, 2009.

[2] J. D. Bayliss. Using games in introductory courses: Tips from the trenches. In *ACM SIGCSE Bulletin*, volume 41, pages 337–341. ACM, 2009.

[3] J. Bennedsen and M. E. Caspersen. Failure rates in introductory programming. *ACM SIGCSE Bulletin*, 39(2):32–36, 2007.

[4] T. Booth and S. Stumpf. End-user experiences of visual and textual programming environments for arduino. In *End-User Development*, pages 25–39. Springer, 2013.

[5] L. Buechley and M. Eisenberg. The lilypad arduino: Toward wearable engineering for everyone. *Pervasive Computing, IEEE*, 7(2):12–15, 2008.

[6] L. Buechley, M. Eisenberg, J. Catchen, and A. Crockett. The lilypad arduino: using computational textiles to investigate engagement, aesthetics, and diversity in computer science education. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 423–432. ACM, 2008.

[7] D. C. Cliburn. The effectiveness of games as assignments in an introductory programming course. In *Frontiers in Education Conference, 36th Annual*, pages 6–10. IEEE, 2006.

[8] D. C. Cliburn and S. Miller. Games, stories, or something more traditional: the types of assignments college students prefer. *ACM SIGCSE Bulletin*, 40(1):138–142, 2008.

[9] R. Daniels. Skills shortages in south africa: A literature review. *DPRU Working Paper*, (07/121), 2007.

[10] O. Dziabenko, J. García-Zubia, and I. Angulo. Time to play with a microcontroller managed mobile bot. In *Global Engineering Education Conference (EDUCON), 2012 IEEE*, pages 1–5. IEEE, 2012.

[11] A. Gomes and A. J. Mendes. An environment to improve programming education. In *Proceedings of the 2007 International Conference on Computer Systems and Technologies*, page 88. ACM, 2007.

[12] N. Gupta, N. Tejovanth, and P. Murthy. Learning by creating: Interactive programming for indian high schools. In *Technology Enhanced Education (ICTEE), 2012 IEEE International Conference on*, pages 1–3. IEEE, 2012.

[13] D. J. Jackson and P. Caspi. Embedded systems education: future directions, initiatives, and cooperation. *ACM SIGBED Review*, 2(4):1–4, 2005.

[14] P. Jamieson. Arduino for teaching embedded systems. are computer scientists and engineering educators missing the boat? *Proc. FECS*, pages 289–294, 2010.

[15] T. Jenkins. On the difficulty of learning to program. In *Proceedings of the 3rd Annual Conference of the LTSN Centre for Information and Computer Sciences*, volume 4, pages 53–58, 2002.

[16] K. Kahn. A computer game to teach programming. In *Spotlight on the Future, NECC '99. National Educational Computing Conference Proceedings*. ERIC, 1999.

[17] Y. Kato. Splish: a visual programming environment for arduino to accelerate physical computing experiences. In *Creating Connecting and Collaborating through Computing (C5), 2010 Eighth International Conference on*, pages 3–10. IEEE, 2010.

[18] M. Koorsse, C. B. Cilliers, and A. P. Calitz. Motivation and learning preferences of information technology learners in south african secondary schools. In *Proceedings of the 2010 Annual Research Conference of the South African Institute of Computer Scientists and Information Technologists*, pages 144–152. ACM, 2010.

[19] H. Kopetz. Internet of things. In *Real-time systems*, pages 307–323. Springer, 2011.

[20] S. Kurkovsky. Engaging students through mobile game development. In *ACM SIGCSE Bulletin*, volume 41, pages 44–48. ACM, 2009.

[21] S. Leutenegger and J. Edgington. A games first approach to teaching introductory programming. In *ACM SIGCSE Bulletin*, volume 39, pages 115–118. ACM, 2007.

[22] M. McCracken, V. Almstrum, D. Diaz, M. Guzdial, D. Hagan, Y. B.-D. Kolikant, C. Laxer, L. Thomas, I. Utting, and T. Wilusz. A multi-national, multi-institutional study of assessment of programming skills of first-year cs students. *ACM SIGCSE Bulletin*, 33(4):125–180, 2001.

[23] M. M. McGill. Learning to program with personal robots: Influences on student motivation. *ACM Transactions on Computing Education (TOCE)*, 12(1):4, 2012.

[24] M. Muratet, P. Torguet, J.-P. Jessel, and F. Viallet. Towards a serious game to help students learn computer programming. *International Journal of Computer Games Technology*, 2009:3, 2009.

[25] B. E. Nzimande. List of occupations in high demand: 2014. *Department of Higher Education and Training*, 2014.

[26] S. O'Donovan, J. Gain, and P. Marais. A case study in the gamification of a university-level games development course. In *Proceedings of the South African Institute for Computer Scientists and Information Technologists Conference*, pages 242–251. ACM, 2013.

[27] M. Prensky. Digital game-based learning. *Computers in Entertainment (CIE)*, 1(1):21–21, 2003.

[28] M. A. Rubio, C. M. Hierro, and Á. Pablo. Using arduino to enhance computer programming courses in science and engineering. In *Proceedings of the EDULEARN13 Conference*, pages 5127–5133, 2013.

[29] K. Sung. Computer games and traditional cs courses. *Communications of the ACM*, 52(12):74–78, 2009.

[30] K. Sung, M. Panitz, S. Wallace, R. Anderson, and J. Nordlinger. Game-themed programming assignments: the faculty perspective. In *ACM SIGCSE Bulletin*, volume 40, pages 300–304. ACM, 2008.

[31] I. Woolard, P. Kneebone, and D. Lee. Forecasting the demand for scarce skills, 2001–2006. *Human Resources Development Review*, pages 458–474, 2003.

[32] M. Zapušek and J. Rugelj. Learning programming with serious games. *Game-Based Learning*, 1:13, 2013.