UNIVERSITY OF CAPE TOWN

DEPARTMENT OF COMPUTER SCIENCE

Computer Science Honours
Final Paper
2015

Title: PyTeacher – Determining the Usefulness of Visual Learning for Computer Science

Author: Eugene de Beste

Project Abbreviation: PyRob

Supervisor: Gary Stewart, James Gain

Category	Min	Max	Chosen
Requirement Analysis and Design	0	20	10
Theoretical Analysis	0	25	0
Experiment Design and Execution	0	20	10
System Development and Implementation	0	15	10
Results, Findings and Conclusion	10	20	15
Aim Formulation and Background Work	10	15	15
Quality of Paper Writing and Presentation	10		10
Adherence to Project Proposal and Quality of Deliverables	10		10
Overall General Project Evaluation (*this section allowed only with motivation letter from supervisor*)	0	10	0
Total marks		**80**	**80**

PyTeacher: Determining the Usefulness of Visual Learning for Computer Science*

Eugene de Beste
Researcher
University of Cape Town
Rondebosch
Cape Town, South Africa
debeste.eugene@gmail.com

Gary Stewart
Supervisor
University of Cape Town
Rondebosch
Cape Town, South Africa
gstewart@cs.uct.ac.za

James Gain
Co-supervisor
University of Cape Town
Rondebosch
Cape Town, South Africa
jgain@cs.uct.ac.za

ABSTRACT

This paper covers the design and implementation of an educational video game-like tool with the aim of assisting students in visualising source code. Accomplishing this involved having students in a first year Computer Science class at the University of Cape Town take part in various surveys. The initial survey revealed that students were interested in puzzle games. These types of games are conducive to critical thinking and learning and as such was adopted as the genre for the tool, in the form of mazes. Students had to use real Python 3 source code in order to navigate the mazes. Results from user testing sessions indicated that students were very interested in having such a tool accompany the traditional course, but that improvements should be made. Students also found the designed levels to be stimulating and challenging. Overall, the results showed that such a tool has potential use in the classroom and also has the potential to enrich the education experience of students. Larger scale study is recommended.

CCS Concepts

•Social and professional topics → Computational thinking; CS1; Computational science and engineering education; *Adult education;* **•Applied computing → Interactive learning environments; Computer-assisted instruction;** *Computer games;* **•Software and its engineering → Virtual worlds software; Interactive games; Virtual worlds training simulations;**

Keywords

Programming; Computer Science; Education; Video Games; Edutainment

1. INTRODUCTION

1.1 Problem Statement

In a world that is constantly becoming more interactive, it is often found that education systems and methods are lagging behind. The conventional ways of teaching students how to program, through textbooks and theory, may no longer be the best way to approach the problem. It has been noted that students today find more pure theory driven classes and courses to be "boring" and otherwise uninteresting [14]. This results in a potential lack of engagement by students of all fields, including students in Computer Science.

1.2 Proposed Solution and Motivation

Video games have become a major part of everyday life for many students and others alike. With the rise in popularity and accessibility of mobile devices, such as smartphones, we see more people of all ages exposed to games and digital technology in general [8, 3, 5].

The proposed solution is a packaged video game system that allows users to input real Python code and have this code translate to actions in the game. This would allow students, particularly new Computer Science students with little to no prior programming knowledge, to potentially gain a better understanding of what their code does through a visual medium with immediate feedback.

Not only do we believe that this would assist in understanding, but it would also allow for students to be more engaged by providing them with an interactive environment which promotes problem solving and critical thinking skills.

1.3 Research Questions

There are two major research questions that this experiment seeks to answer:

- Are new Computer Science students interested and willing to have such a video game or interactive simulation present in their conventional courses?
- Would a video game or interactive simulation such as the one proposed bring any value to the students that are taking an introductory Computer Science course and enhance their learning experience?

1.4 Work Allocation

There are two major parts to this system. One is the video game back-end, which contains all the logic for how the video game should operate. The back-end also includes designing and implementing the levels (stages or obstacles) that need to be traversed. The other side to the system is the front-end, which includes the tools that allow for user interaction and the look of the end product as the end-user views it. Both of these parts are critical to the running and testing of the system. The back-end was worked on by Eugene de Beste while Mark Grivainis focused on the front-end.

*Appendicies are available at http://pubs.cs.uct.ac.za under Honours Project Archive/2015/PyRob. The project website contains a page dedicated to the appendices which can be found on the top right.

1.5 Paper Outline

Section two will focus on detailing the background to the topic of research, along with the technologies and methods that are currently employed in the attempt to make learning Computer Science more engaging. Section three details the system and experimental design approach that was used for this research topic. Section four explains the implementation and its challenges. Section five reports on the experiments that were conducted, as well as their findings. Lastly, section six notes any conclusions discovered through the research and discusses the impact.

2. BACKGROUND

A fair amount of research has been put into the introduction of games as part of education, specifically in the Computer Science field. This section covers the reasons why this may be necessary, what implementations exist today and the impact of the research and implementations.

2.1 Computer Science Education at Universities

Most of the South African universities offering Computer Science as a subject to study introduce students to various programming related concepts by making use of the Python programming language. This involves a curriculum tuned to the international Association for Computing Machinery/Institute of Electrical and Electronics Engineers (ACM/IEEE) standard, in which students are required to study a specific set of topics[1]. The ACM/IEEE curriculum provides guidelines on how the topics should be taught, but it does allow for a fair amount of freedom as well.

2.2 Efforts to Improve Education

It has been noted that there is an increasing lack of interest and drive in students to study. In an ever-increasing digital age it becomes more difficult to learn through traditional methods. This has lead to efforts in developing more engaging ways to teach. A much more practical driven approach has been adopted with many Computer Science courses, as a variety of studies show that learning through interaction, exploring or playing is a useful tool in educating [7, 4, 11]. Examples of this include:

- Visual Technology
  - Making use of this allows important aspects of the topic (such as how algorithms work in Computer Science) to be represented to learners in a way that is creative and stimulating.
    In theory, this should allow a richer learning experience by taking advantage of the human sensory system in terms of visuals and sound. It has, however, proven to have its own unique difficulties in describing and controlling animation as well as scaling to larger audiences and more complex data structures [1]. As a result of this most visual technologies have failed to catch on to mainstream Computer Science education, while appearing to be potentially beneficial [6].
- Greater Focus on Applied Theory
  - Some universities have taken to using strong practical application of the theory taught in the course. In most cases students would be given a fairly large task which needs to be completed by utilising the skills taught in the course.
    Game development has been one of the more common types of application, where the skills conveyed to students would have to be applied in order to design and develop a video game [9, 12].
- Video Games in Recent Times
  - Video games have almost always exclusively been seen as a form of entertainment. The adoption of playing these games has boomed in recent times with easier accessibility. Mobile "smart" devices, such as cellular phones, provide seemingly endless amounts of these games and many people, both young and old, play video games on a regular basis.
    This increase in popularity pushed video games into the mainstream, with perspectives on what can be done through it changing. Some games are now considered to be art.

2.3 Video Games in Education

Various attempts have been made to design and implement video games and interactive systems in order to teach people basic and even more advanced programming skills. However, there has been little adoption of the same in the classroom setting for new tertiary education students.

Various implementations of such platforms are more common in the lower education levels, such as primary and high school, and especially in more developed countries such as the United States of America [10, 13].

There have been efforts to introduce video games and game-like systems into education systems of higher levels. Two examples of this, while not specifically meant for tertiary education levels, are:

- CodeSpells
  - Originally a project by graduate students at the University of California, this video game was designed in order to be a self-contained way of introducing people to programming with no additional material required [2]. The idea with this implementation was to immerse the players in programming by providing them with a rich world to engage with. It features a full first-person perspective and real-time elements in which a player creates "spells" to cast. These spells are designed through Java source code. The player writes Java code to create a spell in a "spellbook" and can then use these creations on various elements in the game world.
- Pex4Fun
  - This project takes a different approach to gamifying learning. It is provided as a standalone web service, but can also be integrated into the classroom. Pex4Fun provides a platform called a "coding duel". These duels take place between players, in which a puzzle (a snippet of source

[1]https://www.cs.uct.ac.za/honours/welcome.html

code along with the output of the full program) that one player has made is given to the other and the challenge is to be able to reproduce the output by completing the code [17].

2.4 Educational Games and Their Effects

Considering that there are many vastly different types of video game genres, one can think that not every type may be suited to educating.

Using various other studies, Kowit Rapeepisarn et al. (2008) tabulated the various different video game genres in order to understand which types of genres are conducive to which types of learning [16].

Learning techniques	Leaning activities	Possible game genres
Practice & feedback	Questions, memorization, association, drill, imitation	Game show competition, flashcard type game, mnemonics, action, sports game
Learning by doing	Interact, practice, drill, imitation	Strategy game, action game, role playing game
Learning from mistake	Feedback, problem	Role-play game, puzzle game
Discovery learning & guided discovery	Feedback, problem, creativity play	Adventure game, puzzle game
Task-based learning	Understand principle, graduated tasks	Simulation game, puzzle game
Question-led learning	Question/ questioning, problem	Quiz or trivia game, game show competition, construction game
Situated learning	Immersion	Immersive style game such as role-playing game, flashcard game
Role playing	Imitation, practice, coaching	Role-playing game, strategy game , reflex game, adventure game
Constructivist learning	Experimentation, questioning	Building game, constructing game
Multisensory learning	Imitation, continuous practice, immersion	Game in which introduce new technologies such as locatable sound or force feedback, reflex game
Learning object	Logic, questioning	Games which are becoming object-oriented
Coaching	Coaching, feedback, questioning	Strategy game, adventure game, reality testing game
Intelligent tutors	Feedback, problem, continuous practice	Strategy game, adventure game, puzzle game, reflex game

Figure 1: Kowit Rapeepisarn et al. (2008) The relationship between learning techniques, learning activities and possible game styles

3. SYSTEM AND EXPERIMENT DESIGN

This section explains the high level overview of the approach to designing the system that is used to evaluate the research question.

3.1 Overview

The main aim of the system is to have enough content to answer the research questions presented. This entails designing and implementing a scaled down version of an interactive system in which students can explore programming code that they provide. This is due to time constraints and the nature of the project.

It was decided early on that the implementation processes would follow an agile methodology. The key points to using the agile methodology are promoted regular incremental builds of software as well as regular face-to-face communication. Using revision software, git specifically, both parts were kept in check with one another at all times.

3.2 Design Goals and Restrictions

Given the limited time for the implementation of the system and the nature of the system in question, a bare-bones implementation would suffice. While it would not be implemented to the level of a commercial video game title, it would have all the necessary features and properties in an attempt to answer the research questions. The basic properties of this system include:

- A player avatar which is engaged with through providing Python code.
- A system that takes in Python code from a user and interprets and translates that to reflect visually to the user in the system.
- Various game levels that firstly introduce the player to the various game mechanics in order to prepare them for later, harder levels that challenge the player to think in a programmatic way.
- Basic gamification elements in an attempt to make users feel more engaged and invested in their actions.

3.3 Programming Language

The programming language through which students are introduced to Computer Science concepts can be an important decision. The programming topics and Computer Science concepts that are taught to students, first and foremost, are dependent on which language is used. For example, if the course starts by introducing students to Java, it would inherently require them to understand basic principles of object orientation first. On the other hand, if a more dynamic language such as Python or Ruby is used students can first be introduced to more core concepts such as data types, loops and others [15].

Students that register to study Computer Science at the University of Cape Town are introduced to basic concepts through the programming language Python, version 3.x to be specific. Please note that the notation ”3.x” refers to any revision of Python 3, since all revisions will be syntactically equivalent. Due to this fact, the programming language chosen for student interaction in the system is Python version 3.x.

3.4 Game Engine

The decision on which video game engine to use was based mostly on the programming language that the students would need to learn in. There are several free to use and well developed game engines available, such as Unity and Unreal Engine. While these would have provided better support in terms of some of the built-in functions that they provide out-of-the-box, Blender was chosen.

Blender is an open source 3D creation suite which includes a game engine built with Python 3.x. It is one of the only video game engines that supports scripting through a Python 3 API. This allowed us to directly evaluate Python code that the player would type directly in the engine.

The Blender Game Engine was written in C++ and therefore provides fairly decent performance on a wide range of systems, allowing the system to be distributed more easily if it were to be more complete.

3.5 Coding Format and Design

The development of the system required work to be done in parallel. As such, it was decided that both parts would be developed as independently from each other as possible.

The only common point are the actual levels, or Blender scenes, which acted as a point to bring the front-end and back-end together. As a result of this, with basic level/scene properties defined, it would prove easy for both parts to be implemented on their own and separately tested.

In focusing on such a modular design, git and the popular git website GitHub were selected to be the versioning tools of choice so that both sections could be up to date with each other. Using this method would also prove to cause little effort to have the systems work together.

3.6 Level Design

Two types of level designs need to be applied for the development of a system such as this. One set of levels needs to exist to introduce the user to the mechanics and functionality of the system. The other set of levels needs to be able to challenge the user on the various mechanics and functionality that they have been taught.

A small set of introductory levels were designed in order to show users how to interact with their virtual avatar and how to use the more advanced functions of the system in order to successfully navigate and progress through various scenarios.

Two larger levels were designed in order to combine the mechanics that the users had to learn through the smaller levels. These levels not only build upon what they were introduced to, but also serve to promote programmatic thinking in these users.

3.7 Experiment Design

An experiment needed to be set up in order to test the effectiveness of the system and gather enough resources to attempt to answer the research questions. This experiment required a set of tests to be performed in order to answer the first research question as well as design the system used to elicit the desired user responses for the second.

The type of research attempted by this research project is qualitative. The goal was to achieve an understanding in student interests and opinions on the system that is proposed. Ethical clearance needed to be obtained in order to perform the surveys that students would need to do as well as user testing with the students. This ethics approval was received through the Faculty of Science Research Ethics Committee as well as the Department of Student Affairs, both of the University of Cape Town. All of the participants were kept anonymous. See Appendix A1.

To answer the first question, a basic survey was designed in order to receive the opinions from various students from the first year CSC1010H class of the University of Cape Town. This survey provided information on which types of video game genres students are interested in so that the system could be designed around that. Google was used for its Forms application in order to have a platform with which the surveys could be easily distributed and results aggregated.

After the system was designed it needed to be tested. Two user testing sessions were set up in which participants, while remaining anonymous, were tasked to rate various aspects of the system, as well as give more general feedback. Apart from obtaining information in order to improve the system, the user testing proved to be key to answering the second research question in this experiment and allowed for a sense of understanding of the experiences of users with the system.

Besides the post-session survey, the participants could also be observed to study their reactions and interpretations of the system to gather more contextual information.

4. IMPLEMENTATION

This section details the various aspects of the system and the processes undertaken to achieve its implementation.

4.1 Implementation Process

It began with an initial proof of concept which was put together in order to demonstrate the idea and ascertain whether it was feasible to continue. After that, both developers had regular meetings in order to discuss the direction of both parts of the system.

The system was built in incremental stages with major features being developed, tested and reviewed. After the system was feature complete, two user testing sessions were held. The first one was followed by relatively minor improvements and additions based on the feedback received.

4.2 Design Documents

As with most video game implementations, the process begins with design documents. Two common documents that are associated with video game design are the game and technical design documents. Both of these documents are created before any development begins and while they are created so early, they are dynamic and subject to change as development progresses. A game design document contains information that helps conceptualise the various aspects of the game in a non-technical way. The technical design documents the various details of how the aspects mentioned in the game design document are to be implemented.

Both partners worked on the game design document in order to lay out exactly what elements would be used in the system. This game design document outlined various aspects in terms of the visual representation that the game would take and the type of interaction a user would have with the game.

A technical design document was produced along with the game design document to detail the approach that would be taken in terms of source code. Both design documents served as a good point of reference during the development process and kept the project on track, preventing scope creep.

Apart from the two design documents, a paper prototype was also created with which various scenarios could be visualised and tested in theory. Refer to appendix B1.1.

4.3 System Structure

4.3.1 System Architecture

The architecture was specified in the technical design document. As illustrated by **Figure 2**, the system was designed from the ground up to be fairly modular. None of the modules completely depend on each other and are capable of acting as independent entities, given minor adjustments.

4.3.2 Game Engine

Blender, the game engine used, lies between what the end-user sees and the code that we develop for the game. While Blender itself is primarily a 3D creation suite targeted at designers, it does have a game engine built in which supports scripting through a Python API.

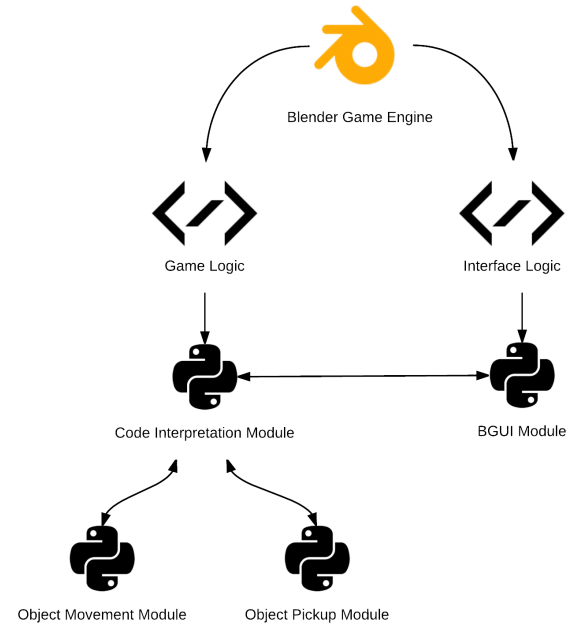

Figure 2: Overview of the game system's architecture

This game engine handles all of the scene rendering (displaying all the objects that constitute a level to the user) and any other game related properties, such as physics simulation.

The blocks of code and scripts that were written for the user interface and game logic are added to the scenes and objects that are created through blender. Blender makes use of what it terms "Logic Blocks", which are comprised of multiple sensors, controllers and actuators that make the scene operate in various ways. The logic for the game and the interface were added to the scenes through the use of these blocks.

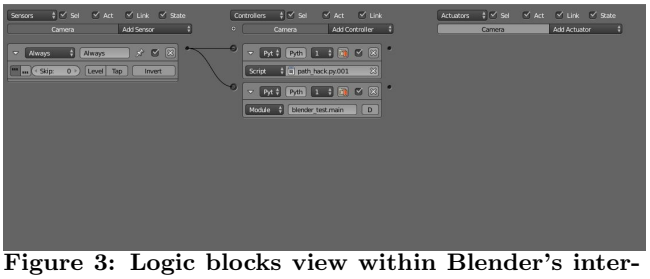

Figure 3: Logic blocks view within Blender's interface

4.3.3 *User Interface Code*

This section of code was done by Mark Grivainis. He made use of a third party plugin for the Blender Game Engine called BGUI[2],which is a free open-source Python library that has Qt bindings in order to provide a graphical user interface with which the player can engage.

Using this, a basic text editor and interaction buttons were placed into all of the Blender scenes in order to allow users to input their code and execute it using meaningful and contextual operations. The user interface system passes the code which the user types to the game logic back-end code, which interprets it as detailed below.

[2]https://github.com/Moguri/bgui

4.3.4 *Python Code Interpretation*

Users are expected to input familiar feeling Python code that has the same syntax as Python to the system in order to achieve objectives and proceed with the game.

In order to keep the syntax as close as possible, the Python 3 "eval" keyword was used in order to compile and execute the given user code during the operation.

```
try:
    codeobj = compile(
                str(self.text),
                '<string>',
                'exec')
    eval(codeobj, globals(), locals())

except Exception as e:
    return str(e)
```

Listing 1: Python Syntax for Code Interpretation

In the above snippet of code the "codeobj" variable stores a compiled code object from the given user input. If there are any syntax problems with the supplied user code it will raise an exception and provide the details of the error, much like Python does normally. If the code compiles successfully it is then evaluated using the eval() function, which is passed the global and local variables being used by the system.

A series of functions were defined in order to act as actions for the user to perform. These functions include commands for moving the game avatar in the four main directions (up, down, left and right), as well as picking up objects.

```
def moveUp():
    tester.move("n")

def pickup():
    tester.pick()
```

Listing 2: Code Snippet for User Input Methods

When the user makes a call to any of the move functions, as well as any other predefined ones, a class method is called from within the interpreter class. This class is passed the appropriate value (such as move North in the above example).

When the user executes their script of Python code, which contains calls to the predefined functions, an array of actions gets built from these functions. Apart from this, the rest of the code supplied is executed directly. This allows users to use any arithmetic operations or loops as they wish.

With regard to the loop creation, a tester method was implemented in order to prevent infinite while loops. Using Python's built-in pattern matching, if a while loop is detected it will automatically insert additional code in order to only allow a finite number of iterations. This method also takes careful consideration of user code indentation.

This method of evaluating a user's source code directly leads to various security concerns. A user can, for instance, write and execute an arbitrary malicious piece of Python code which could solve the level without the intended steps. This, however, would not be a failure and would mean that the user has a good understanding of the system and how to

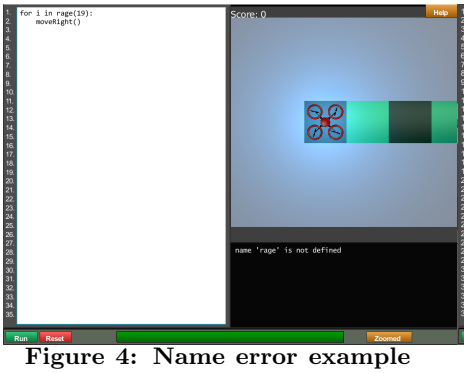

Figure 4: Name error example with incorrect spelling

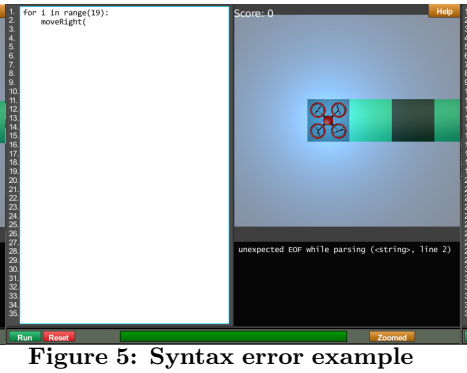

Figure 5: Syntax error example due to missing bracket

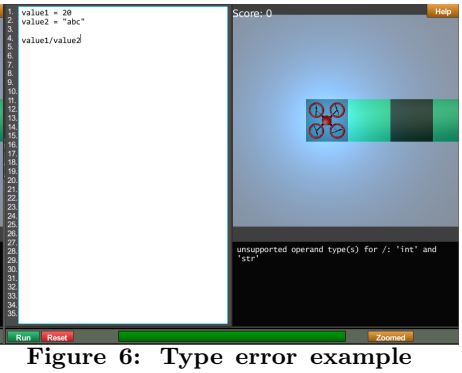

Figure 6: Type error example with integer and string

program. On the other hand, a user could potentially cause damage to the machine that the program is running on. Due to the scope of the project and the experimental nature of it, this was not covered.

4.3.5 *Game Logic*

Since the design of the system was not to achieve commercial title status, the logic for the game is fairly simple. Once the user's code has been evaluated and the array of actions has been built a new loop will execute until this array is empty. In this case the array is being used as a queue, whereby the successful iteration of a move will result in removal of the first item in the queue. If an error occurs in the runtime the array is cleared and the error is returned to the front-end and displayed to the user in a console with the same message that Python would normally raise, illustrated by **Figure 4**, **Figure 5** and **Figure 6**.

There are various switches for the different actions that users can perform and as the queue is traversed, these switches will be triggered and call the appropriate helper class as defined in the architecture overview.

4.3.6 *Levels*

Each of the levels were designed for users to learn the system and eventually apply their knowledge of programming and the system to solve challenges. All of these introductory levels as well as challenge levels were created through Blender's graphical interface. This consists of having multiple scenes within Blender in which there are various objects, such as cubes and planes, that act as surfaces for maze-like structures.

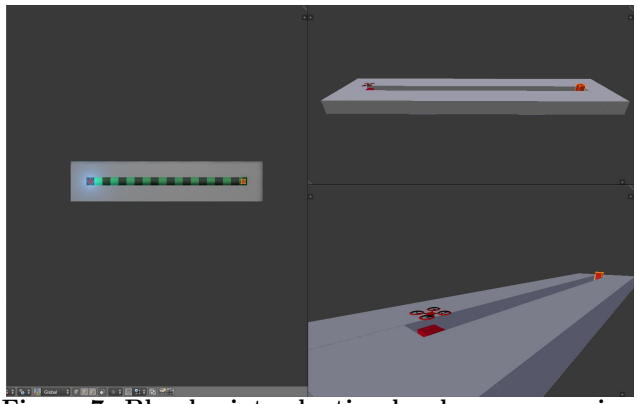

Figure 7: Blender introduction level scene overview

All of the objects in the scenes are given logic through the back-end code, detailed in the section above. Everything that the player sees as an end result is contained in each Blender scene. In order to attach code/scripts to scenes or objects in the game engine, the "logic blocks" are utilized, as detailed in the Game Engine section.

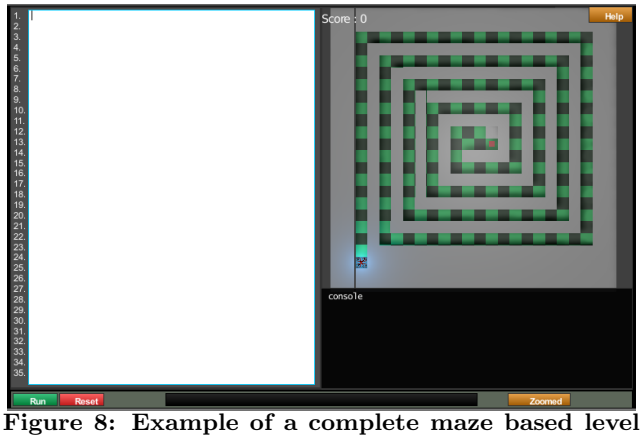

Figure 8: Example of a complete maze based level in the final iteration

5. EVALUATION AND RESULTS

Evaluation was done through prospective user surveys and testing. Students from the first year Computer Science course at the University of Cape Town were asked to attempt to use the system and answer various surveys based on their experiences.

5.1 Feasibility Survey

A survey was held in order to gain an understanding of whether students would be interested in such a tool for assisted learning. Along with this we obtained perspective on what kind of genre students are more or less interested in pertaining video games.

With the initial survey 14 students from the first year Computer Science course submitted responses. The results are as tabulated in **Table 1**:

Question	Yes	No
Do you play video games in your spare time?	12	2
Would you be interested in a video game that assists you in learning your coursework?	14	0
Is this year the first time you have been exposed to programming?	13	1

Table 1: Initial feasibility survey responses

The results from the initial survey were positive. These clearly indicated that most students who were interviewed are involved with video games and that they are all interested in having some form of interactive experience in their learning work. The fact that the overwhelming majority of participants indicated that they were being exposed to programming for the first time perhaps suggests that it is possible that most students who study Computer Science at universities are not exposed to programming concepts before the time.

Along with these general questions, students were also asked which genre of video game they are most interested in, in order for the system to be adapted to be suitable for the majority of students. Each participant was allowed to select multiple genres from the given list. The results are as follows:

Genre	Votes
Action Adventure	8
Fighting	2
First Person Shooter	5
Massive Online Battle Arena	3
Puzzle	9
Racing	13
Role Playing	4
Simulation	4
Sports	8
Strategy	5

Table 2: Genre interest responses from initial feasibility survey

It is clear, from **Table 2**, that many students are primarily interested in racing games. Racing game mechanics, however, do not generally fit with the theme of teaching programming. The fast-paced and twitch-response nature does not work well with the slower pace of sequential execution of code. It was decided that the next highest genre would be used for our system. Along with that, the complexity of implementing a racing game is out of scope for this project. The puzzle setting is one that is quite conducive to learning and critical thinking. This fits well with what the system is attempting to accomplish.

The conclusion here is that students are interested in more modern assisted teaching methods and enjoy puzzle solving.

5.2 User Testing Stage 1

An initial prototype of the systems was developed and user testing was needed to determine major flaws in mechanics and operations. A group of 17 students from the first year Computer Science class at the University of Cape Town were given a build of the system and asked to do a series of tasks. The students were instructed regarding the necessity of playing through the levels of the game in order to understand the mechanics and attempt to solve the challenges.

Once the users successfully completed the levels of the game, they were given a shared Likert scale based survey to complete in which there was a series of ratings from one to five that they needed to provide (one being strongly disagree and five being strongly agree). The survey was intended to answer questions pertaining to both parts of the research project. In addition to the ratings, a comment box was made available to serve as a place where the participants could provide general feedback. All 17 students completed the levels that were presented to them.

Studying the results in **Table 3** provided clear suggestions on what needed to be improved. The participants indicated that they were happy with the concept of the system and that they would like to use something similar in their course. Overall, the participants were also happy with the level of feedback that they received from the system in terms of source code interpretation and visual representation.

Question	Likert Result (Avg.)
Would you have enjoyed playing a game like this as additional material for your first year course?	Strongly Agree
Do you think that the visual feedback is a good indicator to what your code is doing?	Strongly Agree
Do you feel that there should be additional gamification elements?	Strongly Agree
Is this game something that interests you?	Strongly Agree
Is the layout of the user interface elements appropriate/intuitive?	Indifferent
Do you feel that the game challenges your problem solving skills in terms of designing a complete piece of code?	Agree/Strongly Agree

Table 3: Likert scale results of the first user testing session

While we had developed a system that users found enjoyable for the most part, there were flaws. The participants strongly felt that there should be additional gamification elements (Question 3), such as ways to accumulate score and other elements which would make the system feel more like a traditional video game. The user interface was also flawed in its design and interface, with many of the participants disliking it or not finding it very intuitive (Question 5). In terms of the instructional design, participants felt that the system provided more or less the right level of challenge in the various levels that they had to navigate.

The participants provided a lot of feedback through the general suggestions box that was provided to them. One of the most frequent points that were made was that the user interface and experience was not pleasant. Due to various technical glitches and users attempting to use the system in ways that were not foreseen, issues made themselves apparent during the first round of testing. This was something that would need careful attention. The other frequent type of suggestion was of participants suggesting ways to gamify the system more. Many felt that there should be additional mechanics, such as enemies that prevent the players from

reaching their goals or player "lives", which gives players a limited number of tries before failing and having to start at the beginning of the game. Having sound present in the system was also mentioned more than once, due to the fact that there is none.

Refer to appendix C.1 and C.2.

5.3 User Testing Stage 2

Following the first user testing session, improvements were made to the system based on the survey responses and suggestions of the students. The goal for the second round of user testing was to have the same students that completed the first round come back and try the improved system. While attempts were made to achieve this, less students arrived for the second round of testing than the first. Only nine students in total arrived for the second round of user testing.

Along with the same set of levels that the students had to navigate through in the first testing session, an additional level was added to the end of the system.

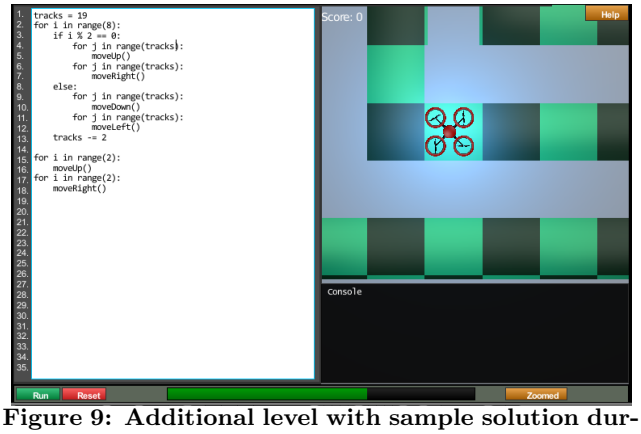

Figure 9: Additional level with sample solution during runtime

This level, illustrated by **Figure 9**, required more concepts in Computer Science in order to promote critical thinking so as to find an optimal solution for it. The results are given in **Table 4**.

Question	Likert Result (Avg.)
Would you have enjoyed playing a game like this as additional material for your first year course?	Strongly Agree
Do you think that the visual feedback is a good indicator to what your code is doing?	Strongly Agree
Do you feel that there should be additional gamification elements?	Agree
Is this game something that interests you?	Strongly Agree
Is the layout of the user interface elements appropriate/intuitive?	Agree
Do you feel that the game challenges your problem solving skills in terms of designing a complete piece of code?	Strongly Agree

Table 4: Likert scale results of the second user testing session

Every question that was asked from the previous user testing session garnered a better response after the system was improved. The participants still felt that they would like to have such a system to assist them in their course work, as well as that the system provides a good level of visual feedback to them.

Participants felt less inclined to want more gamification elements (Question 3). This is perhaps due to the additional challenge of the newer level that was introduced. The second round of user testing took significantly longer than the first round due to the participants working at finding an optimal solution for solving the additional level.

The user interface was significantly improved for this round of testing as well, and as a result participants felt more comfortable using it and enjoyed the experience more (Question 5).

While these results were more positive for the second round of testing, it is important to realise that they may not be statistically accurate. Since there were less participants for this round it is not safe to assume that the improvements were the only cause of the higher ratings.

Refer to appendix C.3.

5.4 Observations and Remarks

The participants of both user testing sessions were observed while they were using the system. It was interesting to note, during the initial user testing session, that participants expected a more complete system or game to begin with. Apart from that, some of the participants also expected more conventional video game controls, such as pressing a key to move the avatar. This could potentially indicate that these participants are not used to or may not have been exposed to more unique or non-generic video game paradigms.

6. CONCLUSIONS, DISCUSSION AND FUTURE WORK

The ultimate goal of this system was to test whether students might have an interest in taking advantage of an interactive platform, such as a video-game, in order to benefit their studying.

Learning to program can be a really difficult experience without the correct understanding of the subject material and when it comes to systematic and programmatic thinking, visualisations can do much to aid understanding. In this experiment a video-game like program was developed in order to determine its usefulness in assisting learning of programming concepts and programmatic thinking through visualisation and interaction. The system was designed using feedback from students that were relatively new to the world of Computer Science, as well as grounded theory from other studies.

The initial survey performed in this paper indicated that there is at least some interest from students in a game-like platform in which they can explore rich visualisation of what their code does. Specifically, students that were interviewed were interested in the idea of having such a system assist them in their normal course activities, perhaps integrated into the course directly.

The students that were interviewed also indicated that they like puzzle games. Puzzle games work well as a genre for encouraging critical thinking, feedback and understand-

ing. This allowed us to challenge the user and promote analytical thinking. When user testing was done, strongly positive feedback was received for the use of this kind of system in the classroom along with their traditional work and the students strongly felt that it challenged them mentally.

The students were of the opinion that there was a lack of gamification elements in the system, i.e. not enough attributes that make it feel like a traditional video game. This is something that could and should be expanded upon in order to garner higher levels of interest from more people. Along with this the user interface and, by extension, interaction experience is paramount to a system as such. The biggest problem that arose in the user testing was related to the experience people had with the interface.

In order to improve the system for future attempts, not only should additional gamification elements be introduced, but the implementation of the code interpretation should be improved substantially. It might perhaps be best to implement a custom parser for such a task. This would greatly improve security in the program.

The results of the experiments in this paper do well to answer positively for both of the research questions presented, but they are not conclusive. A larger quantitative study should be conducted.

7. ACKNOWLEDGMENTS

The author of this paper would like to thank Mark Grivainis, my partner on the implementation of the project and Gary Stewart for pointing us in the right direction whenever we needed it. He would also like to thank Dewald Schoeman and Vyacheslav Shevchenko of the University of the Western Cape for their input and discussion. Lastly, a special thank you to Nicole Thomas from the University of Cape Town for all her help, love and support.

8. REFERENCES

[1] R. Baecker. Sorting out sorting: A case study of software visualization for teaching computer science. *Software visualization: Programming as a multimedia experience*, 1:369–381, 1998.

[2] S. Esper, S. R. Foster, and W. G. Griswold. Codespells: embodying the metaphor of wizardry for programming. In *Proceedings of the 18th ACM conference on Innovation and technology in computer science education*, pages 249–254. ACM, 2013.

[3] M. Feldgen and O. Clúa. Games as a motivation for freshman students learn programming. In *Frontiers in Education, 2004. FIE 2004. 34th Annual*, pages S1H–11. IEEE, 2004.

[4] J. P. Gee. What video games have to teach us about learning and literacy. *Computers in Entertainment (CIE)*, 1(1):20–20, 2003.

[5] W. H.-Y. Huang and D. Soman. Gamification of education. Technical report, Research Report Series: Behavioural Economics in Action, 2013.

[6] C. D. Hundhausen, S. A. Douglas, and J. T. Stasko. A meta-study of algorithm visualization effectiveness. *Journal of Visual Languages & Computing*, 13(3):259–290, 2002.

[7] J. Kirriemuir and A. McFarlane. Literature review in games and learning. 2004.

[8] G. Kiryakova, N. Angelova, and L. Yordanova. Gamification in education. Proceedings of 9th International Balkan Education and Science Conference, 2014.

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

[10] J. Lee, K. Luchini, B. Michael, C. Norris, and E. Soloway. More than just fun and games: Assessing the value of educational video games in the classroom. In *CHI'04 extended abstracts on Human factors in computing systems*, pages 1375–1378. ACM, 2004.

[11] A. Morgan and S. Kennewell. The impact of prior technological experiences on children's ability to use play as a medium for developing capability with new ict tools. *ICT Research Bursaries, Becta*, 2005.

[12] M. Overmars. Teaching computer science through game design. *Computer*, 37(4):81–83, 2004.

[13] M. Papastergiou. Digital game-based learning in high school computer science education: Impact on educational effectiveness and student motivation. *Computers & Education*, 52(1):1–12, 2009.

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

[15] A. Radenski. Python first: A lab-based digital introduction to computer science. In *ACM SIGCSE Bulletin*, volume 38, pages 197–201. ACM, 2006.

[16] K. Rapeepisarn, K. W. Wong, C. C. Fung, and M. S. Khine. The relationship between game genres, learning techniques and learning styles in educational computer games. In *Technologies for E-Learning and Digital Entertainment*, pages 497–508. Springer, 2008.

[17] N. Tillmann, J. De Halleux, T. Xie, S. Gulwani, and J. Bishop. Teaching and learning programming and software engineering via interactive gaming. In *Software Engineering (ICSE), 2013 35th International Conference on*, pages 1117–1126. IEEE, 2013.

APPENDIX

A. SYSTEM AND EXPERIMENT DESIGN

A.1 Experiment Design

A.1.1 Informed Consent Form

The informed consent form was designed by Eugene and Mark and proved to ensure that all potential participants were willing and acknowledged that they would participate in the study.

Available on the PyRob website in http://pubs.cs.uct.ac.za under the Honours Project Archive section for 2015.

Alternatively:
File URL: http://people.cs.uct.ac.za/~dbseug001/res/img/appendicies/consent.pdf

Overview: http://people.cs.uct.ac.za/~dbseug001/res/pages/data.html#ICF

A.1.2 Faculty of Science Ethical Clearance

The University of Cape Town Faculty of Science provided

ethical clearance for the testing and surveys conducted on students from the University.

Available on the PyRob website in http://pubs.cs.uct.ac.za under the Honours Project Archive section for 2015.

Alternatively:
File URL: http://people.cs.uct.ac.za/~dbseug001/res/img/appendicies/FSREC.pdf

Overview: http://people.cs.uct.ac.za/~dbseug001/res/pages/data.html#DoCS

A.1.3 Department of Student Affairs Ethical Clearance

The University of Cape Town Department of Student Affairs provided additional clearance for the testing and surveys conducted on students from the University.

Available on the PyRob website in http://pubs.cs.uct.ac.za under the Honours Project Archive section for 2015.

Alternatively:
File URL: http://people.cs.uct.ac.za/~dbseug001/res/img/appendicies/DSA.pdf

Overview: http://people.cs.uct.ac.za/~dbseug001/res/pages/data.html#DoSA

B. IMPLEMENTATION

B.1 Design Documents

B.1.1 Paper Prototype

The paper prototype served as a critical part to the initial design of the system.

Available on the PyRob website in http://pubs.cs.uct.ac.za under the Honours Project Archive section for 2015.

Alternatively:
File URL: http://people.cs.uct.ac.za/~dbseug001/res/pages/data.html#PP

B.1.2 Game Design Document

The game design document is used to gather ideas and keep track of the implementation of the program with respects to the features in a non-technical way.

Available on the PyRob website in http://pubs.cs.uct.ac.za under the Honours Project Archive section for 2015.

Alternatively:
File URL: http://people.cs.uct.ac.za/~dbseug001/res/pages/data.html#GDD

B.1.3 Technical Design Document

The technical design document is used to detail the implementation of the various aspects of the program.

Available on the PyRob website in http://pubs.cs.uct.ac.za under the Honours Project Archive section for 2015.

Alternatively:
File URL: http://people.cs.uct.ac.za/~dbseug001/res/pages/data.html#TDD

C. EVALUATION AND RESULTS

C.1 Feasibility Survey

C.1.1 Feasibility Survey

The results from the feasibility survey were aggregated and compiled into a graph.

Available on the PyRob website in http://pubs.cs.uct.ac.za under the Honours Project Archive section for 2015.

Alternatively:
File URL: http://people.cs.uct.ac.za/~dbseug001/res/pages/data.html#chartContainer1

C.1.2 Video Game Fenre Interests

The results from the feasibility survey video game interest section were aggregated and compiled into a graph.

Available on the PyRob website in http://pubs.cs.uct.ac.za under the Honours Project Archive section for 2015.

Alternatively:
File URL: http://people.cs.uct.ac.za/~dbseug001/res/pages/data.html#chartContainer2

C.2 User Testing Stage 1

C.2.1 User Testing Feedback

The results from the user testing survey were aggegated and compiled into a graph.

Available on the PyRob website in http://pubs.cs.uct.ac.za under the Honours Project Archive section for 2015.

Alternatively:
File URL: http://people.cs.uct.ac.za/~dbseug001/res/pages/data.html#chartContainer3

C.2.2 User Testing Feedback Likert Scale

After the first user feedback results were collected they were put throught the likert scale.

Available on the PyRob website in http://pubs.cs.uct.ac.za under the Honours Project Archive section for 2015.

Alternatively:
File URL: http://people.cs.uct.ac.za/~dbseug001/res/pages/data.html#chartContainer4

C.3 User Testing Stage 2

C.3.1 User Testing Feedback

The results from the user testing survey were aggegated and compiled into a graph.

Available on the PyRob website in http://pubs.cs.uct.ac.za under the Honours Project Archive section for 2015.

Alternatively:
File URL: http://people.cs.uct.ac.za/~dbseug001/res/pages/data.html#chartContainer5

C.3.2 User Testing Feedback Likert Scale

After the second user feedback results were collected they were put throught the likert scale.

Available on the PyRob website in http://pubs.cs.uct.ac.za under the Honours Project Archive section for 2015.

Alternatively:
File URL: http://people.cs.uct.ac.za/~dbseug001/res/pages/data.html#chartContainer6