Project Proposal for PyTeacher

Eugene de Beste, University of Cape Town
Mark Grivainis, University of Cape Town

Supervisor:
Gary Stewart, University of Cape Town

Co-supervisor:
James Gain, University of Cape Town

1. PROJECT DESCRIPTION

1.1. Backgroud

Students today are less interested in studying computer science than in the past [Carter 2006; Lomerson and Pollacia 2006]. Current students in this field also tend to find the course boring and difficult. This has sparked educational institutions to introduce new ways of teaching, in attempt to make the course more interesting and viable to prospective students [Carver et al. 2007].

1.2. Problem Statement

Children and adults today use various types of electronic devices in their day to day activities. Technology is used as a tool to enhance quality of life and provide entertainment. It also has various other non-related applications. Video games, especially, are a part of the lives of modern children and teenagers alike.

Several attempts have been made to bring the study of computer science up to speed with the modern world. However, there is no existing tool which integrates directly with the progression and flow of introductory courses for this subject. This means that students are less engaged with the content than they potentially could be, leading to less students continuing with computer science into their second year.[Dann et al. 2012]

There are a few questions that need to be answered in order to understand how to approach this problem. Students vary in their background and what is fun for one might not be for the other. It is important to have a good understanding of the video game genres that students like the most. It is also important to know how they feel about the traditional way that the course they are undergoing is taught to them.

1.3. Proposed Solution

In attempt to remedy this problem, we propose a video game that assists lecturers in teaching students how to program. This would integrate into and complement the first year introductory university computer science course. The game would assist players with the introduction to various concepts and principles, following the progression of the course.

Players will be given an avatar and a set of 3D levels which they will need to navigate in order to complete challenges and objectives using computer science principles. Each level that a player completes will lead into the next, adding new principles while building upon previously learnt skills.

Students will be required to produce code in order for them to proceed in the game while being guided interactively. The Python3 programming language will be used as this is the language that they are instructed in. The code that the students produce will be interpreted by the game and it will give instant and direct visual feedback through avatar movements and the interaction of the avatar with the game world.

The avatar that a player is in control of will be customisable, this can lead to a heightened sense of enjoyment and increase in the time that users spend playing the game [Bailey et al. 2009].

Various features will be integrated for ease of use in the classroom. Key features from related works will be adapted to enhance the user experience. An analytics tool will be built into the game allowing lecturers to view the progress of student in the game down to an individual level. This enables the prospect of competitive play between students, as the time that their avatar takes to complete the different levels can be optionally posted to and viewed on a global leaderboard.

This video game is targeted at the desktop platform, including Microsoft Windows, Linux and Apple Mac OS X.

2. PROCEDURES AND METHODS

There are two parts to this project, the larger video game and the smaller web analytics application. Both of these are split into a front-end and back-end section, with each member of the team working on one of the sections. Both team members will plan of the overall design of the game and the website as well as build the poster for the project. The split is detailed below:

2.1. Front-end user interface - Mark Grivainis

Research Question.

Can an educational game using a code based learning environment lead to a state of game flow?

Video Game.

The art and sound assets that are to be used for the game must be designed and/or gathered. Many of these will be freely available online. Only a small portion of time will be dedicated to this as the graphical fidelity is not crucial to the success of the project.

A 3-panel user interface will be created. This will consist of a graphical window that shows game environment, a script editor in which users will give commands to their avatar in order to navigate the world and a console will also be provided to help users with debugging their code and displaying errors.

The script editor will be similar to the Python3 IDLE and will allow basic syntax highlighting as well as a popup dialog that will link to the python documentation. If a user has an error in their code, the line will be highlighted and a popup window will be displayed to aid the player in debugging their code.

Levels will be laid out using game objects that allow for player interaction. Players will need to write code that the avatar will use to interact with these game objects in order to complete the level. Blender will be used as the interface for level design, this will allow for levels to be designed quickly and to make it easy to change levels in the

future.

Web Application.
The web application will allow the lecturer to keep up to date with the progress of his/her class. Using graphs he will be able to analyze sections in which students are struggling and sections that they are excelling in. This will allow them to make adjustments to their course in future if patterns emerge where the entire class struggles with a particular concept. Information will be viewable down to an individual level, displaying hours spent in total and on each specific level.

Testing.
Heuristic evaluation will identify usability problems that are present in the user interface. User testing will allow for observations of first year students interacting with the system and then asking them to answer a questionnaire about their experience using the application.

2.2. Back-end logic - Eugene de Beste

Research Question.
Does translating the course content into a video game challenges allow students to feel a greater sense of enjoyment in learning programming?

Video Game.
The video game backend will consist of the game logic, the scaffolding and progression mechanics, the integration of the course content with the instructional design and the handling of source code as input. As such, a number of elements need to be worked on.

— Game Logic:
The most common and necessary process of designing a video game includes creating logic for the various objects in the world. This task includes the creation of scripts and classes that define rules for the behaviour of the objects in the world related to themselves and others. It would also involve designing logic to allow easy level creation.

— Parser/Interpreter:
The player will be using source code to interact with the world and therefore a parser or translator needs to be built in order to understand the commands. To do this various logic needs to be designed to read in the user input, break it down into an abstract syntax tree and sift through things that may or may not be allowed for specific levels and challenges in the game. On top of this the commands then need to be translated into action of the on-screen avatar and objects in the game world.

User errors also need to be caught and valuable information needs to be displayed to the user through visual cues. This also needs to be done in order to prevent a user from breaking the game with incorrect code or syntax.

— Progression Mechanics/Scaffolding:
The design and implementation of the scaffolding systems and progression mechanics is key to this endeavour. This involves creating the challenges for the users to play through and how the course content is used in these regards. This will be done using the flow of the traditional course to allow the game and the course to run in parallel. The content that is included in the game will be based on a smaller subset of the introductory course, covering things such as data types, arithmetic operations, string manipulation, input-output and conditionals.

Web Application.
The web application back-end requires two parts:

— The video game will contain code that reads and collects statistics of the players and sends this data to a database that will be located on a web server.
— A database will be set up to house the data collected from students and can be used to populate a web front-end.

Testing.
Unit testing will be done for the code base, while the method of testing the effect of the game will involve staff members playing through levels in order to determine whether the challenges are sufficient and the content is being brought across correctly. Students will also be given play testing sessions in order to determine whether they experience a good state of flow when solving the challenges.

3. RELATED WORK

Various work has been done in the realm of gamifying education, specifically in computer science. Most of the attempts for interactive education have lead to visual learning environments, with interfaces that allow drag and drop or code based interactions. There are few that attempt to be a video game or a simulation where the player is in direct control of some avatar that they use to interact with a virtual world directly. Some related works follow:

Turtle Graphics.
Logo is an old programming language that, with the help of Turtle, has been used as a tool to teach more mathematical thinking rather than computer science, but its application cannot be ignored. It involves the use of a turtle pen marker object that can be instructed to draw various things through commands. This allows visualisation of what a user is doing, albeit quite primitively [Agalianos et al. 2001].

Turtle has more recently been adapted into libraries for use with the Python programming language, providing more opportunity for students to learn programmatic thinking and principles through the visualisation of their actions. It remains limited in the scope of what it can teach.

Jeroo.
This is an education platform focused more on the teaching of object oriented concepts to people at the college and university level. Jeroos main interface is comprised as a text editor with an accompanying visual map. The user needs to configure the map as well as control multiple individual objects or Jeroos (a metaphor for a rare animal akin to the Australian wallabies) to interact with the map [Sanders and Dorn 2003].

It has good principles such as source code syntax highlighing and animated execution of code which is written by the user. But with its simple user interface and graphics it provides an unintuitive and non-engaging experience.

Alice2.
Alice is aimed at an older audience (around ages 12-19). The use of Alice in first year computer science has had great success with students who had no previous experience in programming. Students with no programming experience who were taught using Alice had grades comparable to students who had

programming experience and a large percentage of these students continued with second year Computer Science [Dann et al. 2012].

Alice is based around Java, and provides a platform for students to progress into using regular Java. The commands are represented as blocks which take a number of values to perform a function. This system allows students to avoid many syntax errors and create a learning environment that is more about learning concepts than syntax.

Alice includes an interpreter and visual object based language that can be used to teach students arrays, loops, functions and variable passing. The basics of parallelism are also taught by allowing students to choose whether a block should be run sequentially or concurrently.

When a user runs an Alice program, all assets that will be used must be loaded at the start. This results in long load times for complicated scenes. Its is not possible to make dynamic objects during runtime [Villaverde et al. 2009].

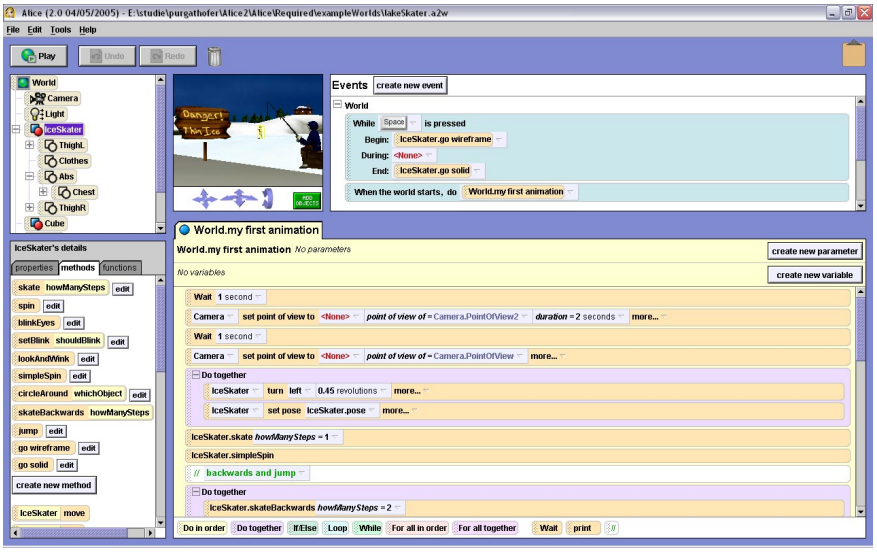

Fig. 1: Alice User Interface

CodeCombat.

CodeCombat aims to teach people how to code at varying levels of skill. Using a mix of the real time strategy and role playing game genres, it provides challenges for players to solve and has various progression mechanics, such as unlocking of characters as the player goes through different levels. It is a 2D HTML5 based game and that has done a good job of proving that serious games can be used as a way for people to learn. However, it does suffer from various user experience issues such as no customisable user avatars and no way to see past code [Chen 2015].

CodeCombat has a leaderboard that shows how long it took the avatar to

navigate a given level. A drawback to this is that CodeCombat allows players to equip items giving them access to more advanced methods. Players new to programming might get frustrated trying to understand why their code is less efficient than the top players on the leaderboard as they cannot use the advanced methods yet.

Fig. 2: CodeCombat User Interface

Below is a comparison of the different aspects that each of these programs adopt in order to portray information to the users:

Table I: Instructional Design Breakdown for Review

Program Name	Instructional Design Approach	User Interface Approach
Turtle (Logo/Python)	No structured instructional design. The turtle marker moved by a user to create lines. How to do so is conveyed through a teacher or by consulting documentation.	Only offers a graphical window to view the result of your code.
Jeroo	Provides a metaphor to allow a user to understand what they are trying to achieve or work towards. It is designed to be used along with an educator that instructs them on how to use it.	Multiple panels that are visible to the user at all times. These consist of a source code editor, graphical window to view the world, message area, cursor location panel (gives the user coordinates of the cursor in the world) and a Jeroo status area to display stats about the Jeroo.
Alice2	No structured instructional design. Additional content is required in order to learn how to use Alice. Once done a user can create various animated scenes through its interface.	There are five panels that are visible at all times to the user. A scene graph that shows how object are parented in the world, a graphical window that shows the 3D scene, a panel that holds all the possible code blocks that users can use and a panel in which these blocks are combined to create a program.
CodeCombat	Makes heavy use of scaffolding. It starts users at a very basic level, interactively introducing them to the concepts and principles of the game. It then build on these and expects users to solve more complicated problems as it progresses. It is designed to work independently of a course.	There are two main panels, a graphical window that shows the level and a code editor. There is a minor panel that features a time slider which allows users to run through their how their program makes the avatar interact with the world.

4. ANTICIPATED OUTCOMES

4.1. System

The complete system, including the back-end and the front-end of the video game will be bundled into a single software package. The web application front-end will be separately hosted at an external site and include a database. The back-end of the web

application will be integrated into the video game back-end and communicate with the aforementioned database. It will be made so that the game does not require the web application services to be functional in order to operate correctly.

The main outcome of this project is a fully playable video game with a 3D world for a player to explore and interact with. It would be packaged for easy deployment on a Windows machine, as well as Linux and Mac OS X.

4.2. Key Success Factors

The key success factors for this project will be determined mainly through user testing. There are four factors that we aim to achieve:

— A system that is easy to learn and use. Not much effort should be required by the player to learn how to effectively interact with the video game.
— Expandability. The system needs to be flexible enough to allow minimal effort when adding new levels to the game.
— Whether the teaching staff or lecturers believe that the system is capable and acceptable for use in the classroom in assisting their teaching.
— Achieving a good state of flow when players are using the system.

4.3. Expected impact of project

Provided that the system meets the success factors above and is integrated into the first year computer science course at UCT we expect five things:

— Students will be able to easily grasp beginner programming concepts and have a place to practice them with direct visual feedback.
— Students will be far more engaged in the content than traditional means and visual programming languages.
— Allows for deeply learning the syntax of Python as well as exploring the cause of errors and mistakes.
— Increases the potential motivation for students through competitive elements like scoreboards for challenges and all of this could lead to the students developing a deeper understanding of the basic programming concepts that they are to be taught in the course.
— Increase in number of students passing first year and continuing into their second year.

5. ETHICAL, PROFESSIONAL AND LEGAL ISSUES

5.1. Issues

Acquisition of the ethical clearance/permission to conduct a survey for student preferences may not happen. This would make it very difficult to design a solution that would be applicable to a larger, more diverse audience. If this is the case we would still be able to and have to resort to only focusing on the literature review to get the relevant information needed to design the game.

5.2. Legal

We would like to place the project, including all assets, under the Creative Commons Attribution Non-commercial (by-nc) license. This, in summary, would allow a person to copy and redistribute the material in any medium or format as well as adapt or change the software in any way as long as the original authors are given credit and all changes are stated. The software may also not be used for commercial purposes.

Some assets that are available online have licenses attached to them dictating that they may or may not be used in conjunction with the license we want to publish the game under. Due to this, careful attention will need to be given to the assets that are chosen for the game.

6. PROJECT PLAN

6.1. Deliverables

This project will include multiple deliverables:

Table II: Table of Deliverables

Title	Due Date
Final Project Proposal	11/06/2015
Website Presence Report	12/06/2015
Initial Feasibility Demonstration	20/07/2015
Background and Theory Section of Final Report	24/07/2015
Design Section (Game Design Documents)	21/08/2015
First Implementation	11/09/2015
Writeup for First Implementation	11/09/2015
Final Prototype	21/09/2015
Writeup for Final prototype	21/09/2015
Final Implementation	25/09/2015
Outline of Complete Report	02/10/2015
Final Complete Report Draft	16/10/2015
Final Project Report	26/10/2015
Project Poster	02/11/2015
Project Website	09/11/2015
Reflection Paper	13/11/2015

6.2. Milestones

Table III: Table of Milestones

Milestone	Date of Completion
Project Proposal Finalised	11/06/2015
Website Design Finalised	12/06/2015
Initial Game Design Review	30/06/2015
First Prototype Feasibility Assessment	20/07/2015
Second Prototype Pre-exam Playtest	14/08/2015
Second Prototype Second Playtest	07/09/2015
First Experiment Evaluation	12/09/2015
Final Prototype Submission	15/09/2015
Final project Implementation Complete	26/09/2015
Poster Design Finalised	10/10/2015
Poster Made	25/10/2015
Website Made	01/11/2015
Reflection Paper Complete	05/11/2015
Final Presentation Complete	10/11/2015

6.3. Resources Required

Main resources that are required are listed below:

Since a requirement for the project is to use the Python programming language, a game engine with strong and well-developed Python wrappers and libraries is needed We have identified the Blender game engine as having this capability. Having this engine will provide a common base for both members to work in parallel.

3D designing and animation software, as well as image manipulation software is required in order to design and create all the visual assets for the game. A computing device with sufficient power to do content creation with these software packages is also needed.

Access to the course content for the first year students at the University of Cape Town is required in order to design and build a suitable game story and challenges that use the concepts and build on them.

People who are willing to sit with the feature-complete game, play through it and evaluate the various aspects of it need to be identified and approached. They

would provide valuable feedback on the success of the project and whether there are critical issues remaining to be fixed.

6.4. Risks and Mitigation

Table IV: Risks and Mitigation Strategies

Risk	Probability	Impact	Mitigation
Survey results are inconclusive.	5	4	Attempt to do additional surveys, as well as get feedback from other sources such as the lecturers of the course.
Performance issues on devices with low-end hardware.	3	7	Front and back-end sections both designed and implemented with performance in mind. Gather assets that do not require high performance machines and develop code that is scalable to a variety of systems, providing settings to allow users to improve performance on their machines.
Game engine is restrictive in implementation of key features.	4	8	Extensive research needs to be done on it. Alternatives are to be researched as well, in the case that the primary one does not comply with what is required from it.
Time taken to learn game engine taking longer than intended.	6	4	External resources such as online support and forums will need to be heavily utilised to keep this possibility low.
Scope creep as additional features are added.	7	6	Ensure that both team members stick to the design documents strictly through regular meetings and progress reports.
Group member leaves.	1	9	A much smaller version of the member who leaves task will need to be done by the remaining member. This will allow the remaining member to demonstrate their section of the project with a more proof-of-concept approach.
Source code becomes unmanageable.	5	5	Make extensive use of version control software in order to create maintainable code.

6.5. Additional Considerations

The window of time available for the development of the project is fairly short and as such more ambitious aspects of the project need to be put on a potential list, which may not make it into the final build of the game. The features that may not make it into the project follow:

— Multiplatform support: Making use of the limited time available to complete and polish a game for one particular common platform would be the best way of approaching this.
— Level/task/quest designer: Providing tools for people to make user-created content for the platform and ensuring that it cannot be exploited and is bug free would be a waste of time if it can rather be spent doing exploit and bug checking in the main product.
— Number and quality of assets: The process will start with a small amount of assets of reasonably high quality. Time constraints could cause less assets and/or lower quality assets than planned to be developed.
— Networked play: The competitive multiplayer aspect might not be possible due to time constraints. It is not a critical feature, however it is a target we hope to be able to reach.

Along with this, one of the big limitations introduced is keeping the game to one programming language, rather than giving a player the choice to use another.

6.6. Timeline

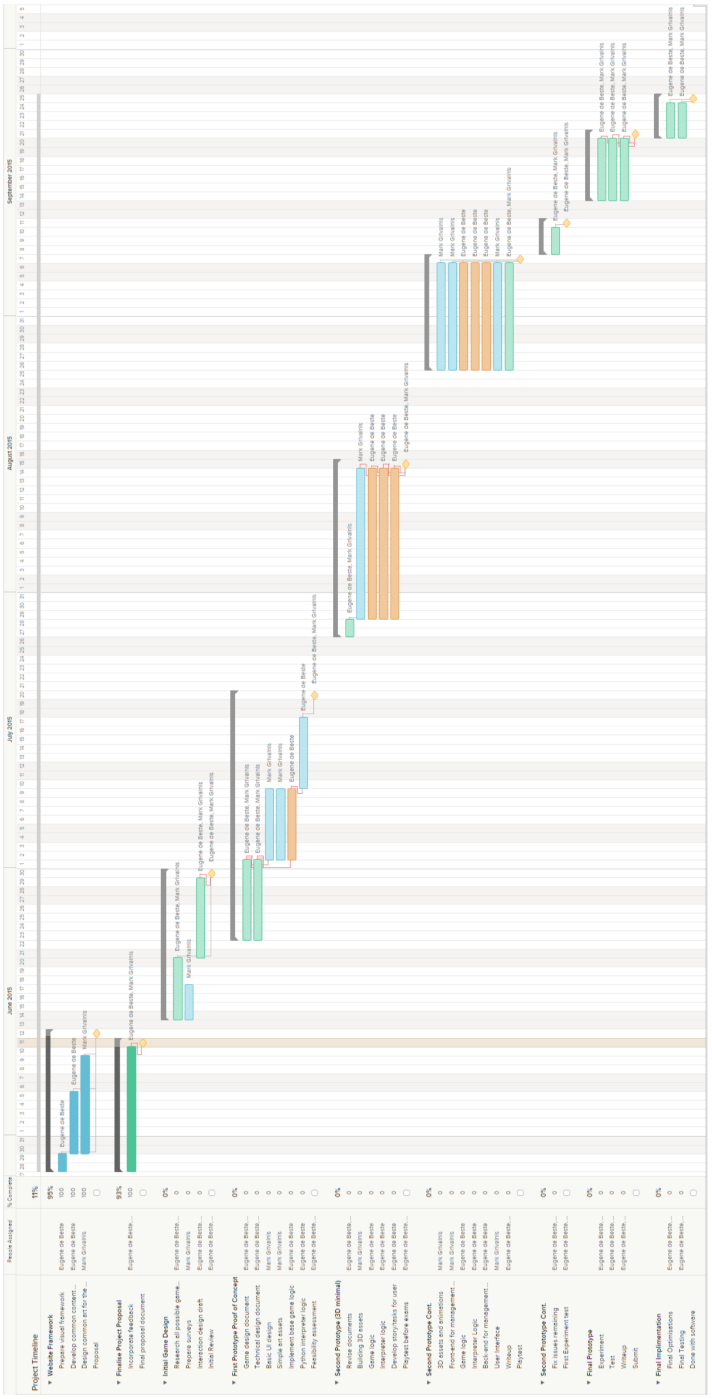

Fig. 3: Work Breakdown Gantt Chart

REFERENCES

Angelos Agalianos, Richard Noss, and Geoff Whitty. 2001. Logo in mainstream schools: the struggle over the soul of an educational innovation. *British Journal of Sociology of Education* 22, 4 (2001), 479–500.

Rachel Bailey, Kevin Wise, and Paul Bolls. 2009. How avatar customizability affects children's arousal and subjective presence during junk food–sponsored online video games. *CyberPsychology & Behavior* 12, 3 (2009), 277–283.

Lori Carter. 2006. Why students with an apparent aptitude for computer science don't choose to major in computer science. In *ACM SIGCSE Bulletin*, Vol. 38. ACM, 27–31.

Jeffrey C Carver, Lisa Henderson, Lulu He, Julia Hodges, and Donna Reese. 2007. Increased retention of early computer science and software engineering students using pair programming. In *Software Engineering Education & Training, 2007. CSEET'07. 20th Conference on*. IEEE, 115–122.

Yingying Chen. 2015. *A serious game Defying Disaster: Earthquake*. Ph.D. Dissertation. WORCESTER POLYTECHNIC INSTITUTE.

Wanda Dann, Dennis Cosgrove, Don Slater, Dave Culyba, and Steve Cooper. 2012. Mediated transfer: Alice 3 to java. In *Proceedings of the 43rd ACM technical symposium on Computer Science Education*. ACM, 141–146.

William L Lomerson and Lissa Pollacia. 2006. Declining CIS enrollment: An examination of pre-college factors. *Director* (2006), 07.

Dean Sanders and Brian Dorn. 2003. Jeroo: a tool for introducing object-oriented programming. In *ACM SIGCSE Bulletin*, Vol. 35. ACM, 201–204.

Karen Villaverde, Clint Jeffery, and Inna Pivkina. 2009. Cheshire: Towards an Alice based game development tool. In *International Conference on Computer Games, Multimedia & Allied Technology*. 321–328.