An Evaluation of Gamification to Assess Students' Learning on Their Understanding of First Year Computer Science Programming Module

Daniel Gebremichael

A dissertation submitted in partial fulfilment of the requirements of

Dublin Institute of Technology for the degree of M.Sc. in Computing

(Advanced Software Development)

January 2016

DECLARATION

I certify that this thesis which I now submit for examination for the award of MSc in Computing (Advanced Software Development), is entirely my own work and has not been taken from the word of others, save and to the extent that such work has been cited and acknowledged within the text of my work.

This dissertation was prepared according to the regulations for postgraduate study of the Dublin Institute of Technology and has not been submitted in whole or part for an award in any other Institute or University.

The work reported on in this dissertation conforms to the principles and requirements of the Institute's guidelines for ethics in research.

Signed: ___________________________
Daniel Gebremichael

Date: __*03 January 2016*______________

ABSTRACT

This research examines the use of gamification to develop an assessment tool, to assess students' learning of a first year computer science module. The students' undertaking of the first semester *Programming and Algorithms* module in 2015 were assessed on their knowledge of the programming language Python.

The incorporation of gamification when assessing students can have various potential benefits. The research aims to identify these benefits and issues. Assessments and games have almost opposite effects on opinions on people, as games are usually expected to have an entertainment value but this is not the case for assessments. The research examines if the game elements in the assessment tool causes the students to see this tool differently.

A variety of experiments were carried out and the organisation of these experiments were vital to the success of the project. The first assessment was to test students through a written test. The findings of this experiment were used in order to develop a game to assess the students. Interviews with computer programming lecturers were conducted before the game was developed (to derive requirements).

A game was developed and used to assess the students learning. The game is accessible via a website, thus platform agnostic, enabling feasibility in partaking of this experiment. The opinions of the students in relation to the game were also gathered immediately after the students had completed the game.

The second experiment stage then took place to assess the students with a written test that had the same questions as the written test before the students played the game. Interviews with computer programming lecturers were conducted after all other experiments were completed (to evaluate the outcomes).

The game was able to assess students' learning and obtain vast amount of information. All students indicated they liked the game and enjoyed it which also means that the students enjoyed the assessment process. Several beneficial elements for incorporating gamification were identified. Such as improvement of students' knowledge which was determined through the comparison of the written test performance.

ACKNOWLEDGEMENTS

I would like to thank my dissertation supervisor Damian Gordon, who has been a source of inspiration, knowledge and guidance throughout the project.

I would like to thank all the students in the in Dublin Institute of Technology (DIT) in the first year of the course DT255 BSc Information Technology and Information Systems for participating in the research, the encouragement and friendliness. Further thanks to Damian Gordon, for providing the access to students who participated in the research.

Finally, I would like to thank, Ciaran O'Leary and Michael Collins for taking time out of their very busy schedule and participating in informative interviews.

TABLE OF CONTENTS

TABLE OF FIGURES

TABLE OF TABLES

1 INTRODUCTION

1.1 *Project Background*

First year computer science students in third-level education come to their institute with some-to-none programming-related knowledge. It can be an issue for some students that they are intimidated by other students who appear to have more knowledge than them about a given topic. The rate of which students understand the material may vary dramatically and may leave some students lagging behind others. This could lead to classes progressing too quickly for some and too slow for others. If a student is struggling with some of the material and fails to ask for additional assistance due to the fear of standing out from the class, this could lead to the student having a negative experience or opinion of the course. On the other hand, it could also cause the class to progress to slowly and may be difficult for the lecturer to cover all their materials.

An issue that third-level institutes have to face is that students could be memorising material they covered rather than being able to understand and apply, that knowledge. Students who have done a Leaving Certificate in Ireland specifically could be arriving at third-level education with their formula of remembering material and recalling it. This may have worked for them in the past but in a computer science course, students are expected to learn the skills rather than remembering materials only.

Bloom's Taxonomy cognitive domain (1956) involves "*knowledge and the development of intellectual skills*". This is an attempt to classify forms and levels of learning. It arranged levels in a hierarchy where one level must be mastered before reaching the next level. The levels were ordered: Knowledge, Comprehension, Application, Analysis, Synthesis, and Evaluation.

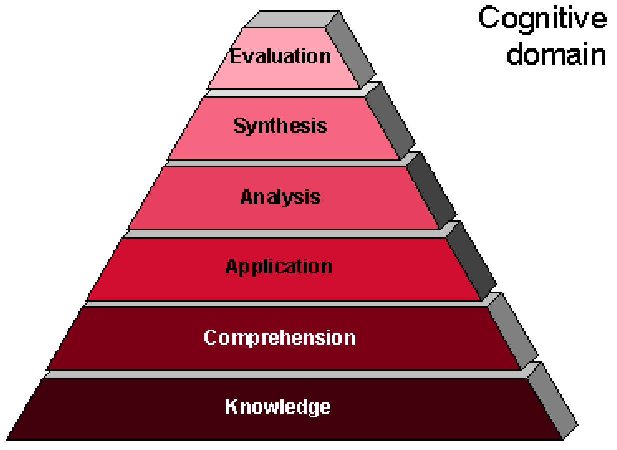


FIGURE 1.1: BLOOM'S TAXONOMY OF THE COGNITIVE DOMAIN

The knowledge level requires the student to be able to recognise information and be capable of remembering it. This is the level the students may have overly relied on before their journey to third-level education. The comprehension level relates to the student being capable of understanding and interpreting information obtained from prior learning. This is the level students require during their first year of computer science in specific modules that contain programming.

The application level involves the student being able to complete or solve a task with little direction. The application level is demonstrated by the students as they take part in assignments and laboratory work. The analysis level requires a student to be capable of analysing, comparing and contrasting information. The students' ability to compare and contrast information can be an indication of how well they understand their knowledge of the material they are being taught.

The synthesis level relates to the student being capable of creating something new to them, such as designing, hypothesising or developing. This level may be observed by first year computer science students but it is not expected of them. The evaluation level is the final stage of the Bloom's Taxonomy cognitive domain. It involves in the student being capable of critiquing and justifying. First year computer science students are not expected to achieve this level. Since then, (Krathwohl, 2002) made a significate modification and revised the taxonomy of the cognitive domain. Being able to create new knowledge within the domain is the top category in the revised taxonomy of the cognitive domain.

The SOLO (structure of observed learning outcomes) taxonomy developed by Biggs and Collis (2014) shows similarity to that of Bloom's taxonomy, as it describes levels of complexity in students' understanding of a subject through five stages. Classroom assessment is an approach designed to assist lecturers in identifying what and how well students are learning. There are a variety of techniques one may use to assess students and no one method can be identified as the ideal technique. Instructors can focus more effectively to meet the learning needs of the class by determining what students have learned and what is unclear to them (Angelo & Cross, 1993).

It is difficult for lecturers to identify whether all of the students are absorbing information from the lecturers and if the students understand it clearly. Students studying computer science modules for the first time may not necessary know how to share their lack of understanding of certain aspects well. It can be difficult to ask for assistance when it is difficult for one to explain the problem in the first place. This could have a dramatically negative impact on the students' attitude towards the module or course. This could lead to a lack of motivation and cause dissatisfaction.

Students could do well at classwork and homework but this does not always mean that they understand their work or the concept. Lecturers struggle with identifying this types of issues, if it appears to them that the students are doing well with their classwork and homework. This is an issue for computer science modules as students may obtain information online without understanding it well. Students could end up plagiarizing solutions from online resources and not understand the solutions clearly. This could cause a lack of good fundamental knowledge that could make students future learning more difficult.

Gamification is a term used to describe the use of game elements in non-gaming systems to improve user engagement and experience (Deterding *et al.*, 2011). In third-level education, gamification could possibly be used to see the progress of students throughout their course and how well they understood certain subjects. It may be capable of identifying which students are struggling at which areas of study. The aim of a gamification approach is that it should not make the students feel like they are doing an exam or test, but rather testing themselves for their own benefit. A way for the students to understand their homework and prepare for any tests or exams. The ability for the student to observe their progression could make the student more encouraged for future progress.

1.2 Project Description

This research project is designed to assess students' learning of their understanding of a first year computer science module. The first year students undertaking of the first semester *Programming and Algorithms* module in 2015 were assessed. A variety of experiments were carried out and the organisation of these experiments were vital to the success of the project. The first assessment was to test students through a written test. The findings of this experiment were used in order to develop a game to assess the students.

A game was developed and used to assess the students learning. The game was placed on a website for students to access and participate in playing the game. The opinions of the students in relation to the game were also gathered immediately after the students had completed the game. Interviews with computer programming lecturers were conducted before the game was developed (to derive requirements). The second experiment stage then took place to assess the students with a written test that had the same questions as the written test before the students played the game. Interviews with computer programming lecturers were conducted after all other experiments were completed (to evaluate the outcomes). The experiment is divided into two main stages, a pre-game and post-game stage.

Pre-Game Stage

- Assess students through a written test
- Interview lecturers

Post-Game Stage

- Assess students through the game
- Conduct a survey about the game
- Assess students through another written test
- Interview findings with lecturers

The pre-game experiments were vital to the development of the game as it helped to identify what the difficulty level of the game should be. This experiments are located in the design chapter rather than experimental as it was carried out before the development of the game and influenced the design of the game. All the experiments conducted on students were carried out under a controlled environment, where the research writer supervised the experiment in person from the beginning till the end. All interviews were conducted in locations were the interviewee felt comfortable and had no distractions. Interviews were all recorded as evidence.

1.2.1 Pre-Game

The assessment of students using a written test contributed to various findings and the direction of the remainder project work. It identified the current level of students' knowledge in their module, the mistakes students were making, students' assessment of the difficult of all questions and students' own assessment of their knowledge of different programming concepts. These information collected contributed to identifying a suitable level of difficulty for the game. Interviews with lecturers were designed to obtain lecturers' experience and knowledge of assessing students in a first year computer science programming module.

1.2.2 Post-Game

The game developed was used to assess students on the materials the students struggled with from the pre-game written test. It assessed each students performance by recording the time and number of attempts it took for students to complete each level. Students' opinions on the game, its features and other uses for it were collected.

A second written test was used to assess students' knowledge in their module, the mistakes students were making, students' assessment of the difficulty of each question and students' own assessment of their knowledge of different programming concepts. The aim of the second written test was also to identify if any changes had occurred in students' performances and opinions. Interviews with the same lecturers in the pre-game interviews were conducted to obtain lecturers' perspectives in some of the unexpected results obtained.

1.3 *Research Questions*

All research questions have a set of hypotheses, the expectations is for the questions to be answered by the verification of the one of the set of hypotheses described.

Research Question 1

Using a written test that contained a method for students to grade their knowledge of different programming concepts, would the students' assessment of their knowledge differ significantly after they have participated in playing the game developed to assess them?

First hypothesis 1 – The game developed to assess the students' knowledge would not change the students' opinion on their level of knowledge of different programming concepts.

Second hypothesis 1 – The students would assess their knowledge of different programming concepts differently after they have participated in playing the game developed to assess their knowledge.

Research Question 2

Using a written test before and after the students have participated in playing the game developed to assess their knowledge, would the average score of the students' written test differ significantly?

First hypothesis 2 - Using a written test before and after the students have participated in playing the game developed to assess their knowledge, the average score of the students' written test would have no to little variation.

Second hypothesis 2 - Using a written test before and after the students have participated in playing the game developed to assess their knowledge, the average score of the students' written test would change significantly.

Research Question 3

Using a game developed to assess students' learning, would students prefer to be assess through games rather than programming in class or written test?

First hypothesis 3 - Using a game developed to assess students' learning, students would not prefer to be assessed through games.

Second hypothesis 3 - Using a game developed to assess students' learning, they see some benefits of being assessed through games.

Research Question 4

Using a game developed to assess students' learning, would students like the game even though it is used to assess their knowledge?

First hypothesis 4 - Using a game developed to assess students' learning, students would not show signs of liking the game.

Second hypothesis 4 - Using a game developed to assess students' learning, students would show some signs of liking the game even if it was assessing their learnings.

Research Question 5

Using a written test, would students find the questions in the written test to be less difficult after playing a game developed to assess the students learning?

First hypothesis 5 - Using a written test, students will not find the questions in the written test to be less difficult after playing a game developed to assess the students learning.

Second hypothesis 5 - Using a written test, students will find some or all questions in the written test to be less difficult after playing a game developed to assess the students learning.

Research Question 6

Using interviews and surveys, can a game developed to assess students also be used for revision purposes?

First hypothesis 6 - Using interviews and surveys, the game developed to assess students will be identified to not be useful as a revision tool.

Second hypothesis 6 - Using interviews and surveys, the game developed to assess students will be identified to have some useful beneficiaries as a revision tool.

1.4 Research Objectives

The research objectives of this project is as follows;

- Examine educational games in general and in the field of computer science with focus on gamification.
- Investigate current views in game design.
- Review appropriate research related to tool for assessing student learning.
- Identify the tools to be used for developing the game.
- Design experiment to assess student learning before, during and after gamification.
- Record the information gather by all the tools used.
- Analyse the information gathered.
- Make recommendations for future research in this area.

1.5 Research Methods

1.5.1 First Stage

This research obtains its initial stage of information from literatures related this research. Relevant studies were searched for and collected using the following electronic databases and scholarly resources: DIT Electronic Library, Institute of Electrical and Electronics Engineers (IEEE) IEEEXplore, The ACM Digital Library, ScienceDirect, Academic Search Complete and ISI Web of Knowledge. The search engine Google Scholar was used to find the electronic databases and scholarly resources that contain relative studies.

Boolean searches allowed for more refined searches to be carried out, so that it has all of the search terms, some of the search terms and even excluding certain search terms. Some of these databases provide a wide range of fields to search by, as well as allowing use of Boolean searches in these fields, thus allowing greater precision, when searching by searching title, keywords, authors and so much more.

The main key search terms and texts searched are as follows;

1. Concept of Educational Games: “educational games” OR “serious games” OR “learning games” OR “teaching games”

2. Concept of first year student: “first year” OR “freshman” OR “fresher” OR “new student” OR “new undergraduate”

3. Concept of third level education: “third level education” OR “higher level education”

4. Concept of assessing: “assessing” OR “testing” OR “evaluating” OR “examining”

5. Concept of computer science/ programming: “computer science” OR “computing” OR “programming” OR “coding” OR “python”

6. Concept of gamification: “gamification” OR “game based” OR “game mechanic” OR “game”

7. Concept of Educational theory: “educational theory” OR “learning theory”

1.5.2 Second Stage

Three lecturers who in the past or presently are teaching programming to first year students provided the second stage of information. This was done qualitatively. All three lecturers participated in an audio recorded semi structured interviews. Each lecturer did two interviews, one before the development of the game and another at the end of the project. This type of interview provided greater flexibility in the questions asked depending on the answers given. This also led to a more detailed explanation of some answers due to cross checking.

1.5.3 Third Stage

This stage involved the collection of data from the students who participated in the experiments. This stage is both qualitative and quantitative. It is important to note that experiments before the development of the game are located in the design chapter, as it was vital for the design of the game and remaining experiments found in the experimental and evaluation chapter. This stage provided both qualitative and quantitative information. Two written tests provided both a qualitative and quantitative information. The quantitative information gathered from the written tests were the results of the students, their assessment of the difficulty of each question, their own assessment of their knowledge in different programming concepts. The qualitative information obtained from the written tests was the feedback of the students about the written test and its contents. A survey was used for gathering qualitative information about the game developed and played by the student.

1.6 Scope and Limitations

1.6.1 Research Scope

Work on this project commenced with the meeting of the supervisor of the research writer on the 25th of September 2015 up to the electronic submission date of this paper on the 3th of January 2016. The research was conducted at Dublin Institute of Technology (DIT), Kevin Street, Dublin 2.

The student participants were all first year DIT computer science students with a total number of 22, studying the "Programming and Algorithms" module. This was a first year module for computer science students studying under the DT DT255 programme. The lecturers who participated in the interviews were 3 lecturers employed by DIT.

1.6.2 Research Limitation

- This research was limited to only 22 students.
- The interviews conducted depended on the interviewees' choice of date and duration of interview.
- The interaction of the students with the game they played were not video recorded, to prevent students from feeling uncomfortable of being recorded.

1.6.3 Research Assumptions

- The students will abide to the rule of no cheating.
- The resources required for the competition of the project will be ensured of availability by the research writer.
- The lack of students in attendance will require the rearrangement of date the experiments will run, it may cause a delay in the project schedule.

1.6.4 Research Constraints

This research had a timeframe of approximately 3 months.

1.7 Organisation of Dissertation

- Chapter two reviews literatures in serious games, gamification, game design, digital badges, educational theories and teaching programming.
- Chapters three and four contain the design of the game. Chapter three focuses on the existing games that have influenced the design of them game, incorporation of educational theory content covered in chapter 2, Methodologies, puzzle game designs, the concepts of the game and paper prototypes. Chapter four focuses on the software to be used for the development of the game and requirements of the game.
- Chapter five details the development of the game to be used for assessing student learning. It provides a very descriptive information on the procedures taken to create the game.
- Chapter six details the experiments conducted using the game created in the previous chapter and other methods with the inclusion of evaluation. This chapter details how the experiments were prepared and conducted. It provides the results of the experiment. This chapter also evaluates the results obtained throughout the project.
- Chapter seven provides a conclusion to the research and all of its contribution in its respected field of knowledge. Any future work and suggestions are identified here.

2 LITERATURE REVIEW

2.1 Introduction

This chapter reviews educational games by examining both serious games and gamification. Many aspects of game design are reviewed. There is also an examination of well-known and researched learning theories. These theories are Behaviourism, Cognitivism and Constructive. The acquirement of knowledge by change in behaviour is examined in Behaviourism. Cognitivism explores how knowledge can be obtained with self-cognition and interaction. Constructivism deals with one’s mental activity being a major factor in obtaining new knowledge. Instructional design and its systematic development are reviewed in this chapter as well.

2.2 Educational Games

Games that are designed to assist the player to learn about a specific subject, reinforce development or learn new skills are referred to educational games. Both serious games and gamification fall in the category of educational games. Serious games are games that are designed with the entertainment aspect of the game being not the primary focus. Gamification involves the incorporation of elements of game playing to encourage engagement.

2.2.1 Serious Games

Serious games are games that are used to simulate real world events in order for the player to solve problems. How serious games have more purpose than just entertainment was discussed by (Susi *et al.*, 2007). They explained that serious games are used to give learners experience that are impossible to do so in the real word for reasons such as, cost, time and safety. It was indicated by them that there is a challenge for identifying research evidence for the benefits of educational games, due to the several verity of disciplines related to games.

According to (Johnson, Vilhjálmsson, & Marsella, 2005), game designers are increasingly employing techniques that promote long term engagement in serious games. They explored if Artificial Intelligence in Education (AIED) technologies complemented and enhanced serious game design technologies. Since the game design assumes some responsibilities for promoting learning, they indicated that the serious game context makes it easier for the AIED development.

In regards to the potential positive impact of serious games (Connolly *et al.*, 2012), examined literatures on serious games in regards to the positive impact it may have on learning, engagement and skill enhancement. The research revealed that playing serious games can have a cognitive, behavioural and motivational impacts and outcomes.

Yip and Kwan (2006) examined the usefulness of online gaming for undergraduates in vocabulary learning. 100 first year engineering students participated in an experiment where one group learned from two website with games while the control group learnt through traditional activity-based lessons. The preliminary tests did not show a difference between the groups but the group who participated in online gaming showed a significantly better result than that of the control group, after intervention took place. The majority of the students indicated that the games where enjoyable, at a suitable difficulty and that the instructions where easy to follow.

Rossiou and Papadakis (2008) evaluated the effectiveness of an online multi-player game which was used to teach recursive algorithms. The students who participated in playing the recursive algorithm game performed better than the students who participated in tutoring but not in playing the game.

Vahed (2008) examined if a board games could improve the first year students' performance in learning about dental morphology. The group of students who used board games in their learning performed significantly better than those who did not use board games. It was indicated that not only that the students' knowledge of tooth morphology improved but also that the students' attitudes towards the material had a positive impact. The research showed that the learners' favoured the instruction design of the board game as illustrated in the figure below.

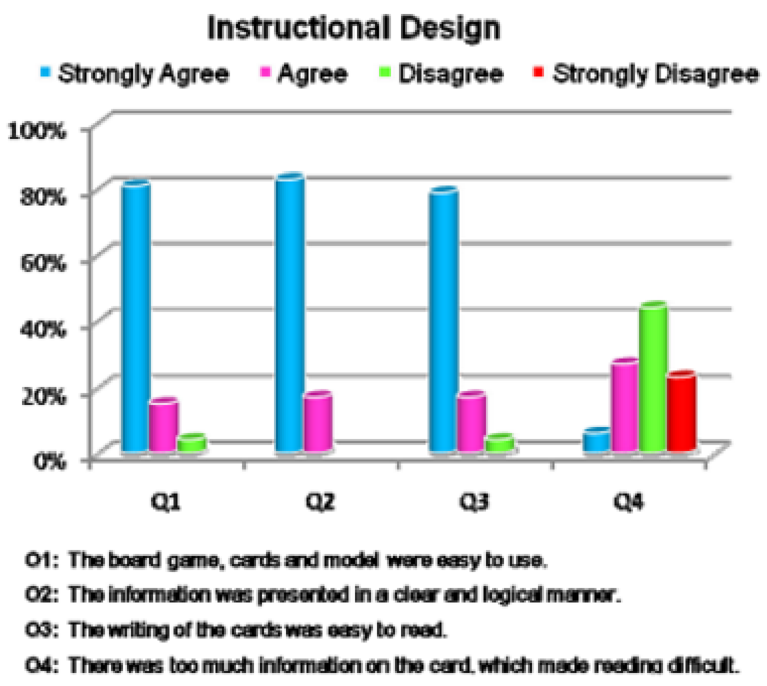


FIGURE 2.1: LEARNERS' RESPONSE TO INSTRUCTIONAL DESIGN

Nte and Stephens (2008) developed a computer game that explained the statistical concept of normal distribution. First year psychology students participated in the experiment and 70% of the 55 participants agreed that the game was useful in their learning on statistics. The game has shown to reduce students' fear of statistics.

The behavioural changes of university students playing a first-person perspective game where examined by (Eastin, 2006). The study indicates that playing a computer opponent rather than human decreases aggressive thoughts. There were clear indications that playing as a female gamer against a male opponents, caused an increase of aggressive thoughts. It was found to be the opposite, when a male gamer played against a female opponents consistently, due to decreases in aggressive thoughts.

2.2.2 Gamification

Gamification is a term used to describe the use of game elements in non-gaming systems to improve user engagement and experience (Deterding *et al.*, 2011). For the process of gamification "to be successful, it must include game design, not just game components. Games are not a replacement for thoughtful experience and interaction design; they are an alternate lens for framing that process" (Deterding, 2012). Coins, badges or simply points are used to strengthen user motivation but "feedback needs to be specific and relevant if it is to motivate and direct" (Siegall, 1988). Thus, for points to be seen as motivational they need to be able to show some feedback on the performance of the user.

Ferrara (2012) argued how design and design elements in games can be used to appeal to players with common motivations. The four elements below are suggested to strengthen motivation.

1. <u>Autonomy:</u> Allowing a player to have a large variety of choices gives the player a sense of freedom. The removal of time restraints can also give the players more autonomy. Games that allow players to skip certain events are deemed to be giving the players autonomy as it allows them to not participate in events that they have little interest in.
2. <u>Competence:</u> Players' interest can be lost if the game becomes too easy. An increase in difficulty as the player progresses is important. Providing the players the ability to lower the difficulty or providing some sort of assistance to help them with the level. The challenge can make playing the game enjoyable, as long as it's possible for players to progress in the game.
3. <u>Social Image:</u> Players may want to show their achievements, badges or game progression to other players. This impacts the players' social image and can motivate them as they maintain or improve their social image by displaying and comparing their achievements with other players.
4. <u>Creativity:</u> Games can be appealing if the player is able to display their creativity within the game. Given the player the opportunity to do certain objectives in a variety of ways can allow the player to be creative.

O'Donovan examined gamification of a university course and stated, "*Our goal with gamification was to improve lecture attendance, content understanding, problem solving skills and general engagement.* [These were] *measured using course marks,* *lecturer evaluations, lecture attendance, and a questionnaire*" (O'Donovan *et al.*, 2013).

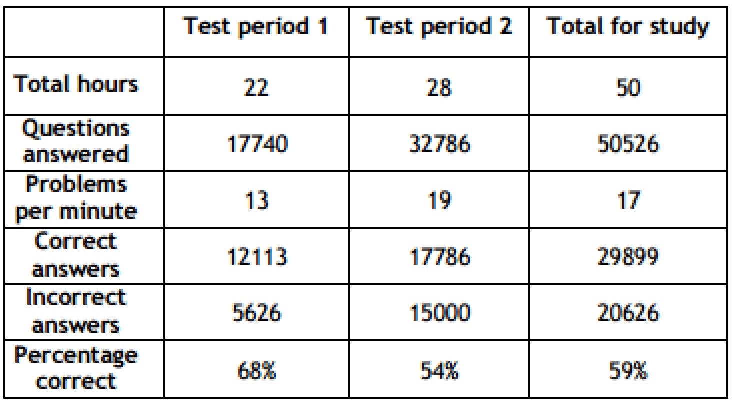
In third level education, gamification can be used to see the progress of the students throughout their course and how well they understood certain subjects. It can be used to identify which students are struggling at which subjects. The gamification approach should not make the students feel like they are doing an exam or test, but rather testing themselves for their own benefit. A way for the students to understand their homework and prepare for any tests or exams. The ability for the student to see their progression could make the student recognise their progress and could end up making the student more encouraged for future progress.

2.3 Game Design

Ludology is the study of games and playing in general, leaving computer games just a particular branch of study. Game design has existed long before computer games. Game design were used by people, even when they don't realise that they are. A good example is young children designing and playing a make belief game. The rules of the game does not need to be strict.

Mary Jo Dondlinger investigated published research related to education game design in specific to design elements conductive to learning and the theoretical underpinnings of game design (Dondlinger, 2007). She noted that a *"number of distinct design elements, such as narrative context, rules, goals, rewards, multisensory cues, and interactivity, seem necessary to stimulate desired learning outcomes*".

Educational video games can require for the users to have a higher order of thinking than simple comprehension and can involve the user to be more strategic or have problem solving skills. Having the learning content relevant to the narrative plot can establish engagement. Lee discussed that a math facts game provided to primary school students encouraged the students to complete more problems than those using worksheets. The students who participated in the use of this game voluntarily increased the difficultly when playing the game, thus led to them completing more difficult problems than those using worksheets (Lee *et al.* 2004). The Table below summarises the data gathered.



	Test period 1	Test period 2	Total for study
Total hours	22	28	50
Questions answered	17740	32786	50526
Problems per minute	13	19	17
Correct answers	12113	17786	29899
Incorrect answers	5626	15000	20626
Percentage correct	68%	54%	59%

TABLE 2.1: SUMMARIZATION OF DATA FROM STUDENTS

Narrative Context

Narratives have been indicated to be an important design element (Dickey, 2006). Dickey believed that in a cognitive framework for problem-solving, this is because the narrative storyline in games offers an atmosphere in which players can recognise and construct fundamental patterns which integrates what is known (such as backstory and rules) with that which is hypothetical yet reasonable within the context of the story".

Waraich (2004) studied a variety of design elements on a game based on computer science architecture. It was concluded that *"For any learning task to be meaningful to the learner they must have both a sufficient context for the learning and motivation to perform the tasks that will help them to learn. We believe that game based learning environments that incorporate a strong narrative can meet these requirements if the learning tasks are appropriately designed and tightly coupled with the narrative"*.

Goals and Rules

Objectives and goals are an important game design element (Waraich, 2004; Zagal, Nussbaum, & Rosas, 2000). Swartout and van Lent (2003) discussed how goals in games can keep players engaged, thus acting as a motivation for the player to continue playing. Three level of goals where elaborated, there were short-term, medium-term and long-term. The short-term relates to a goal that could be accomplished within seconds, such as choosing the correct answer. A medium-term goal relates to something that last minutes, such as finishing a level. The long-term goal is the final overall goal of the game and lasts until the game is completed. These goals are used to provide the players feedback which indicates to them that they have achieved on progressing or completing the game. In educational games, this achievement is the growth of the learner's knowledge.

Interactivity and Multisensory Cues

Another element embedded in game objectives and the narrative context is the interaction between the player and the game environment. A game environment that offers hints and feedback though objects and characters are effective for successful game play (Fisch, 2005). A game that which is highly interactive, that creates tension in the story and the player's freedom of interaction was deemed to be an ideal game by (Swartout & van Lent, 2003).

2.4 Digital badges

A digital badge is a visual representation of an accomplishment. Digital badges don't necessary have to be related to accomplishments only, they could present an affiliation or interest. The use and practice of digital badges is due to a combination of online reputation systems from commerce sites, digital games and real physical status icons, such as ribbons and trophies. The awarding or displaying of a digital badge could occur at various moments. This clearly depends on the developer and what they are trying to achieve. For example, a digital badge could be awarded and displayed to indicate the completion of an objective. Digital badges have also been used by social media sites, such as foursquare.com which awarded badges to users who accumulated credits by frequently using the website. The badges did not only motivate desired behaviours for them, but also provided recognition for the users in the community and a status for them too.

In education, badges are used to encourage students to engage in positive learning behaviours. Before a badge could be recognised as a meaningful learning indicator, it must be linked with an activity or experience during the learning engagement. (Gibson 2006; Mayrath *et al.*, 2012).

A developer of a digital badging system may embed metadata concerning relationships in order to support the meaningful and educational use of digital badges. These metadata may include information about the issuer of the badge, the standards achieved, the activities undertaken or an event experienced, and the quality of the performances.

Digital badges affect education in four key areas. Digital badges have many beneficial impact on education, they can be used for motivational purposes, status recognition and evidence of one's knowledge.

FIGURE 2.2: SAMPLE OF DIGITAL BADGE

2.4.1 Motivation

The learner can be motivated by obtaining a digital badge as an award that indicates their achievement of a learning outcome. By partaking in learning activities for the purpose of obtaining digital badges is also a drive to acquiring knowledge and skills (Abramovich *et al.*, 2013). A series of badges are common in non-educational games, for example the game League of Legends has a ranking system that goes from bronze to silver to gold and so on. But in each one of them, there are five divisions. For example to go from a bronze rank to a silver, one would have to climb up from a bronze five, to a bronze four and so on until they achieve bronze one before they attempt to achieve a silver rank. Although there are more to the rules than mentioned, it meets the purpose to demonstrating a series of badges. In particular to education, series of badges could be used to indicate the progression level and future learning outcomes.

2.4.2 Status recognition

In games, the collection or possession of badges is an indication of credibility as it permits players to participate in new level or activities. In education, credibility of the learners' achievement in their learning is important, as this could be a collection of evidence for a learner to complete a program or to showing their knowledge to employers. Thus digital badges can be used for providing an evidence of the learners' knowledge. The appearance of an expert badge on a network profile can be seen as an indication that, this person could possibly be capable of helping you with a problem one may be facing in an area where this person is an expert at.

2.4.3 Evidence of achievement

Digital badges can contain metadata or a direct link to information about the badges they have collected. The badges can be used as proof of one's achievement but this can be further backed with additional information about the badge, for example when and how it was achieved and who issued it. In education, this is proof of one's learning achievement and can be vital part in one's future employment and education. Gamification and digital badges are being associated with each other more than ever. Both non-educational games and non-game educational programs use digital badges for very similar reasons, so it is no wonder that digital badges are being seen more than ever in educational games.

2.5 Educational Theory

2.5.1 Behaviourism

The scientific study of human behaviour is known as behaviourism. The objective of it is to provide the basis for the predicting and controlling of humans. Ivan Pavlov (1897) published the results of the experiment he carried out on the conditioning after studying digestion in dogs. He accidentally discovered classical conditioning. While this Russian physiologist was looking at salivation in dogs in responsive to being fed, he discovered that the dogs started salivate whenever he entered the room, whether he was about to feed them or not. Classical conditioning involved the learning to associating of an unconditional stimulus that causes a particular response with a new stimulus, in the aim of the new stimulus causing the same response.

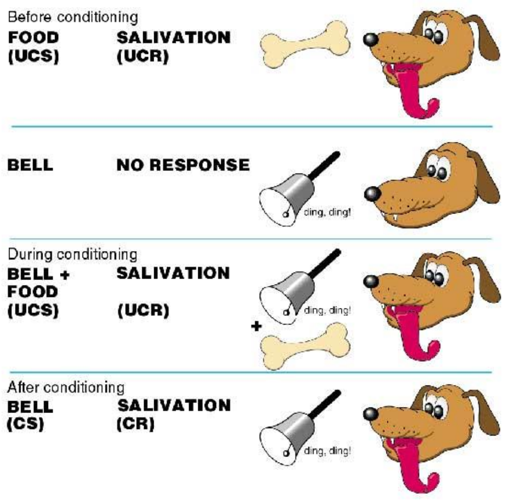


FIGURE 2.3: BEHAVIOURISM

John B. Watson (1878-1958) after observing Pavlov's experiments, Watson believed that classical conditioning could explain aspects of human psychology. Watson and Rayner (1920) showed that classical conditioning could be used to create an irrational fear that is out of proportion to the danger. They proved this by carrying out an experiment with an infant named Little Albert. The experiment started with Little Albert having no fear when being shown animals and masks. A hammer striking a steel was used to induce fear when Little Albert was being reintroduced to these stimuli. After several experiments, Little Albert showed fear from these stimuli even when there were no strikes on the steal bar.

The stimulus-response model had three assumptions. First is that a change in behaviour can be observed through learning. Second is that the elements in the environment determine what one learns rather than the individual learner. The third assumptions is that, *"The principles of contiguity (how close in time two events must be for a bond to be formed) and reinforcement (any means of increasing the likelihood that an event will be repeated) are central to explaining the learning process"* (Smith, 1999).

Classical conditioning can be used in a classroom by teachers to associate positive emotional response with learning to students. As a negative experience of being bullied at school or being punished by teachers could lead to a fear of school.

2.5.2 Cognitivism

Cognitivism is the psychology of learning which emphasizes one's intelligence as an endowment which allows them to form hypotheses and develop intellectually. The concepts of cognitivism involves how one thinks and gains knowledge. This involves the examination of learning, skills, problem solving, memory and intelligence.

A famous and well known cognitive theory is that of Jean Piaget. He was the first to make an organised study of cognitive development. Piaget focused in child cognitive development and proved that children think in different ways when compared to adults. This was important as before Piaget proved his theory, the assumption was that children were only less competent thinkers when compared to adults. Piaget's cognitive theory contains three main components. These are: Schemas, Process of Adaptation, and Stages of Development.

2.5.2.1 Schemas

Piaget gave the definition of a schema as "a cohesive, repeatable action sequence possessing component actions that are tightly interconnected and governed by a core meaning" and that the cognitive structure can be shown by it (Wang, H. 2014). Schemas are seen has blocks or units of knowledge. According to Piaget, schemas become more elaborate and numerous as a child grows.

2.5.2.2 Process of Adaptation

According to Piaget, the process of adaptation was the leading cause of intellectual growth. He indicated that this occurs through Assimilation, Accommodation and Equilibration.

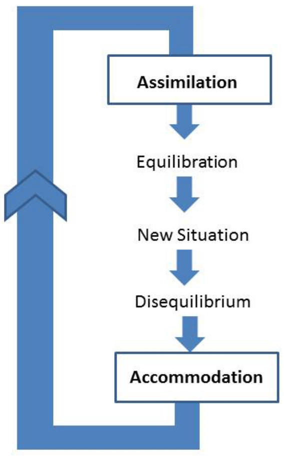


FIGURE 2.4: THE PROCESS OF ADAPTATION

Assimilation: This refers to the use of an existing schema to comprehend a new situation or object.

Accommodation: Accommodation occurs due to the failure of an existing schema and requirement to modify it in order to comprehend a new situation or object.

Equilibration: Piaget described that the learning process is driven by equilibration, this refers to accommodation occurring in order to restore balance due new information not being able to fit existing schemas.

2.5.2.3 Stages of Development

The cognitive development suggested by Piaget is the development of a mental model of the world. He believed that children go through 4 stages of cognitive development. He stated that all children would go through all four stages in the same sequence. Although, he indicated that not all students would progress through the stages at the same rate and even went into noting that some may not even achieve the later stages. The four stages are: Sensorimotor, Preoperational, Concrete Operational and Formal Operational.

Sensorimotor

Object permanence is term used for an importance milestone an infant achieves at the approximate age of 8 months. This is when the infant realises that an object exists even if it is no longer visible. This stage occurs in the first 2 years of an infant's age.

Preoperational

In this stage, children are capable of thinking of objects symbolically. Memory is developed and allows them to differentiate between future and past. Their language becomes more mature. They also develop imagination, allowing them to participate in make –believes. This stage occurs usually between the ages 2 and 7 of a child.

Concrete Operational

Children develop logical thinking and concrete reasoning. Children become to realise that their perspective and feelings of things can differ from others. This stage usually occurs in ages 7 to 11.

Formal Operational

In this stage, they become capable of abstract reasoning such as having more than one idea as they are able to consider possibilities and come up with hypotheses. This stage usually begins from ages 11 or plus.

2.5.3 Constructivism

Constructivism is a philosophy that attempts to understand how people construct knowledge. The questions being asked by constructivism theorists (Hofer and Pintrch, 1997; Jonasson, 1996) are:

- What does knowing something mean?
- How does one know it?
- What effects does it have on ones thinking processes?

There are two dimensions in which major forms constructivism belong to. The first dimension defines the constructivism between an understandings of reality being objective or subjective. According to (Anderson & Kanuka, 1999), this dimension is "*Where educators fall in this first dimension will influence not only how knowledge is constructed (i.e., what are we trying to understand?) But also the way educators will facilitate learners to construct these understandings in the learning process*".

The second dimension defines the constructivism between whether knowledge is socially acquired or individually acquired. Educator that fall in this dimension will influence their learning practices and also teaching. According to (Anderson & Kanuka, 1999), "*assumptions of how we construct knowledge on this continuum will influence the emphasis that will be placed on social interaction, group process, and the learning and practicing of socio-linguistic skills*".

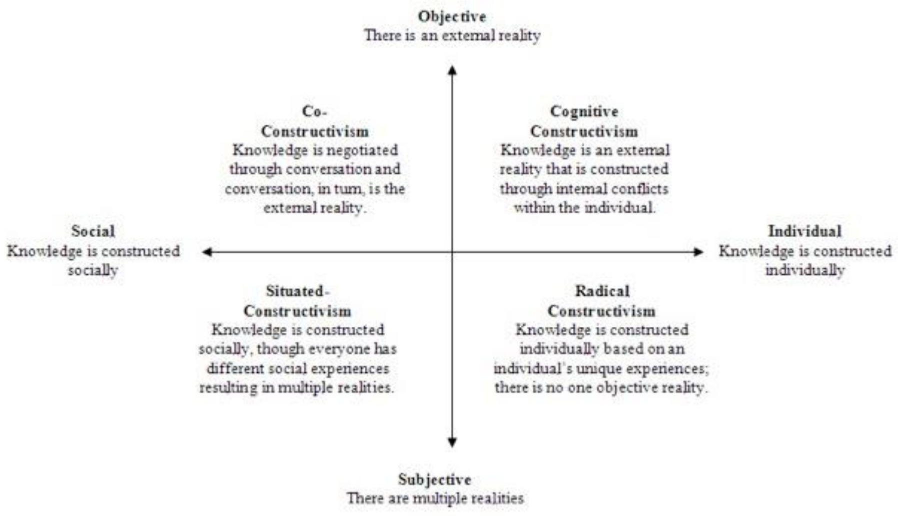

Objective
There is an external reality
Co-Constructivism
Knowledge is negotiated through conversation and conversation, in turn, is the external reality.
Cognitive Constructivism
Knowledge is an external reality that is constructed through internal conflicts within the individual.
Social
Knowledge is constructed socially
Individual
Knowledge is constructed individually
Situated-Constructivism
Knowledge is constructed socially, though everyone has different social experiences resulting in multiple realities.
Radical Constructivism
Knowledge is constructed individually based on an individual's unique experiences; there is no one objective reality.
Subjective
There are multiple realities

FIGURE 2.5: EPISTEMOLOGICAL CONSTRUCTIVISM POSITIONS

2.5.4 Instructional Design

Instructional Design involves the practice of optimising the appeal, effectiveness and efficiency of instructions for knowledge acquisition (Merrill *et al.,* 1996). The determination of the current state and requirements of the learner, identifying the end goal of the instruction and the creation of an intervention to assist the learner, are the processes found in Instructional Design.

Three ideas where expressed by Kurt Lewin in the late 1930s, when dealing with instructional design. These were active learning, cohesive approach and impact of the social environment. These ideas are still used to improve long distance learn, specifically for courses taught through the Web.

Active Learning

It is believed by many educators that when a learner learns a new knowledge by their own has more belief than those presented to them. The educator can yet still feel as if the learner may not learn exactly what they stated unless it is presented to them. The perspective a lecturer has is to make sure the student understands and can show a lack of trust towards the student being capable of understanding it for themselves. This behaviourism adopted by the lecturer is no longer constructive but instructional design instead. A lecturer focusing on the outcomes they want to achieve can end up ignoring the process. The lecturer may rely on lectures in order to achieve his goals but this method may not be effective when it comes to long term learning.

Learning instructions know what the students will do when a content is presented to them and are planned. Students must interact with the instructional content and these activities should support both open ended and self-directed learning. The design must be planned and has to maintain dialogue but ultimately the lecturer must show trust in the students that the outcome will occur in due time. This may have a huge impact on the students' actions in future events.

A Cohesive Approach

Lewin believed that a cohesive approach must be used to support changes of behaviour, cognition and affect. The implementation of this requires the addressing of affective issues which stimulate recognition for the requirement of change for the learner. The planning of tasks that are cognitively challenging are required and also opportunities for action. Motivation is still required to engage with the learner.

Impact of the Social Environment

According to Lewin, changes of perception by the learner of their social environment and self are required before a change in their behaviour, ideas and attitude. He believed that social context were much easier to create changes in than that of an individual one. This theory is supported by (Fischer, 1995).

There are two categories that divide models of instructional design. These are Macro and Micro. Macro is the term used to identify instructional designs which are concerned in the planning and design of entire programs. Micro on the other hand is concerned with the planning and design of a teaching session or an individual teacher.

An example of a Macro model is Bloom's Taxonomy on the cognitive domain from (Bloom *et al,* 1956) which involved *"knowledge and the development of intellectual skills".* This was an attempt to classify forms and levels of learning. Since then, (Krathwohl, 2002) made a modification and revised the taxonomy of the cognitive domain. Being able to create new knowledge within the domain is the top category in the revised taxonomy of the cognitive domain. Forehand (2010), used the image shown below to display the changes made.

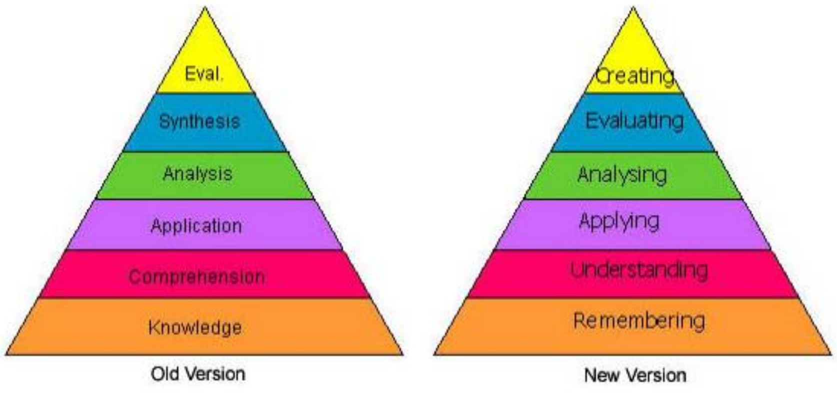


FIGURE 2.6: REVISION OF BLOOM'S TAXONOMY ON THE COGNITIVE DOMAIN

The SOLO (structure of observed learning outcomes) taxonomy developed by (Biggs & Collis, 2014) has fairly clear links to that of Bloom's taxonomy, as it describes levels of complexity in student's understanding of a subject through five stages. Each stage must be attained before progressing to the next as the stages embrace previous stages and add on to them. The stage in sequence are: Pre-structure, Unstructural, Multistructural, Relational and Extended Abstract.

<u>Pre-structure:</u> The acquirement of pieces of information that make no sense as they are not organised.

<u>Unistructural:</u> There are connections being made with the pieces of information but these connections are simple and does not comprehend it fully.

<u>Multistructural:</u> Numerous more connections are being made and the significance of these connections can be recognised but not the significance of all the pieces as a whole.

<u>Relational:</u> The significance of all the pieces as a whole is recognised.

<u>Extended Abstract:</u> Connections are being made beyond within and now beyond the subject area, being able to generalise or transfer ideas underlying a specific instance.

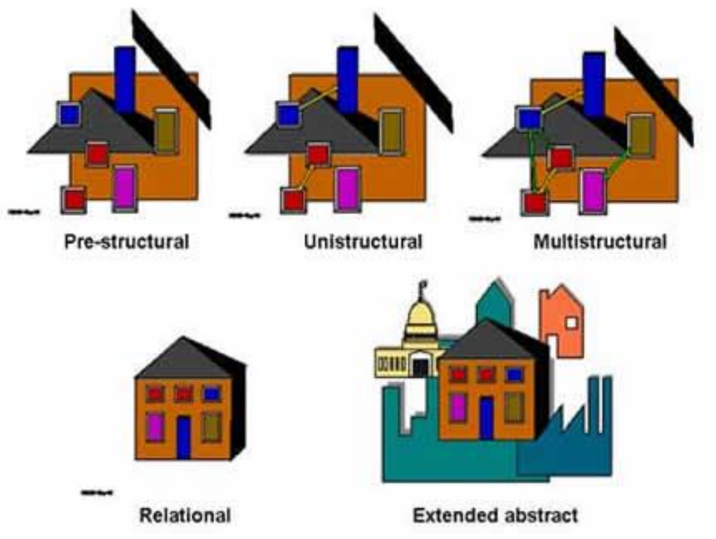


FIGURE 2.7: SOLO TAXONOMY STAGES

2.5.5 Gagne's Nine Events of Instruction

An example of a Micro Model is Gagne's Nine Events of Instruction which was formulated by (Gagne, 1985). These nine events of instruction are as follows;

1. Gain attention

A stimulus is presented to the learner in order to gain their attention. This ensures that the learner is ready to part take in activities.

2. Inform learners of objectives

Objectives are provided before the instruction begins. This allows the learner to be informed of the objectives and explain to them what they are going to learn.

3. Stimulate recall of prior learning

New information can be taught with the help of referencing it to something the learner has experienced or know of already.

4. Present the content

Providing a demonstration after an explanation and the content being organised in a meaningful way provides a more effective instruction.

5. Provide “learning guidance”

Giving guidance to the learner by giving them advice of strategies which can assist them with learning content.

6. Elicit performance (practice)

Initiating the learner to use the new skill or knowledge in order to confirm if the learner has the correct understanding of the concepts.

7. Provide feedback

In order to facilitate learning, student are provided with feedback.

8. Assess performance

The effectiveness of the instructional events are evaluated by examining if the learning outcomes have been achieved.

9. Enhance retention and transfer to the job

In order to assist learners develop expertise, the learners should internalise new knowledge.

2.5.6 Elaboration theory

An instructional design theory known as elaboration theory, claims that materials that are to be learned should be organised. This organisation is done in order from the simplest to the most complex. The aim of this theory is to maximise learning by assisting with the selection and sequence of the material to be learned. Eight steps in elaboration theory was devised by (Reigeluth, 1999).

1. Organizing Course Structure

A complete organisation of the course and how it is structured.

2. Simple to complex

Begin with the simplest ideas for the first lesson and embrace these ideas in future lessons and add on to them.

3. Within-lesson sequence

Simple to complex, general to detailed and abstract to concrete.

4. Summarizers

Summarizers are content reviews.

5. Synthesizers

Assist the learner integrate content elements into a meaningful whole, presentation devices are available. They can help with integrating them into prior knowledge.

6. Analogies

Analogies relate content being learnt to the leaners prior knowledge.

7. Cognitive strategies

Cognitive strategies required for appropriate processing of material can be triggered by a variety of cues – pictures, diagrams, mnemonics, etc.

8. Learner control

The exercise of control over content and instructional strategy is encouraged upon the learner. Effective leaner control is facilitated with labelling and separation of strategy components.

2.6 Teaching Programming

According to (Nagarajan *et al.*, 2010), there is a growing number of students owning computers and having access to the internet and thus e-learning is becoming more popular and effective. Historically, the School of Computing in Dublin Institute of Technology (DIT) allowed lecturers to use different methods of delivering their lectures whether in laboratories, classes or through recorded videos.

In terms of the actual lectures themselves, lecturers were given the freedom to experiment and use different methods teaching programming. There is little systemic evidence available to determine that specific approach for teaching a programming language to learners at an introductory level (Pears *et al.*, 2007).

According to (Milne *et al.*, 2002), students struggle in their understanding of object oriented programming languages until "*they gain a clear mental model of how their program is 'working'—that is, how it is stored in memory, and how the objects in memory relate to one another*".

Muratet, (2009) presents a study around a serious game devoted to the reinforcement of programming skills. Real-Time Strategy, was established to be the most suitable kind of game to support such a serious game. The paper indicates that serious games can be adopted to teach programming.

3 DESIGN: CONCEPTUALISING THE GAME

3.1 Introduction

In this chapter, the design of the game is discussed. Existing games, interviews and literature reviews that have an influence on the design of the game are discussed. The development and execution of assessing student learning prior to the development of the game is elaborated in detail with evaluations. This chapter discusses methodologies used and the game concepts. A paper prototype is used to identify what the game would look like and explain its mechanics. This chapter identifies what the objectives of all the levels are as well.

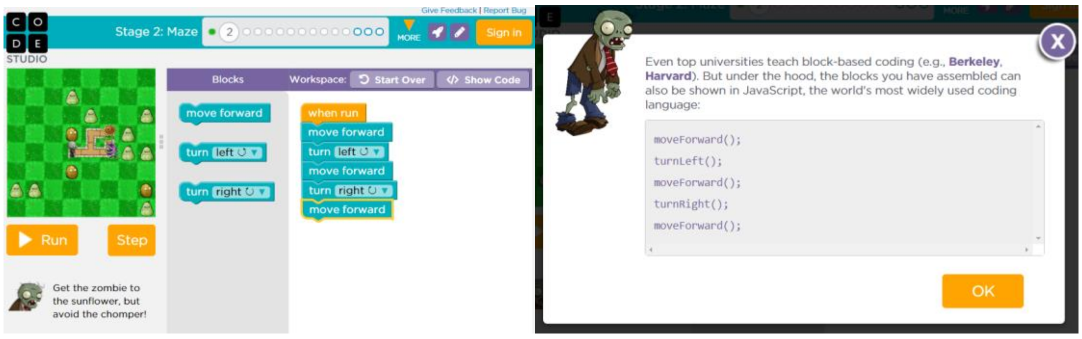
3.2 Existing Games

In this section some existing games that will inspire this project will be discussed.

3.2.1 Code.org

Code.org is an organisation that is registered as public and non-profit. It was first launched in 2013 with the aim to increase the participation of women and underrepresented students in computer science. They aim to provide students an opportunity to learn computer science. Their courses are available in over 40 languages and used in over 180 countries. Currently over 7 million students are enrolled and over 178,000 teachers have signed up to teach courses. The organisation has 28 major partners and corporate supports, such as Google, Facebook, Amazon, Microsoft and Apple (Code.org, 2015).

The majority of the games available on their website are puzzle based. The game mechanics involved the dragging and dropping of puzzle pieces in the correct order to meet the objectives of the game. The games allow the players to see what there code looks like in JavaScript.



Get the zombie to the sunflower, but avoid the chomper!

Even top universities teach block-based coding (e.g., **Berkeley**, **Harvard**). But under the hood, the blocks you have assembled can also be shown in JavaScript, the world's most widely used coding language:

```
moveForward();
turnLeft();
moveForward();
turnRight();
moveForward();
```

FIGURE 3.1: GAME FOOTAGE

3.2.2 Scratch

Scratch is a project currently being run by the Lifelong Kindergarten. Scratch allows users to program their own personal interactive stories, animation and games. Scratch provides a platform for those that want to share their creation. Scratch indicates that it is designed for young people between the ages of 8 and 16 but implies that this doesn't mean those at different age wouldn't use it. Scratch is available in 40 different languages and has been used in 150 countries. They indicate that Scratch has been used by students from the elementary level to college. In relation to this research, they note that it has been used by students in computer science. The contents of their website expresses how important it is for people to code computer programs and how useful Scratch is to students (Scratch.mit.edu, 2015).

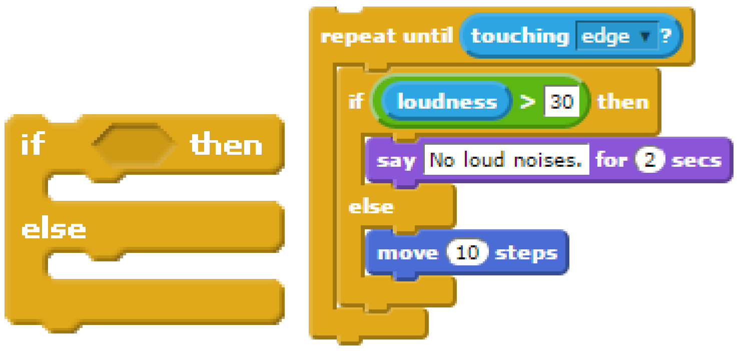


FIGURE 3.2: IF STATEMENTS IN USE

3.3 Incorporation of Gagne's Nine Events of Instruction

The game to be developed will incorporate Gagne's Nine Events of Instruction (Gange, *et al.*, 1985);

1. Gain attention:

The use of graphics and animations to gain the users' attention throughout the game. Providing feedback of the users' performance in a form of a score that updates in real time. The user will be presented with the time it took them to complete a level. A certificate of completion of the game with the users name in it.

2. Inform learners of objectives:

The users will be informed with the use of a tutorial level about the game mechanics and where on the screen the explanation of the objectives of the level are located.

3. Stimulate recall of prior learning:

The users will have to use knowledge they have obtained from their lectures in order to complete each level.

4. Present the content:

The user is provided with pieces they can interact with in order to compose a working Python code.

5. Provide "learning guidance"

The users are introduced to the game mechanics with a tutorial level. The users will be continuously provided with a feedback with every decision they make in order to meet the objectives of each level.

6. Elicit performance (practice)

The users will have to use the knowledge they have obtained from their lectures to obtain a good score. The user will also have to learn from their mistakes, in order not to repeat them at a different level.

7. Provide feedback

The concept of gamification is integrated to providing feedback. The users will be provided with a feedback of their performance during and after every level they complete.

8. Assess performance

The users will have to complete the game. In order to complete the game, they much complete each level. The user must be able to complete all the levels. Even if a user finds a level difficult, they will still be able to complete the level.

9. **Enhance retention and transfer to the job**

After the use of this game, the users are expected to learn from the mistakes they have made. The knowledge they will obtain from the game is part of their course and is important.

3.4 Methodologies

Two methodologies are incorporated together for game design and development process. This two methodologies were both devised and identified by Adams (2014) book on *Fundamentals of Game Design* and another book on *Fundamentals of Puzzle and Casual Game Design* which were released in the same year. The Scott Kim's Eight Steps for developing puzzle game were incorporated with the fundamentals of game design.

3.4.1 Fundamentals of Game Design

This methodology consists of three stages. These stages are the concept stage, elaboration stage and the tuning stage.

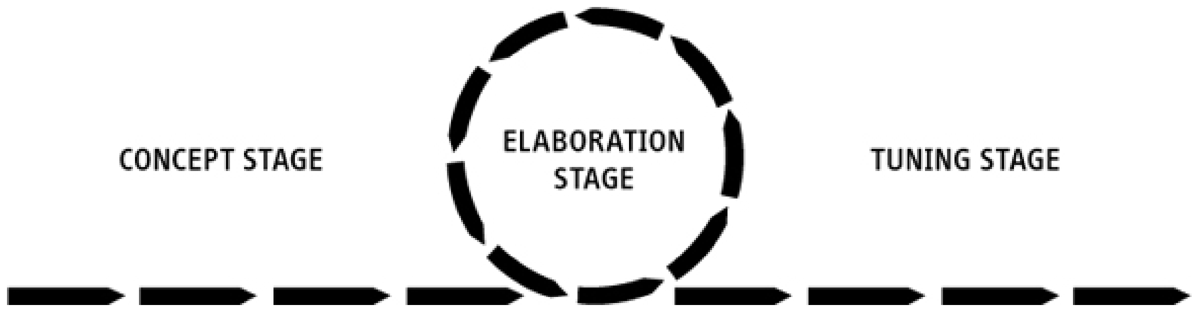


FIGURE 3.3: THREE STAGES OF THE DESIGN

The concept stage involved, getting a concept, defining an audience and determining the player's role. This stage was aligned with the first two steps of Scott Kim's Eight Steps with the exception of defining the audience. In the case of this research, the audience are the students being assessed.

The elaboration stage involves prototyping, defining the primary gameplay mode, designing the protagonist, defining the game world, designing the core mechanics, creating additional modes, designing levels, writing the story, build, test, and iterate. The last 6 steps of Scott Kim's Eight Steps of puzzle game design several of the steps mentioned in the elaboration stage. It is important to note that the design of the protagonist was not part of the eight steps and was deemed to be irrelevant for this project.

In the tuning stage no more features are added. This stage is most usually dictated by the schedule. This stage is also known as feature lock and the design is locked even if more time for design was available. The work involved in this stage include, small adjustments to levels and the core game mechanics, but it is important to note that, no new feature should be added to the game. The game is published as this level, this is usually a process of subtraction as imperfections are removed.

3.4.2 Scott Kim's Eight Steps

Adams (2014) mentioned Scott Kim, "*At the 1999 Game Developers' Conference he gave a lecture entitled "The Art of Puzzle Game Design*" in which he identified eight steps in puzzle game design". These eight steps are as follows;

1. Find inspiration.

Inspiration can come various sources. Other games can be a source of inspiration. Television shows, movies and books are just some more examples.

2. Simplify.

An idea for a puzzle game could be simplified by understanding the main concepts of the game. One must know which concepts are the most essential areas of the game and understanding the difficulty of them. It is important to remove irrelevant details that may end up making the design or the game more difficult. If possible, controls should be simplified as well.

3. Create a construction set.

This step is all about trying or playing the game before fully developing it. You are not required to code the game in this step. Using a paper prototype of the game, one can adjust the rules and design the levels.

4. Define the rules.

This step is an important part of the design. Some of the questions being asked in this step are such as;

- Is there a board in the puzzle game, and if so, what kind, for example, is it a grid.
- The shape, colour or images related to the board or the pieces of the puzzle.
- The attributes of the pieces.
- The location of the pieces.
- The rules of what pieces can move and what cannot move.
- What effects whether a piece can move or not.
- What effects moving a piece has.
- The conditions that need to be met for victory or to achieving the goal.
- Whether the goal has to happen partially or exactly.

5. Construct the puzzles.

In this step the puzzles or levels are constructed. There are many things to consider, such as whether one puzzle needs to be completed before attempting the next puzzle or how challenging the puzzle should be. There has to be a moment whereby the player realises how the puzzle works and how it can be solved. It is important to make sure that the game is not unfair, as players should not be kept in the dark about an important aspect of the game.

6. Test.

This step is all about testing. The difficulty of the game and whether the players can solve the puzzle. Testing can help to discover errors in the rules. The user interface is examined to determine how smooth it is and that it does not lead to frustration, this can be done by letting people play the game.

7. Devise a sequence.

The sequence of all the puzzles are devised in this step. The most common arrangement is a sequence that has the levels increase in difficulty and the player progresses. Another arrangement is a sawtooth shape, the sequence of the puzzles is all about having an easy puzzle between difficult puzzles. It alternates from an easy to a difficult puzzle.

It is also important to decide, if the players should have the freedom to play puzzles in a sequence or to give them the freedom to choose.

8. Pay attention to presentation.

This step is the final step and deals with details such as graphical style, user interface, animation, sound and so on. The presentation of the levels are decided here.

3.5 Game Concepts

From the existing games investigated, a drag and drop type of game have shown to be successful in teaching some basic and even advanced concepts of programming. This type of game mechanic was taken into consideration for this project.

The “Programming and Algorithms” module is a two semester long module that teaches students programming using both pseudocode and python. The first semester module contents were under investigation as it started at the similar time as this project.

This research was limited to a specific time, only the student learning of the first half of the semester could be evaluated. The students were expected to have knowledge of the following concepts which were taught at the weeks indicated.

- Week 1: Students were introduced to pseudocode and python.
- Week 2: Students were taught about sequence and variables.
- Week 3: Students were introduced to selection, which dealt with teaching students IF statement, CASE statement and Boolean logic/
- Week 4: Introduced the students to iteration. This included students learning about loops, such as WHILE, FOR and DO loop.
- Week 5: At this stage, students are introduced to some common algorithms such as, Prime Numbers and Fibonacci Numbers.

In order to identify which concepts should be included in the game, two methods were devised. One was to interview lecturers that have taught first year students a programming module. The second was to test the knowledge of the students by giving them a test.

Three lecturers who have taught a first year module were interviewed in a semi structured and audio recorded interview. Two interviews lasted longer than 40 minutes while the remainder lasted approximately 15 minutes due to the lecturer specifying for a short interview.

One lecturer indicated that, students first big obstacle is understanding complicated if statements. The lecturer believed, “*without a doubt the one area every year students struggle with are pointers, memory addresses and accessing memory, dynamic memory allocation*” and proceeded to indicate that the reason of this is that “*student are finding it difficult to understand where variables have an address in memory and that you can access the contents of an address these particular approach*”.

Another lecturer was adamant that there is no specific programming concept students struggle with the most. That lecturer indicated that all students vary and that some students will understand certain concepts faster than others but believes that most students struggle when it comes to combining different concepts together.

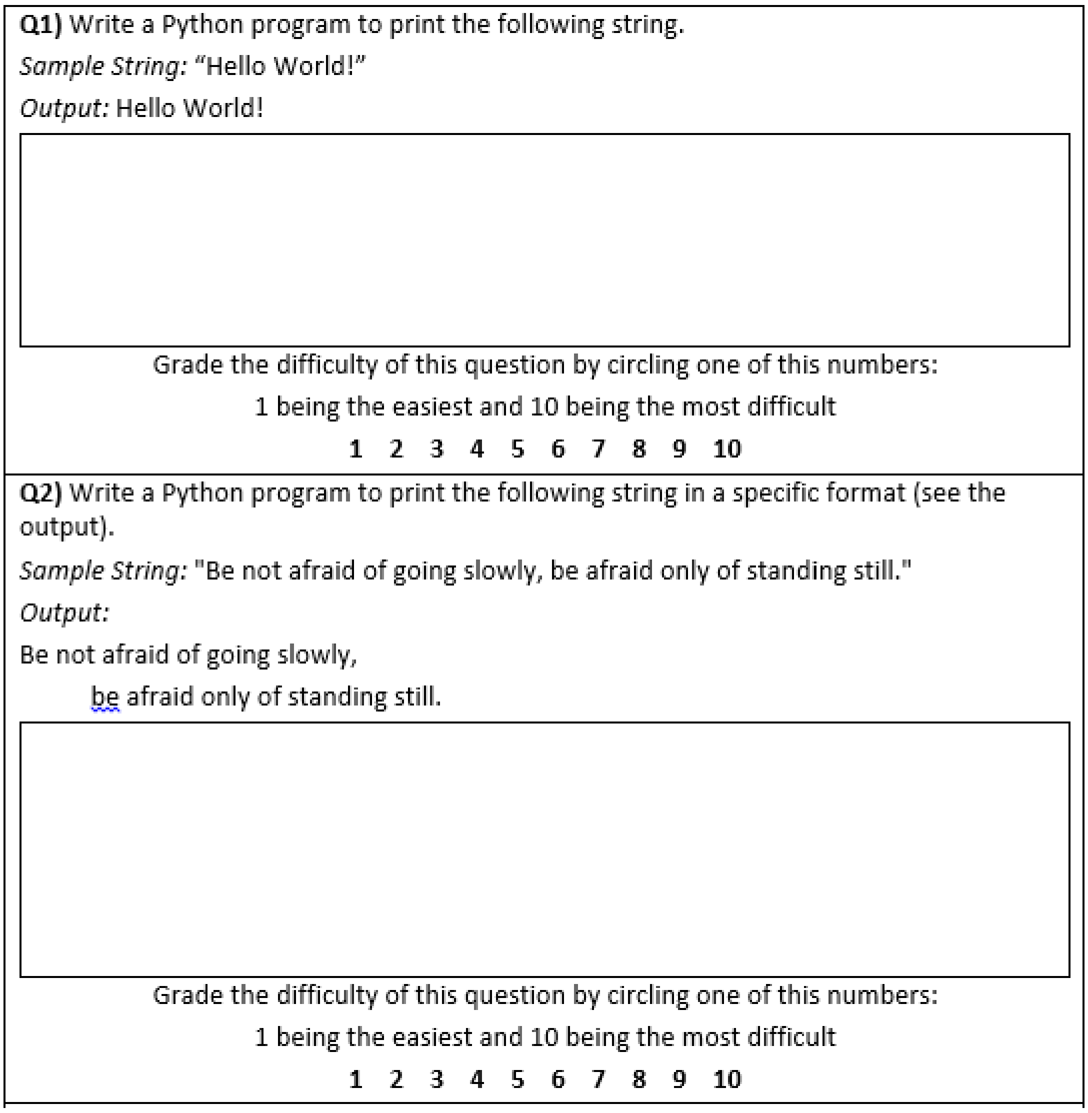
The third lecturer responded with similarly to the second lecturer with the belief that no one single concept causes a big obstacle in first year students learning a programming language. The lecturer believed that student struggle to combine the concepts they have learnt, "*I think putting it all together, rather than anything else*". The lecturer referred to a recent experience where students were able to do produce code to two different problems but when the problems are combined, they were not able to create a code that incorporated both functionalities.

The combination of different concepts being a challenge to students were shared by two of the three lecturers. The combination of different concepts in programming are taken into consideration but before doing so, the actual knowledge of the students need to be identified, as it was not ideal to assume students' knowledge and later find out that their level of knowledge was not aligned with the assumptions made.

In order to identify the students' current knowledge, students needed to be assessed. The students were given a written test which they were not prior notified of, at the end of their laboratory on the 9th of November 2015. A handout of the test was given to the students at 5:30pm and they were given 30 minutes to complete it.

The questions in the test were devised in order to assess students' knowledge of the material they were taught in their first 5 weeks of programming. A large focus was placed on questions related to the material they were taught in the first 3 weeks. In order to identify what questions to ask students in the game, it was important to evaluate whether the students understood the most important and basic programming skills they were taught. Students also graded the difficulty for each question by providing a number between 1 and 10 where 10 indicated the question being the most difficult.

The first and second questions asked are as follows;



Q1) Write a Python program to print the following string.

Sample String: "Hello World!"

Output: Hello World!

Grade the difficulty of this question by circling one of this numbers:

1 being the easiest and 10 being the most difficult

1 2 3 4 5 6 7 8 9 10

Q2) Write a Python program to print the following string in a specific format (see the output).

Sample String: "Be not afraid of going slowly, be afraid only of standing still."

Output:

```
Be not afraid of going slowly,
        be afraid only of standing still.
```

Grade the difficulty of this question by circling one of this numbers:

1 being the easiest and 10 being the most difficult

1 2 3 4 5 6 7 8 9 10

FIGURE 3.4: QUESTION 1 AND 2 FROM THE WRITTEN TEST

The purpose of the first question is to identify whether the students could print a string. There are a variety of reasons for asking the first question. The version of python students were taught was the latest version of Python but the answer to this question would had been different if students were asked for a solution that would compile for an older version of python, such as 2.x. Students may be unknowingly learning a different version of Python when they search for information online. In the version students were taught, they needed to use the print statement with parenthesis as in *print ("Hello World!")*.

It was also important to identify if students are aware how important capitalizing words are in programming. This questions allows to identify if students use capital letter(s) to call the built-in function *print()*.

The second question varies from question 1 slightly as it evaluates if students know how to indicate for a new line and to tab. The students' knowledge of whether they knew the difference between the backslash "\" and forward slash "/".

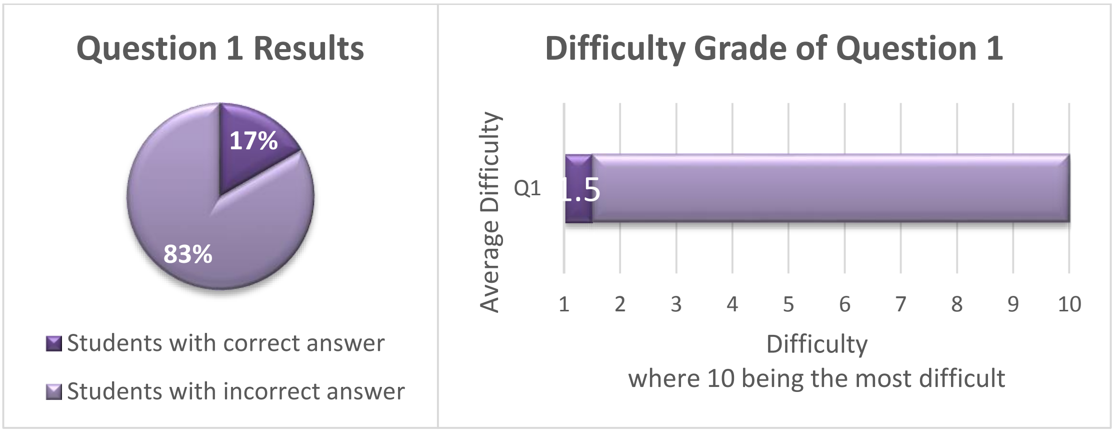


FIGURE 3.5: RESULTS FOR QUESTION 1

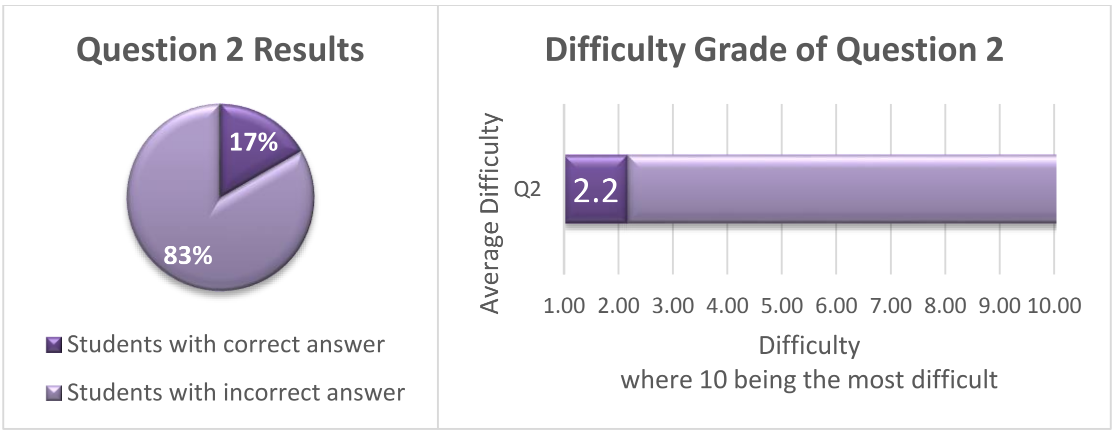
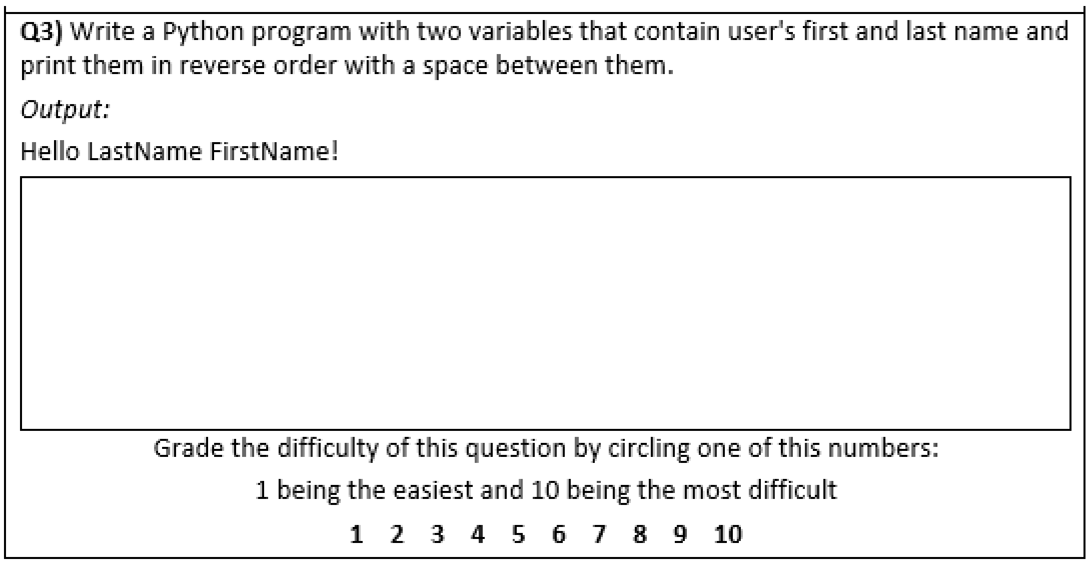


FIGURE 3.6: RESULTS FOR QUESTION 2

The majority of the students failed to answer both question 1 and 2. All of the students who answered question 1 correctly were able to answer question 2 correctly. Some students who failed to answer question 1 correctly were unable to identify the difference between the backslash and forward slash.

The major issue identified with the incorrect answers were as follows, the use of capitalization of the print function, the lack of parenthesis and the incorrect use of forward slash.

The third question is as follows;


Q3) Write a Python program with two variables that contain user's first and last name and print them in reverse order with a space between them.

Output:

Hello LastName FirstName!

Grade the difficulty of this question by circling one of this numbers:

1 being the easiest and 10 being the most difficult

1 2 3 4 5 6 7 8 9 10

FIGURE 3.7: QUESTION 3 FROM THE WRITTEN TEST

This question was designed to assess the students' knowledge of declaring variables and ability of printing out variables alongside with a string.

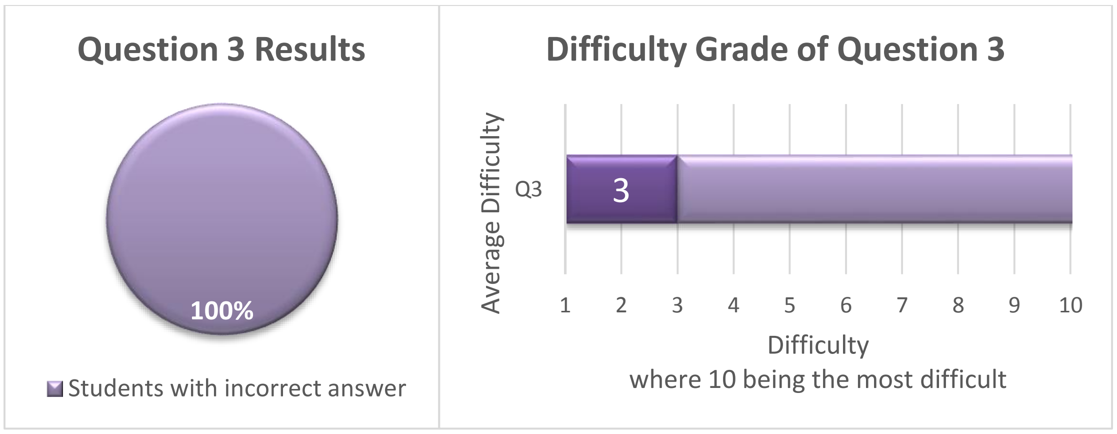
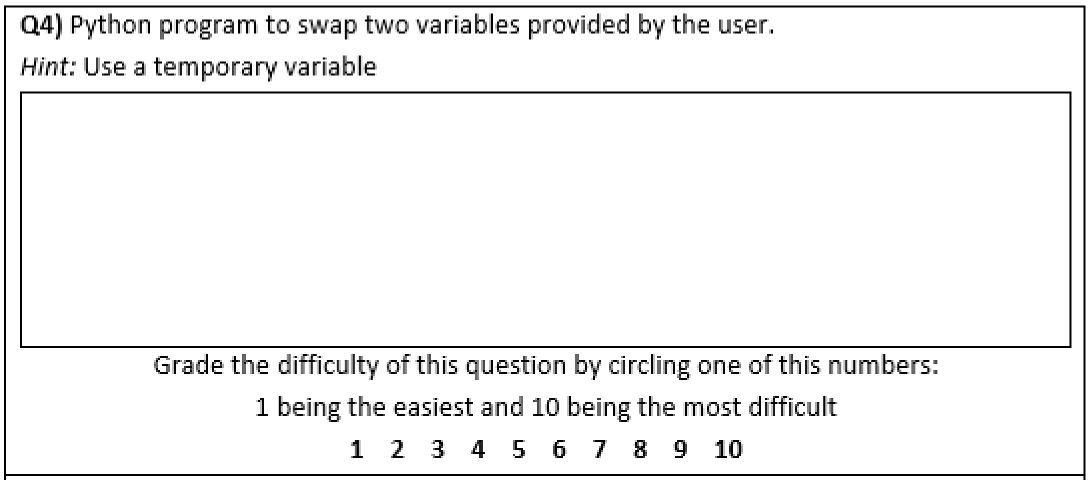


FIGURE 3.8: RESULTS FOR QUESTION 3

The results of questions 3, showed that none of the students were capable to writing a code that was correct or that would compile. The same mistake as in the previous two were observed.

The students gave a higher grade in difficulty for this question when compared to the previous 2 questions. The grade of three out of ten in terms of difficulty is relatively low when compared to the fact that no students got the question write. This is an indication that some students believed their answer was correct and some knowing that the actual question was in fact not that difficult to other problems they might have solved in there laboratories.

Question four is as follows;


Q4) Python program to swap two variables provided by the user.
Hint: Use a temporary variable

Grade the difficulty of this question by circling one of this numbers:
1 being the easiest and 10 being the most difficult
1 2 3 4 5 6 7 8 9 10

FIGURE 3.9: QUESTION 4 FROM THE WRITTEN TEST

The question was designed to test students' knowledge of variable to a greater level than that of question three. This question was expected to be more difficult than that of question three.

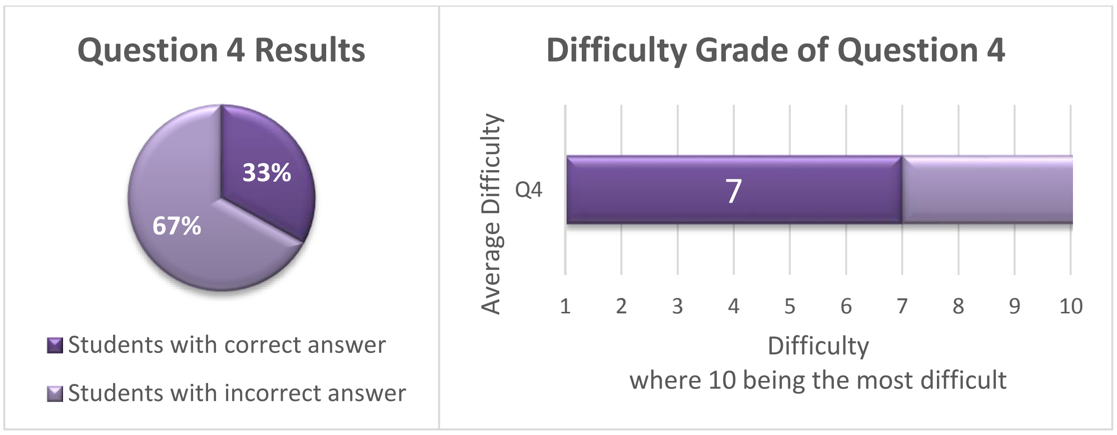
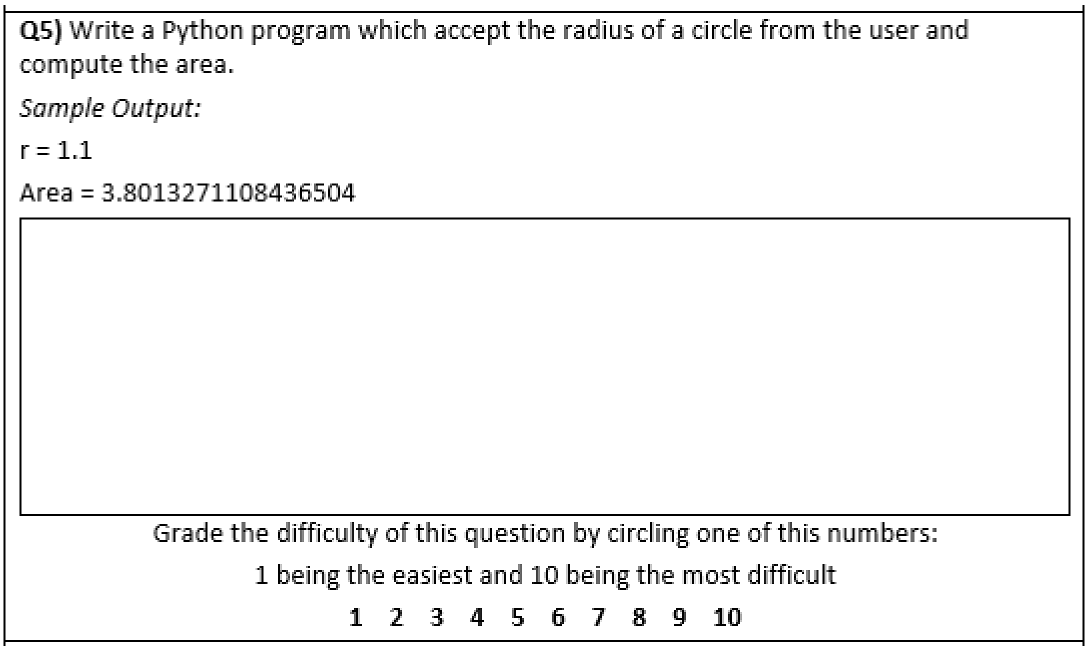


FIGURE 3.10: RESULTS FOR QUESTION 4

The results obtained were unexpected. 33% of the students provided the correct answer. The reason why this result was unexpected is due to the fact that no student was able to get the correct answer for the easier variables related question yet 33% of them were able to get question 4 right.

This lead to inspection of why and where the students went wrong when answering the third question. The students who got question four correct, had their question 3 incorrect answers inspected. It was apparent that the students struggled with the print function rather than with variables.

Question five is as follows;



Q5) Write a Python program which accept the radius of a circle from the user and compute the area.

Sample Output:

```
r = 1.1
Area = 3.8013271108436504
```

Grade the difficulty of this question by circling one of this numbers:

1 being the easiest and 10 being the most difficult

1 2 3 4 5 6 7 8 9 10

FIGURE 3.11: QUESTION 5 FROM THE WRITTEN TEST

This question was designed to identify whether the students are capable of obtaining user inputs, being capable of calculating values and printing out the result in the correct format.

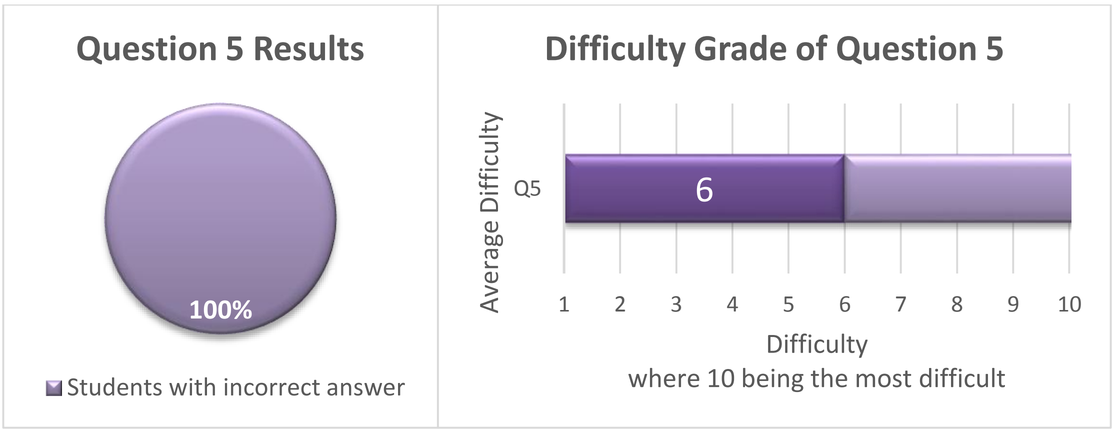
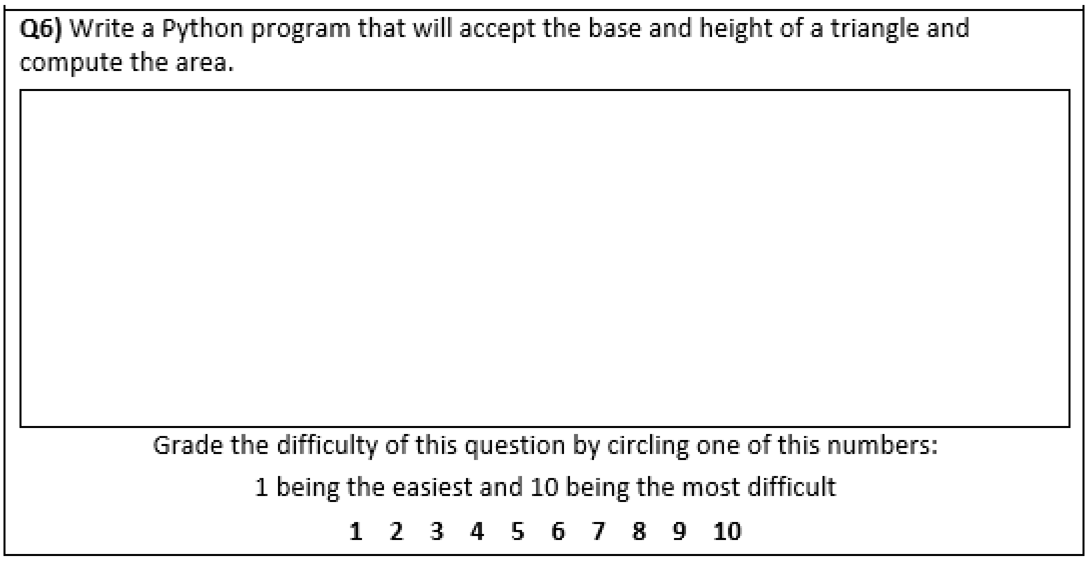


FIGURE 3.12: RESULTS FOR QUESTION 5

None of the students were able to provide the correct solution. The majority of the mistakes were due to the student incapability of obtaining information from the user in a float value as most made a mistake in obtaining the radius value as a string instead. One other important information found was that, not all the students knew the equation for calculating the area of a circle, as one student indicated that he did not know the equation. This indicated that if the students were to get this question in the game or in the future that this question will need to show the students the equation.

Question 6 is as follows;



Q6) Write a Python program that will accept the base and height of a triangle and compute the area.

Grade the difficulty of this question by circling one of this numbers:

1 being the easiest and 10 being the most difficult

1 2 3 4 5 6 7 8 9 10

FIGURE 3.13: QUESTION 6 FROM THE WRITTEN TEST

This question was similar to that of question five but is used to identify whether if the students would be able to answer a similar question to that of question 5, but with an easier equation thus calculation.

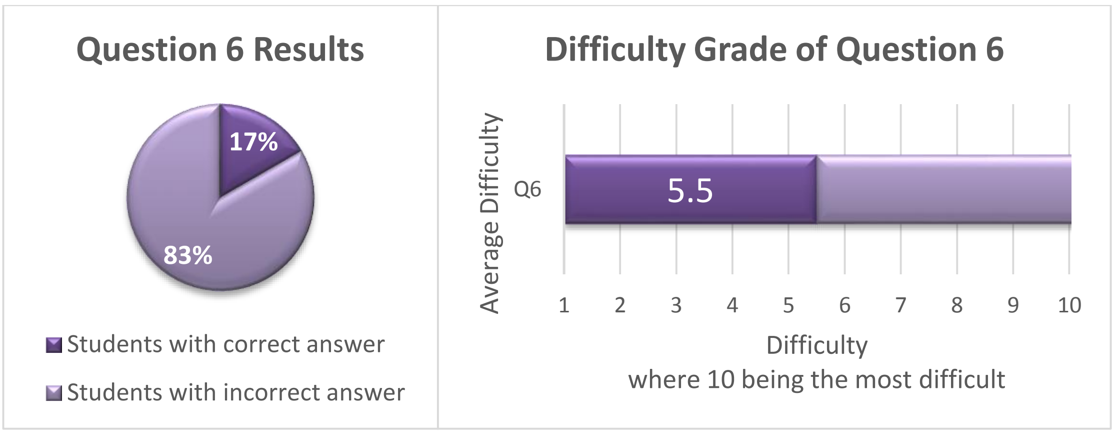
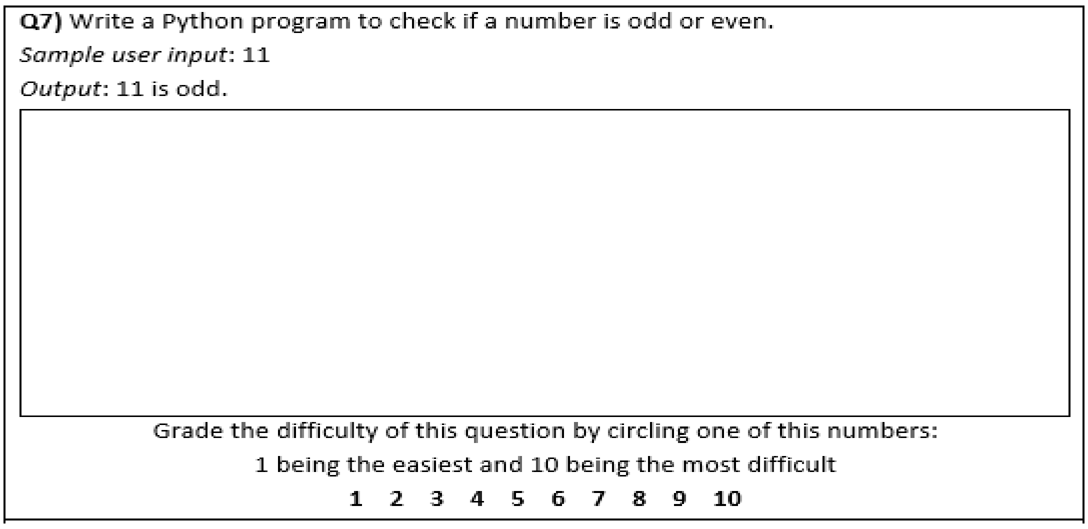


FIGURE 3.14: RESULTS FOR QUESTION 6

This question proved that, the some students struggle with the calculation in question five yet they were capable of answering another similar question with an easier equation. None of the students were able to answer a similar question with a difficult equation yet when the equation was simplified, 17% of the students were capable of answering the question. It is also important to note that, the same student that indicated their lack of knowledge for the equation of question five also indicated their lack of knowledge of the equation for this question. Although, this is only one student, it is important to provide the equation for this question if this question ever gets asked again.

Question 7 is as follows;



Q7) Write a Python program to check if a number is odd or even.

Sample user input: 11

Output: 11 is odd.

Grade the difficulty of this question by circling one of this numbers:
1 being the easiest and 10 being the most difficult
1 2 3 4 5 6 7 8 9 10

FIGURE 3.15: QUESTION 7 FROM THE WRITTEN TEST

This question was designed to assess the students' knowledge of IF statements and their logical thinking.

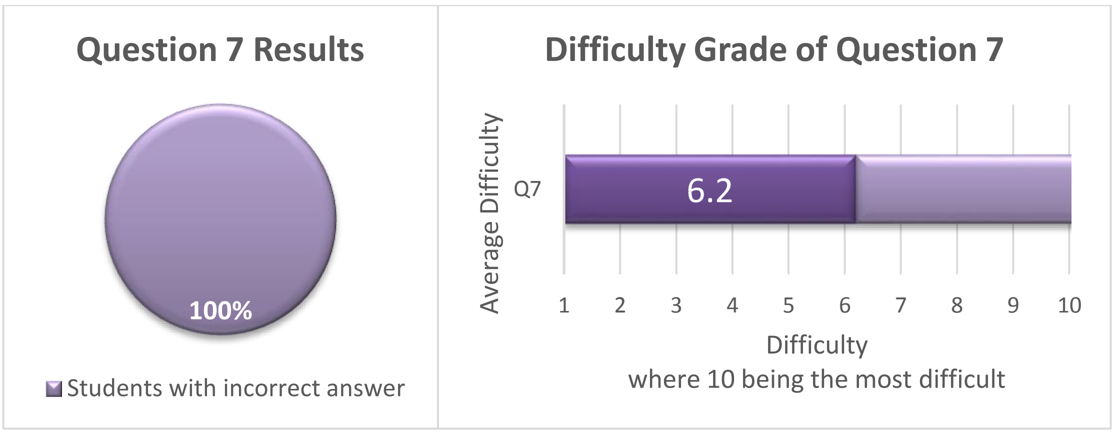


FIGURE 3.16: RESULTS FOR QUESTION 7

None of the students where capable of answering this question correctly. There were many various mistakes found in the students' answers. There was no specific or noticeable pattern of mistakes. In some cases, students were unable to even attempt this question.

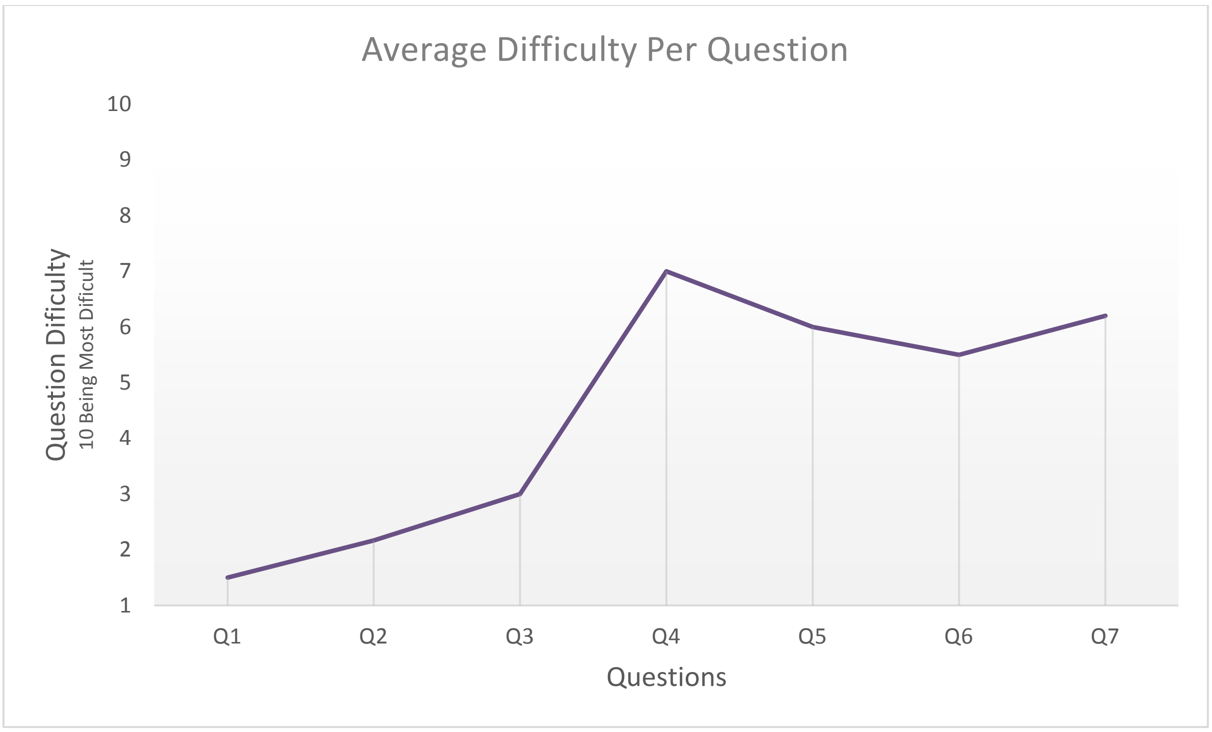
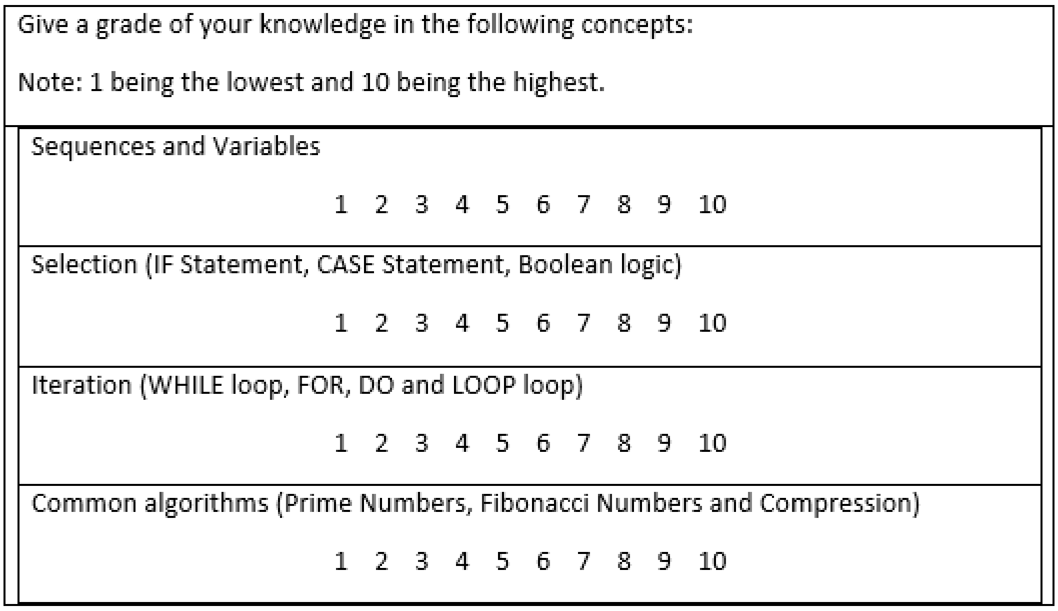


FIGURE 3.17: GRAPH OF AVERAGE DIFFICULTY PER QUESTION

The students also graded their knowledge of the different concepts they were taught. The image bellow show the method in which this was done.


Give a grade of your knowledge in the following concepts:

Note: 1 being the lowest and 10 being the highest.

Sequences and Variables 1 2 3 4 5 6 7 8 9 10
Selection (IF Statement, CASE Statement, Boolean logic) 1 2 3 4 5 6 7 8 9 10
Iteration (WHILE loop, FOR, DO and LOOP loop) 1 2 3 4 5 6 7 8 9 10
Common algorithms (Prime Numbers, Fibonacci Numbers and Compression) 1 2 3 4 5 6 7 8 9 10

FIGURE 3.18: METHOD FOR STUDENT SELF-ASSESSMENT

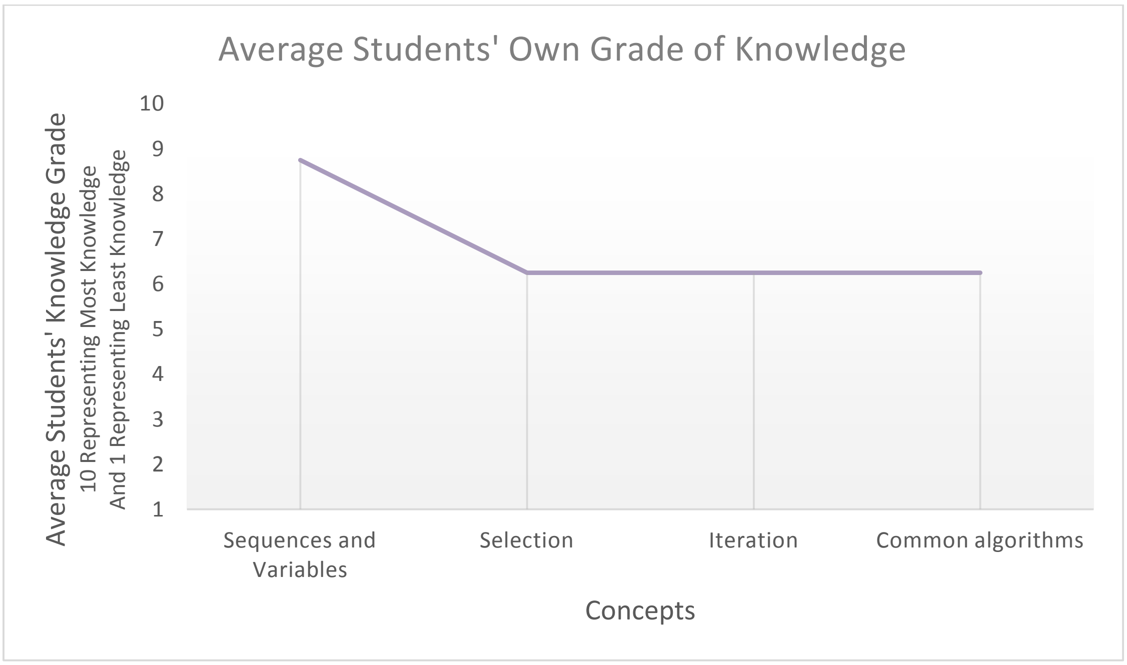


FIGURE 3.19: GRAPH OF AVERAGE STUDENTS' OWN GRADE OF KNOWLEDGE

The average students' own grade of knowledge indicate that student believe they have a very good knowledge in Sequences and Variables. The remaining three concepts obtained identical average value indicating that students have equal amount of knowledge of each concept. This result requires follow up with student being asked to grade their knowledge on a later date after they have been assessed through a game.

3.6 Paper prototype

Before creating a paper prototype of the game, the resolution of the game was decided. The default resolution for the web player are 960 pixels in width and 600 pixels in height. This default resolution was deemed to be appropriate for the monitors the students use in their laboratory.

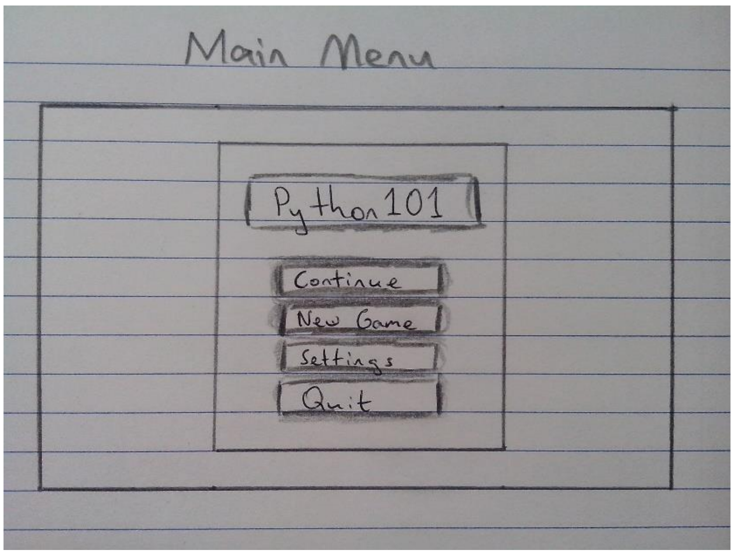


FIGURE 3.20: GAME MAIN MENU PROTOTYPE

The game was given the name Python101. The name was created by mixing Python with 101, where Python is the programming language the game is assess student knowledge in and 101 is usually used in course codes in many parts of the world for an introduction to a course. Rather than having the students directly entering their names, it was decided to have a main menu in order to present a game like appearance. This was because this would be the first thing the students would see.

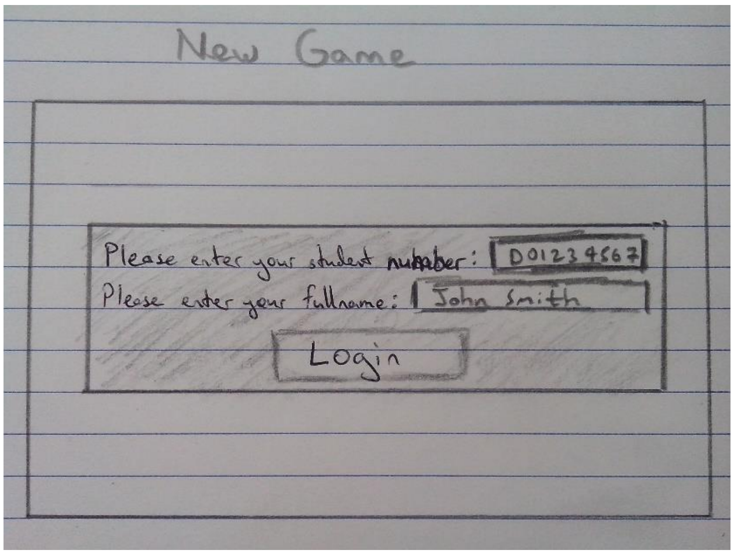

New Game
Please enter your student number: D01234567
Please enter your fullname: John Smith
Login

FIGURE 3.21: PLAYER REGISTRATION PROTOTYPE

Once the students have selected a new game, students are asked to provide student name and student number. This particular scene should be not be clustered and easy for the students to understand what is going on and what they are required to do.

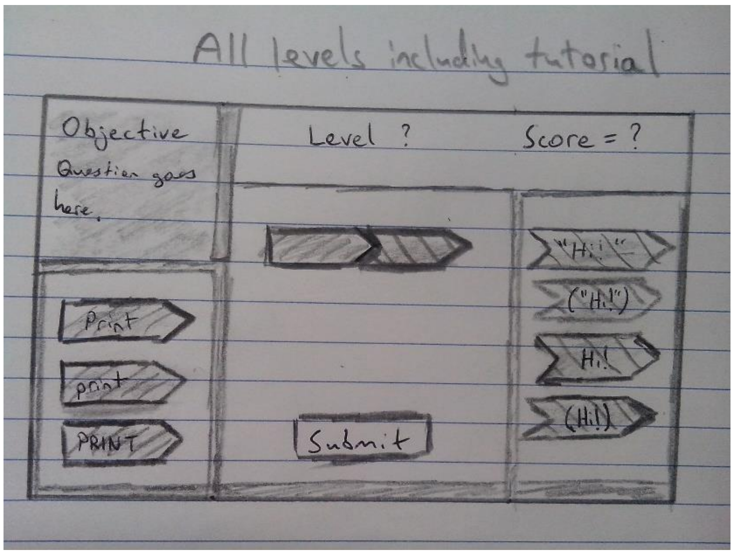


FIGURE 3.22: GAME LEVELS PROTOTYPE

All levels and the tutorial level would have the same structure. Once the student has completed the tutorial level, it is important to keep the structure the same, in order to ensure that the student becomes familiar with the game and its mechanics. The objectives of each level will be located at the top left of the game. The level they are at will be displayed in the top centre of the game. The current up to date score will be displayed on the top right of the screen. There would be two types of puzzle pieces that differ in shape and colour. They would be located on the left and right side of the screen. The middle area of the screen is where a puzzle to be solved lays, the students fill in the puzzle by dragging and dropping the correct puzzle pieces. When the correct puzzle piece is dropped students would be notified with a colour change of that particular puzzle area and an increase in score. Incorrect pieces would not be able to drop in the puzzle area and would be return to its original location, the puzzle area would notify this through a change in colour and a deduction of the score. Students would be required to submit their work when they have completed the puzzle.

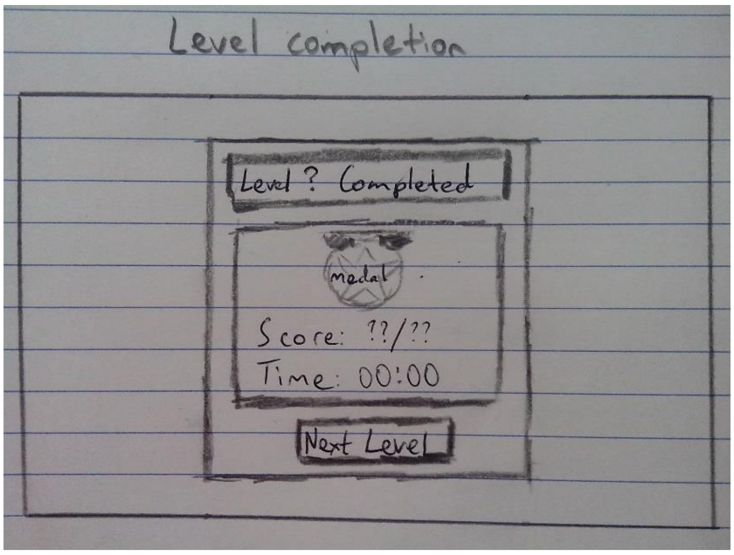


FIGURE 3.23: GAME REWARD SCENE PROTOTYPE

Students would be provided with notification that they have completed a level with the score and time it took them, when they finish all levels including the tutorial level. This would be the location where digital badges would be assigned to students for completing a level. Once the student is ready, they can proceed to the next level.

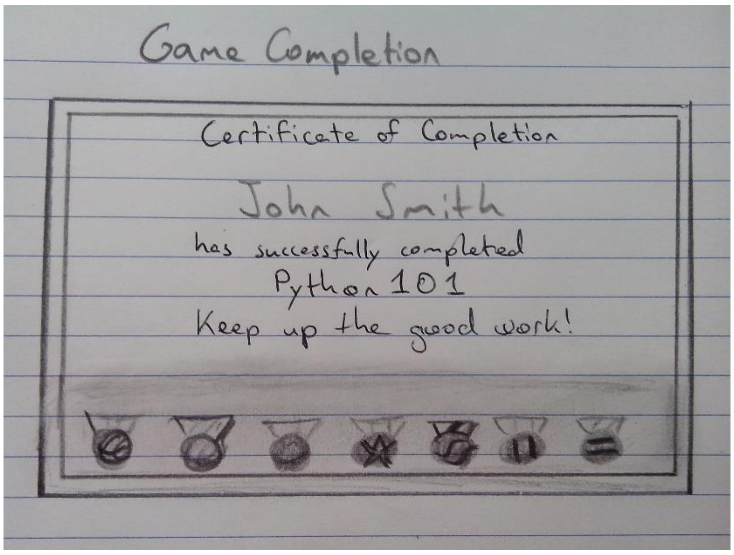
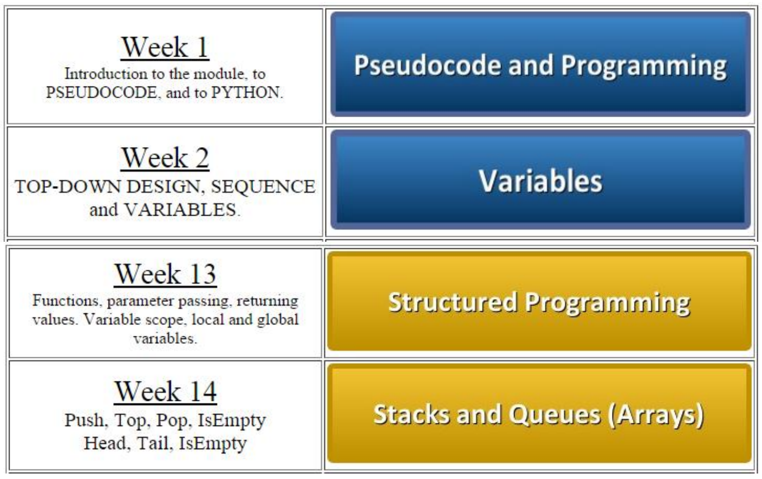


FIGURE 3.24: FINAL AWARDS SCENE PROTOTYPE

This would be the last thing the students would see as it indicates that the game is complete. In order to give the students a sense of accomplishment, this would be a scene where the game displays a digital certificate that shows the students name and accomplishments with all the digital badges they have collected.

3.7 Level Design

The choice of colour for the two type of puzzle pieces was important. An investigation discovered that the colours blue and yellow were associated with the students' experience with Python in two ways. One was that the website where the students obtain their lecture notes divided the contents of the first semester with blue images and second semester with yellow images. The second was that the students see the blue and yellow symbol used by Python on websites and Python IDLE.



Week 1 Introduction to the module, to PSEUDOCODE, and to PYTHON.	Pseudocode and Programming
Week 2 TOP-DOWN DESIGN, SEQUENCE and VARIABLES.	Variables
Week 13 Functions, parameter passing, returning values. Variable scope, local and global variables.	Structured Programming
Week 14 Push, Top, Pop, IsEmpty Head, Tail, IsEmpty	Stacks and Queues (Arrays)

FIGURE 3.25: SEMESTER 1 AND 2 LECTURE NOTE LINKS

FIGURE 3.26: PYTHON SYMBOL

FIGURE 3.27: COLOUR SCHEME OF PUZZLE PIECES

The following table contains the objectives set for all levels. The objectives were created to either be the same or very similar to questions student answered in the written test. The results show that the students struggled with these questions and making the objectives more difficult may led to discouraging students and causing a negative attitude towards the game and the module.

Tutorial Level	Drag and Drop the correct blocks towards the empty blocks in the middle of the screen. Assign the string "Lamborghini" to the variable car.
Level 1	Drag and Drop the correct blocks to print the following string. String: "Hello World!" Output: Hello World!
Level 2	Drag and Drop the correct blocks to print the following string. Output: Hello World!
Level 3	Drag and Drop the correct blocks to print the following string. String: "Hello World!" Output: Hello World!
Level 4	Drag and Drop the correct blocks to print the following string. Output: Hello World!
Level 5	Print the variables that contain the first and last name in reverse order with a space between them. Variable a, should contain the first name and b should contain the last name. Output: Hi! LastName FirstName

Level 6	Print the sum of the two variables. Output: 7
Level 7	Complete the python program that swaps two variables provided by the user.
Level 8	Complete the python program that swaps two variables provided by the user. (NOTE: Solved differently to Level 7)
Level 9	Complete the python program which accept the radius of a circle from the user and compute the area. Equation: Area = pi times the square of the radius Sample Output: r = 1.1 Area = 3.8013271108436504
Level 10	Complete the python program that will accept the base and height of a triangle and compute the area. Equation: Area = (base x height)/2
Level 11	Complete the python program to check if a number is odd or even.
Level 12	Complete the python program to check if age is 18 or over. Output (if age is 18 or over): Can Enter Output (if age under 18): Cannot Enter

TABLE 3.1: LIST OF ALL THE OBJECTIVES OF THE GAME

Ferrara (2012) argument of the importance of autonomy as discussed in the literature review chapter was taken into consideration. A decision of not putting a time restraint was made in order to give the players a sense of freedom.

3.8 Conclusions

The investigation of existing educational games to teach program served as inspiration for the game type to be a puzzle game which contained a drag and drop mechanism. The importance of digital badges in games, which was mentions in the previous chapter led to its inclusion in the game to be developed. Gagne’s Nine Event of Instruction was incorporated to the design of the game.

In order to achieve some of the objectives of this research, this chapter was able to assess students’ learning with the use of written tests. This allows for future written test to be used in order to identify if assessing students through gamification has any other impact on the students’ knowledge. A game that can be used to assess first year computer science students’ learning the programming language Python was devised.

4 DESIGN: SOFTWARE ARCHITECTURE

4.1 Introduction

In this chapter, the toolset to be used for the development of the game in analysed. The reason of choice is justified through it capabilities, ease of use, number of books and documentations available on the tool and its ability to create the game. The functional requirements of the game are discussed with the use of use case and sequence diagrams. The diagrams are used to illustrate the interactions between the player and the game.

4.1.1 Unity 5

The software identified to be used for the development of the game is Unity. Unity is a development platform for creating multiplatform games (Unity3d.com, 2015). It is capable of creating both 3D and 2D games. Unity 5 engine has a vast amount of features, example of some notable ones are;

- Scripting with C#, JavaScript or Boo
- Integrated Animation Editor
- Various graphic effects
- 2D physics
- 3D physics and accurate collision detection
- Access to Web Data through WWW Function
- Hierarchies of mixers
- Multiplayer Networking with RakNet

Both Unity 5 personal edition which is free and professional edition which has a monthly cost, are royalty-free. This means that users do not need to pay royalties or pay revenue share, with games and application created using Unity 5. The free, personal edition of Unity was deemed to be a cost effective approach to build the game for this project.

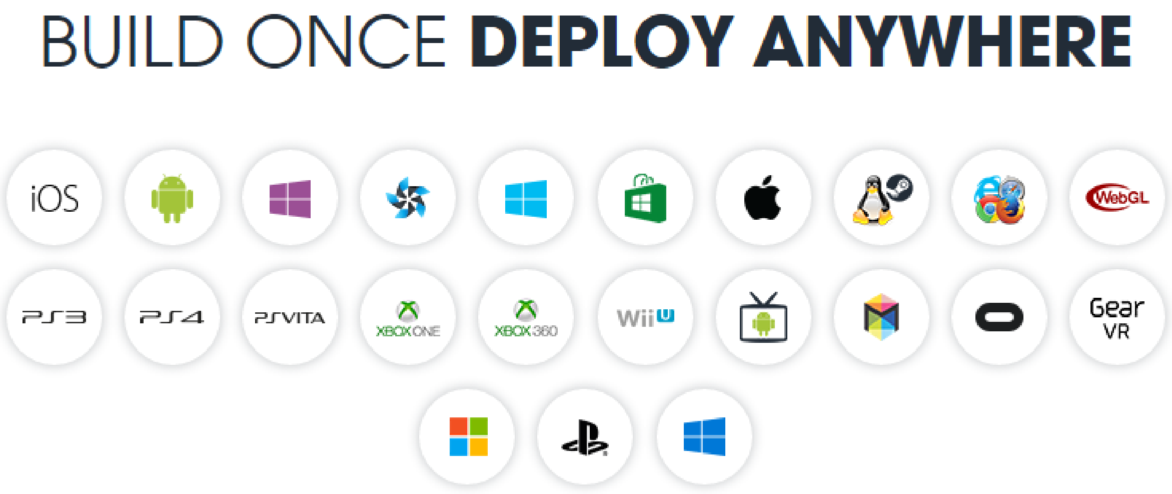


FIGURE 4.1: DEVELOPMENT FOR MULTIPLATFORM GAMES

The number of registered Unity developers has been going at a fast pace but most significantly in the 2014-2015 year. Some of Unity's well known customers include Microsoft, Disney, Coca-Cola, NASA, Electronic Arts and Warner Bros. Unfortunately, this could not be verified as they claim that the report is unreleased by the source.

Unity claims that their game engine is more popular with developers when compared to any other game development software. The source of this data was further investigated and was verified to be accurate. The survey was based on 10,000+ app developers.

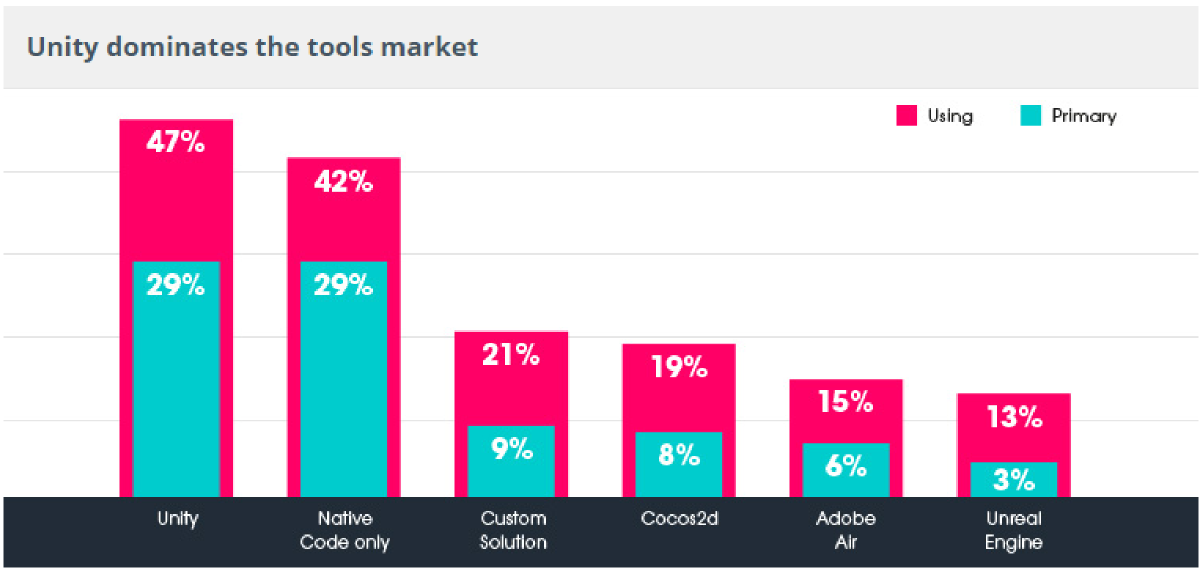


FIGURE 4.2: POPULARITY OF UNITY

There are many books written to teach people how to use it. The Unity website provides video demonstrations and vast amount of information with a manual and scripting API documentation. Unity has been used in many other researches in education. It has been used in investigation on urban design study (Indraprastha *et al.*, 2009). It has even been used in a research to develop a game for Cardiopulmonary resuscitation (CPR) education by applying a gamification theory (Oak *et al.*, 2014).

Its increase of popularity is also observant in it use as the software of choice on game development in recent years. Unity is not simply used for game development alone, instructors are also using it to teach C# programming to students (Norton, 2013). In specific to this project, the ability to create a game to be played on a browser was ideal. This allowed for students on any platform that ran the Firefox browser to play the game.

4.2 Requirements of Analysis

The requirements of the game are covered in this section. The goals of the software are discussed with the use of use case diagrams and sequence diagrams. The functional requirements are stated clearly in this section.

4.2.1 Use Case Definition

A use case describes how users use a system in order to accomplish a specific goal. It acts as a software modelling technique, which helps to define the features to be implemented. It can also help to identify any error that could occur ahead of time.

The interaction between the external actors and the system in order to accomplish a specific goal is defined by the use case. The basic elements that structures a use case are as follows;

- **Actor** - These are type of users that interact with the system
- **System** - Functional requirements which state the intended behaviour of the system.
- **Goals** – The activities and variants involved in obtaining the goal of the user.

4.2.2 Sequence Diagram Definition

In the context of Unified Modelling Language (UML), a sequence diagram represents the collaboration of objects in order to define the sequence of events between objects for a specific goal.

The interaction of objects usually begins at the top of the diagram. The interactions of objects occur through messages. The messages occur in both horizontal and vertical dimensions and are displayed as arrows with names. A lifeline is a rectangular box which has dashed lines going through it and it indicates a role. These rectangle boxes are a representation of instances of objects.

4.2.3 Functional Requirements

- Students must be able to choose to play a New Game.
- Students must provide their details (student number and name) before they play the game.
- System is required to be able to save and display the information provided by students.
- System must display up to date information of the score.
- System must apply the changes in score when students get an answer correct or wrong.
- System is required to be able to save and display students' score and time taken for all levels
- Students must be able to drag all unused puzzle pieces.
- System must ensure only the correct puzzle pieces can drop in the puzzle area and incorrect puzzle pieces are returned back to their original location.
- System must ensure puzzle pieces in the correct puzzle area are no longer drag able.
- System must display a change in colour of the puzzle drop area to provide feedback.

4.2.4 Use Case Diagrams

A use case diagram was developed in order to identify a simplistic view of the functional requirements of this game. The actor is the person playing the game. As the students would be playing the game, the actor is referred to as Player rather than Student. There are five main use cases that define the game and are shown in the figure bellow.

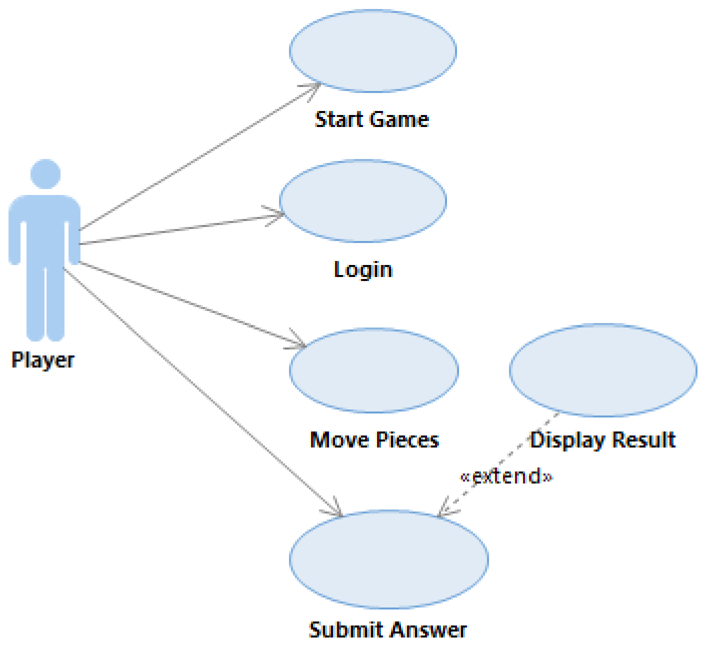


FIGURE 4.3: PLAYER INTERACTION WITH THE SYSTEM

4.2.5 Sequence Diagram

The sequence diagram devised below illustrates the initial counters of the player and the sequence of events. Once the web player has been initialized, it loads the main menu scene. It is in this scene where the player is able to indicate that they wish to play a new game. The ChangeScene classes will run the SceneChange function that will load the scene where students would be required to input student information. The information submitted by the student is verified through the StudentNumber class. Once the information is verified it is saved before changing to the tutorial level scene.

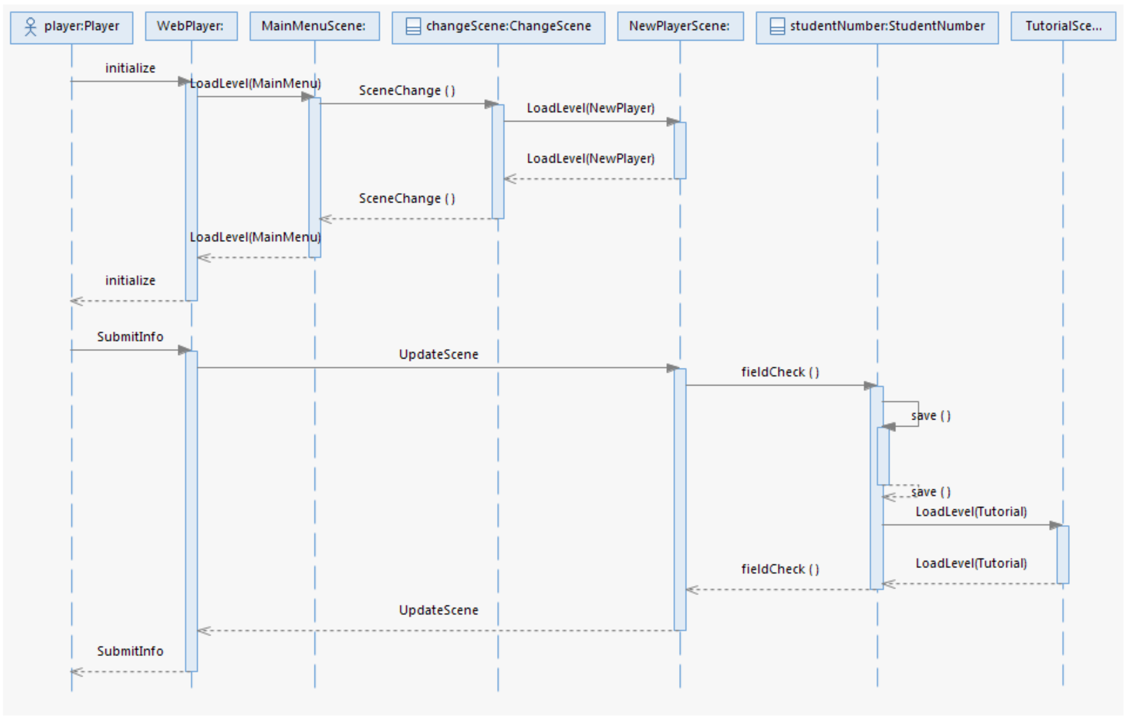


FIGURE 4.4: PROCESS OF LOADING THE TUTORIAL LEVEL

4.2.6 Class Diagram

Class diagrams are a type of diagram that define and provide an overview of the system. This is done in terms of classes, attributes and methods with the inclusion of the relationships between the classes. Development of a class diagram for this particular system is unnecessary and impractical. A total of 73 classes were developed. This does not include the extremely large number of classes used that are provided by Unity 5. The number of classes in total would lay well over 100 classes. Some of this classes interact very differently at different levels. The development chapter of this project, details the interaction of the most important classes with snippets of actual code.

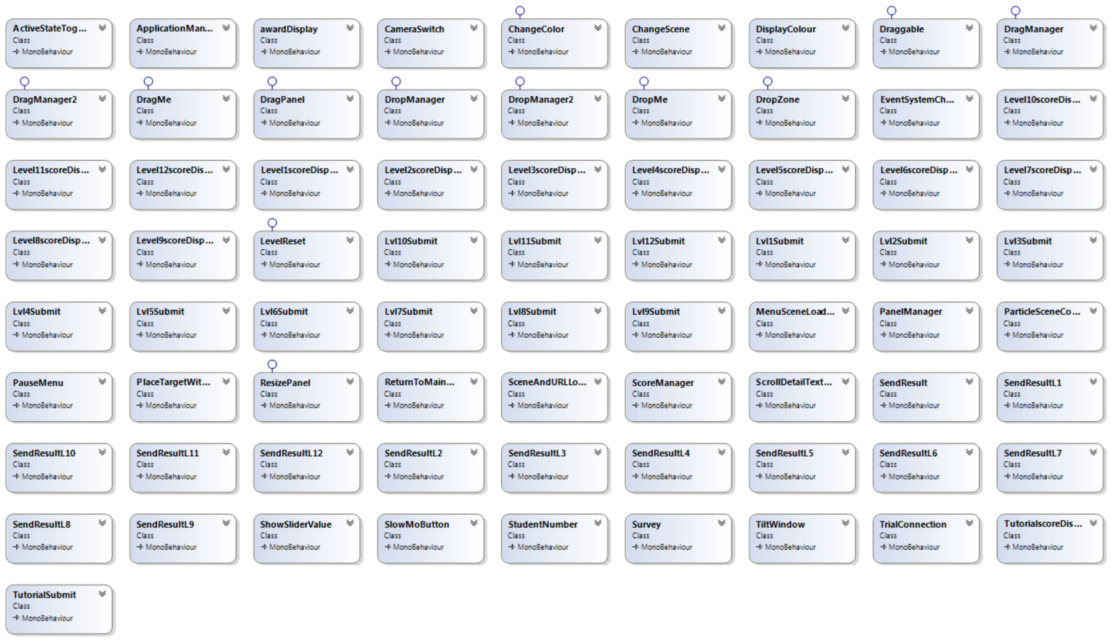


FIGURE 4.5: DIAGRAM OF CLASSES

4.3 Conclusions

This chapter was able to identify the software to be used for the development of the game for assessing students' learning. It justified the reasons of choosing Unity to develop the game with its vast capabilities, use in other researches and its ability to be used to develop the game described in the previous chapter. This chapter identified all the main functional requirements of the game. Diagrams such as use case diagrams and sequence diagrams were used to detail the sequence of events and identify how players would interact with the game.

5 DEVELOPMENT

5.1 Introduction

This chapter discusses the development of the game. The development of the game is based on the design and methodology discussed in previous chapters. The version of Unity used to develop the game was the latest during the development process, this version is Unity 5.2.1f1. This chapter discusses the environment setup of Unity before development could take place. The order of development of the scenes are important as some scenes ware very similar. Development of the game occurred in a sequence of the development of the Main Menu scene, Login scene, Tutorial scene, Tutorial Reward scene and 12 levels and 12 reward scenes.

5.2 Environment Setup

Creating a new project is not difficult to do but there are certain things to be aware of. When opening up Unity, one will be faced with a window shown the figure bellow. Although it may look as simple as providing a project name and clicking “create project”, it is important to take one’s time and understand the options that are available on screen. First of all, it is important to know where the project is going be saved and to change the location of the address if one wishes to. In this case, the unity project was created in a Dropbox folder as a precaution in case if anything unwanted happened to the computer used during this project.

The reason Dropbox was chosen was due to the fact it already existed in the computer being used in this project but also due to the fact that all files related to the project where saved in Dropbox thus keeping everything in one safe location was decided. One important fact to realise is that, Unity allows us to decide whether the project we are creating is going to be 2D or 3D, but be aware of this as, once you chose 2D, you cannot change to a 3D set up. The 3D setup was chosen not simply due to the fact that it allowed us to create in 2D as well and for precaution reasons, but to also keep an open mind to some of the animation that may be used later on in a the game development. We are also given the option to import asset packages. These contain a collection of assets that can be used. It is important to note that, you can decide to import assets after you have created the project. Assets can be downloaded from the same website Unity was downloaded which is Unity’s official website. In our case, all the assets were chosen, this is not recommended but it was found to be a great method to investigating the wide range of images, animation and capabilities of Unity. The most important assets are the Visual Studio 2015 Tools, Medal Icon Kit, Unity Samples UI, and Standard Assets. All these assets are available on the asset store mentioned for free.

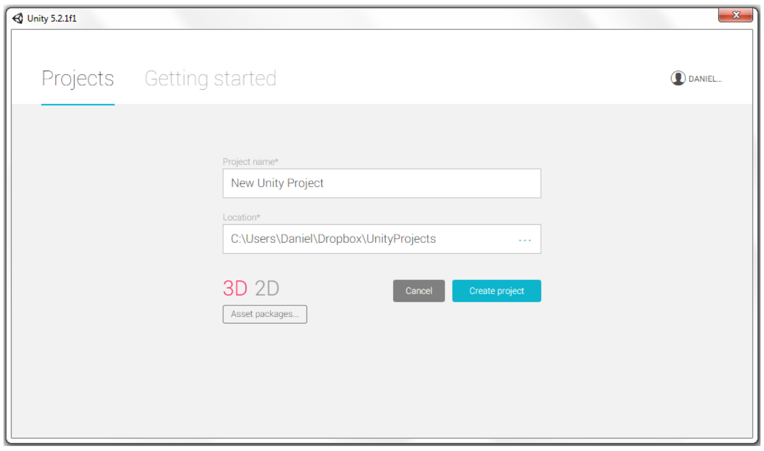


FIGURE 5.1: CREATE NEW PROJECT

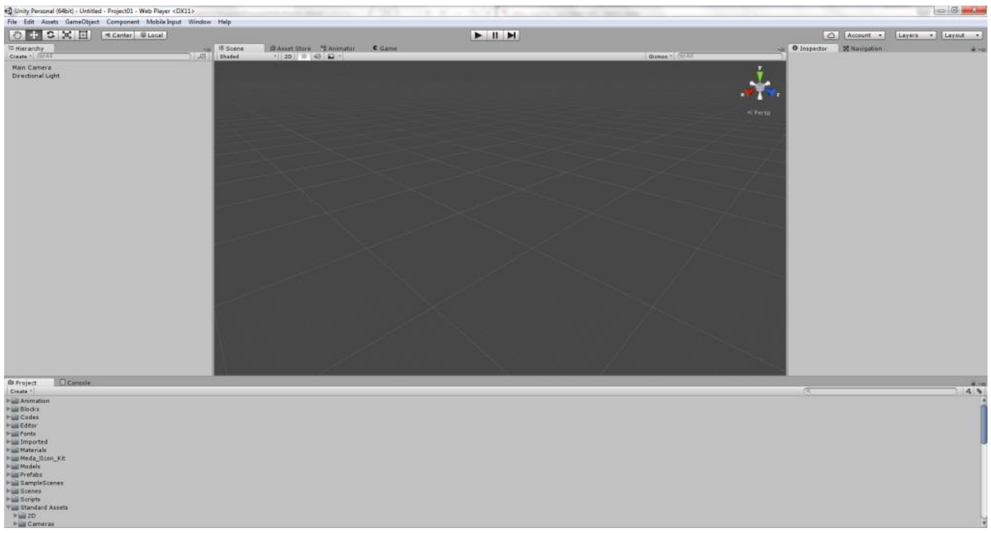


FIGURE 5.2: BLANK SCENE IN PROJECT

Once the project was created, the project automatically presents us with an untitled scene. This figure bellow is a magnified version of the figure above with the 2D view selected. It is important to note that, we are free to change from a 2D vision to a 3D vision and vice-versa. It is important to note that the play button allows you to run the game in the Game window. Any changes made while the play button is activated is reverted once stopped.

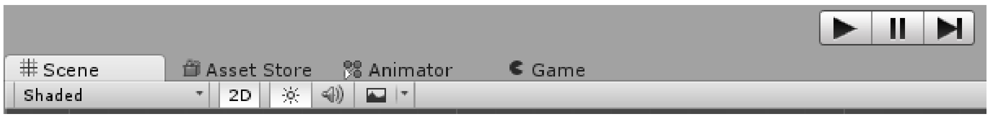


FIGURE 5.3: MAGNIFIED VIEW OF SCENE WINDOW

The background and background particle effects were obtained from our asset "Unity Samples UI". This asset was created by the developers of Unity and is free for Unity developers to use.

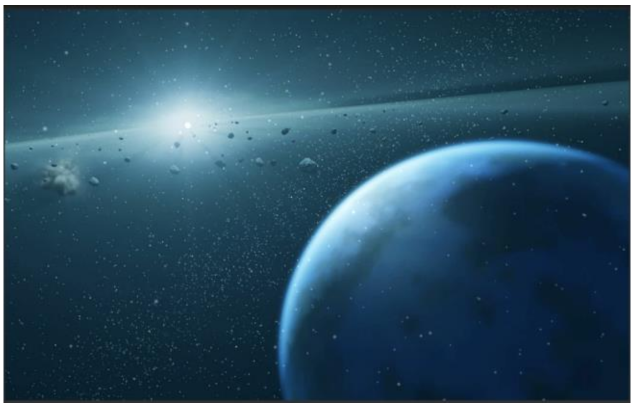


FIGURE 5.4: BACKGROUND AND PARTICLE SYSTEM

The resolution of the game was selected to be 960x600 in the game window. In the drop down menu, this resolution did not exist and was add using the plus symbol in the drop down menu and selected the resolution that was desired.

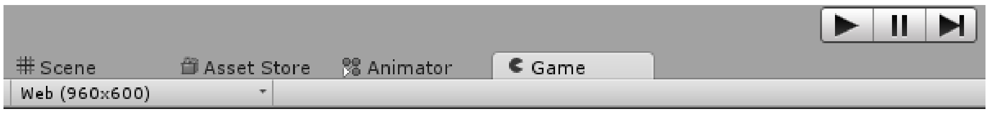


FIGURE 5.5: MAGNIFIED VIEW OF GAME WINDOW

5.3 Development of Main Menu Scene

In the hierarchy of this scene, our Main Camera, Directional Light and EventSystem were automatically created by Unity. The "SF Scene Elements" is the background and background effects, already discussed at the beginning of this chapter. The prefab was located in the project window and it was simply dragged and dropped. A canvas was created by right clicking in the hierarchy window, in the pop up UI was clicked and then canvas was clicked. The canvas is a game object where the UI elements will be inside of. The canvas has a render mode setting, which was used to me it render the UI, this was done through the inspector window.

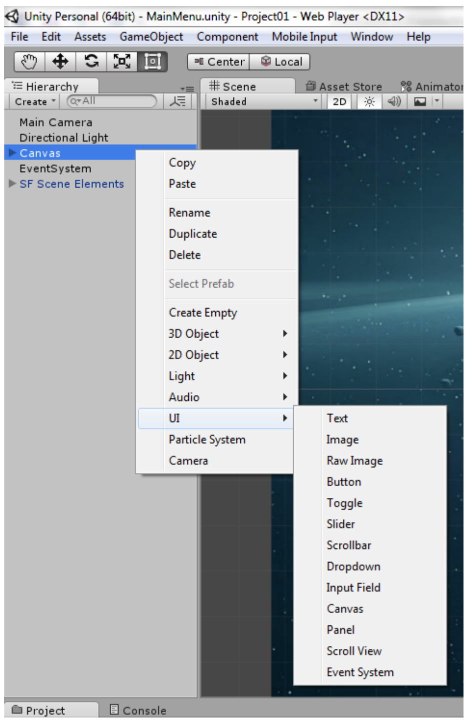
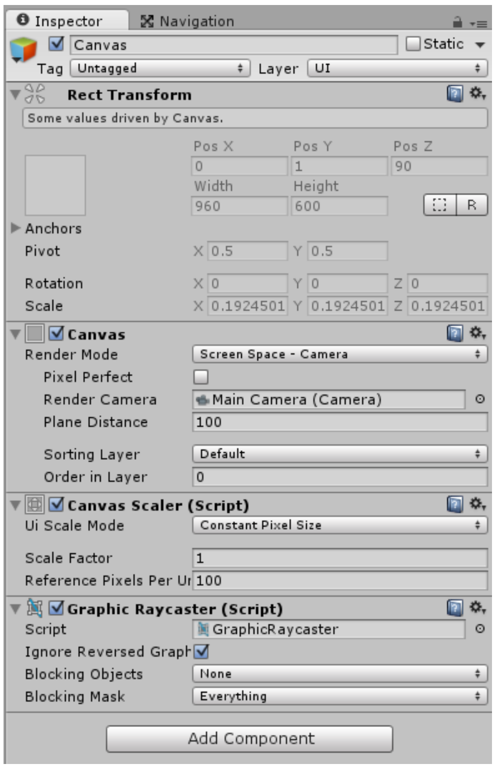


FIGURE 5.6: ADDING GAME OBJECTS IN THE HIERARCHY



Canvas Static
Tag Untagged Layer UI

Rect Transform
Some values driven by Canvas.
Pos X 0 Pos Y 1 Pos Z 90
Width 960 Height 600
Anchors
Pivot X 0.5 Y 0.5
Rotation X 0 Y 0 Z 0
Scale X 0.1924501 Y 0.1924501 Z 0.1924501

Canvas
Render Mode Screen Space - Camera
Pixel Perfect
Render Camera Main Camera (Camera)
Plane Distance 100
Sorting Layer Default
Order in Layer 0

Canvas Scaler (Script)
Ui Scale Mode Constant Pixel Size
Scale Factor 1
Reference Pixels Per Ur 100

Graphic Raycaster (Script)
Script GraphicRaycaster
Ignore Reversed Graph ✓
Blocking Objects None
Blocking Mask Everything

FIGURE 5.7: INSPECTOR ON THE CANVAS

The game objects shown in the figure bellow show all the game objects used in the "MainMenu" scene. All the game objects in the canvas were UI panels except for "Continue", "New Game", "Settings" and "Quit", were all UI buttons.

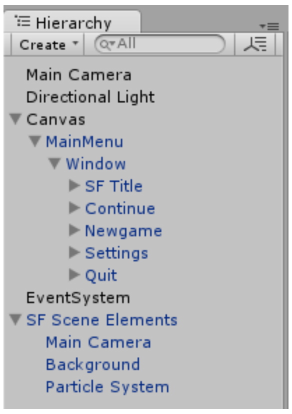


FIGURE 5.8: ALL OBJECTS USED SHOWN IN THE HIERARCHY

The figure bellow shows all the game objects used. The inspector window was used to change the fonts, colour and size of the UI elements.

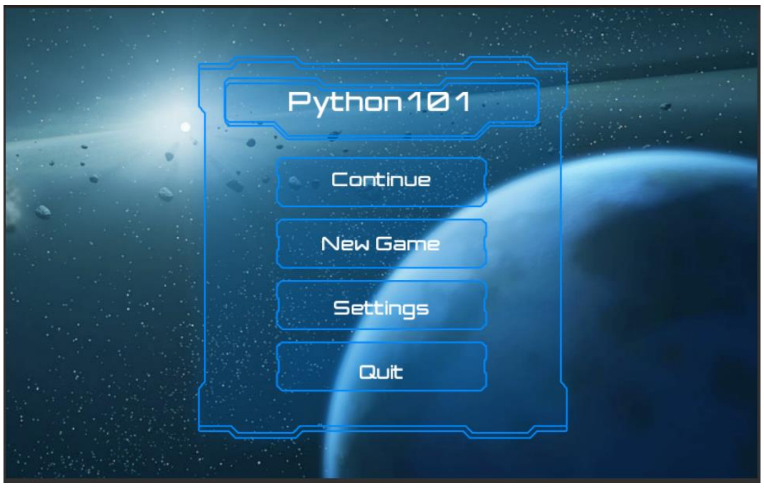


FIGURE 5.9: MENU SCENE

A new folder was created in the project window, named "Codes" and by right clicking on the folder, clicking on Create and then clicking on C# Script. All scripts created in this project, were created in this manner and this process will be assumed when mentioning all scripts created. In this case, the script created was named ChangeScene.

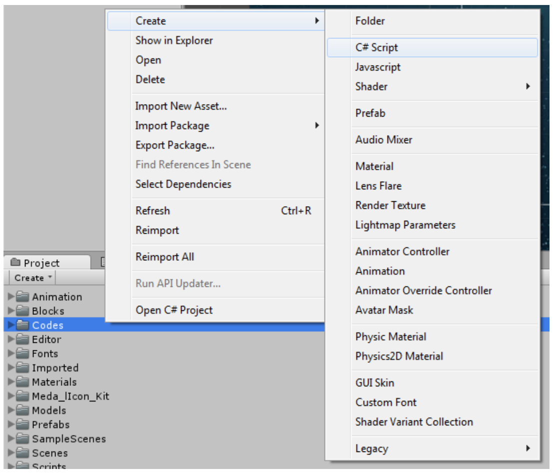
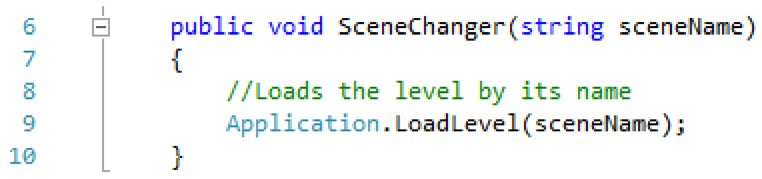


FIGURE 5.10: ADDING NEW SCRIPTS IN PROJECT WINDOW

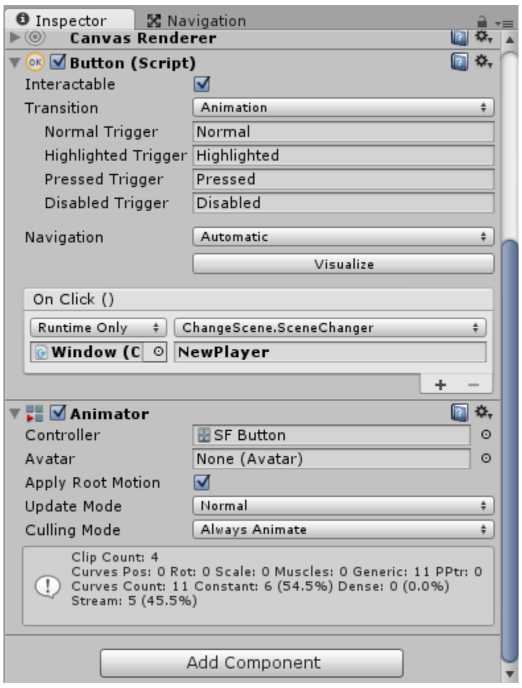
The script loads a level using its name. There two important things to note here, when loading a new scene, all the current game object are destroyed. Second is that before a scene can be loaded, it must first exist in the list of levels or scenes. This is done through File->Build Setting.



```
public void SceneChanger(string sceneName)
{
    //Loads the level by its name
    Application.LoadLevel(sceneName);
}
```

FIGURE 5.11: SCENECHANGER FUNCTION FROM THE CHANGESCENE CLASS

The UI panel "Window" was selected and a script was added to it using the add component button found in the inspector window. The add component button allows us to add scripts by simply searching for their names. This is the step used to add scripts to game objects, thus when mentioning a scripts that were added to other game object, the process remains the same. In this chase specifically the ChangeScene script was add. The UI button "New Game" was selected in the hierarchy and in its inspector the "On Click ()" function was used to call a function from the ChangeScene C# script. "NewPlayer" is the scene name of the scene to be loaded when the New Player button has been clicked.


Inspector | Navigation

Canvas Renderer

Button (Script)

Interactable ✓

Transition: Animation

Normal Trigger: Normal

Highlighted Trigger: Highlighted

Pressed Trigger: Pressed

Disabled Trigger: Disabled

Navigation: Automatic

Visualize

On Click ()

Runtime Only | ChangeScene.SceneChanger

Window (C | NewPlayer

Animator

Controller: SF Button

Avatar: None (Avatar)

Apply Root Motion ✓

Update Mode: Normal

Culling Mode: Always Animate

```
Clip Count: 4
Curves Pos: 0 Rot: 0 Scale: 0 Muscles: 0 Generic: 11 PPtr: 0
Curves Count: 11 Constant: 6 (54.5%) Dense: 0 (0.0%)
Stream: 5 (45.5%)
```

Add Component

FIGURE 5.12: INSPECTOR ON “NEW GAME” BUTTON

5.4 *Development of Login Scene*

A new Scene is developed named “NewPlayer”. This scene is the login scene were the game obtains the player’s student number and full name. The hierarchy window bellow displays all the game object used to develop this scene.

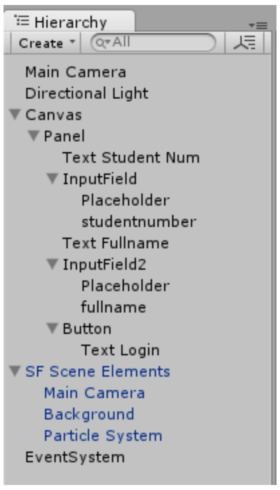

```
Hierarchy
Create ▾  All

Main Camera
Directional Light
▼ Canvas
  ▼ Panel
      Text Student Num
    ▼ InputField
        Placeholder
        studentnumber
      Text Fullname
    ▼ InputField2
        Placeholder
        fullname
    ▼ Button
        Text Login
▼ SF Scene Elements
    Main Camera
    Background
    Particle System
EventSystem
```

FIGURE 5.13: HIERARCHY OF “NEWPLAYER” SCENE

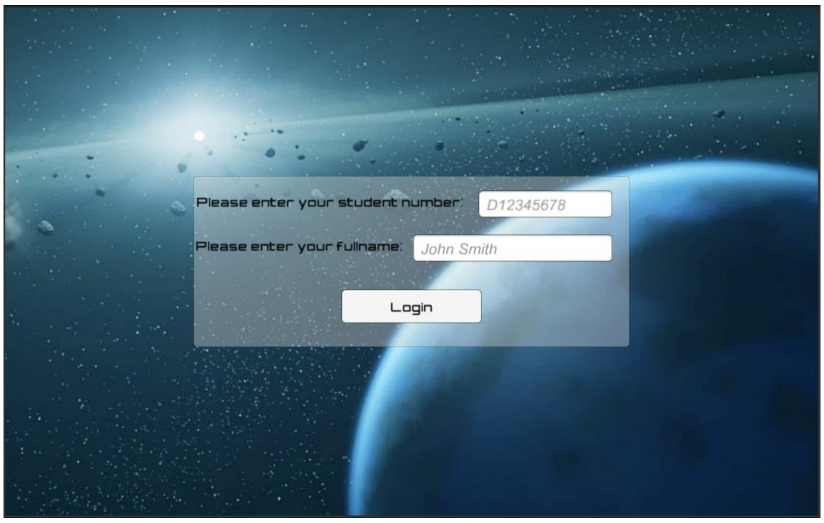
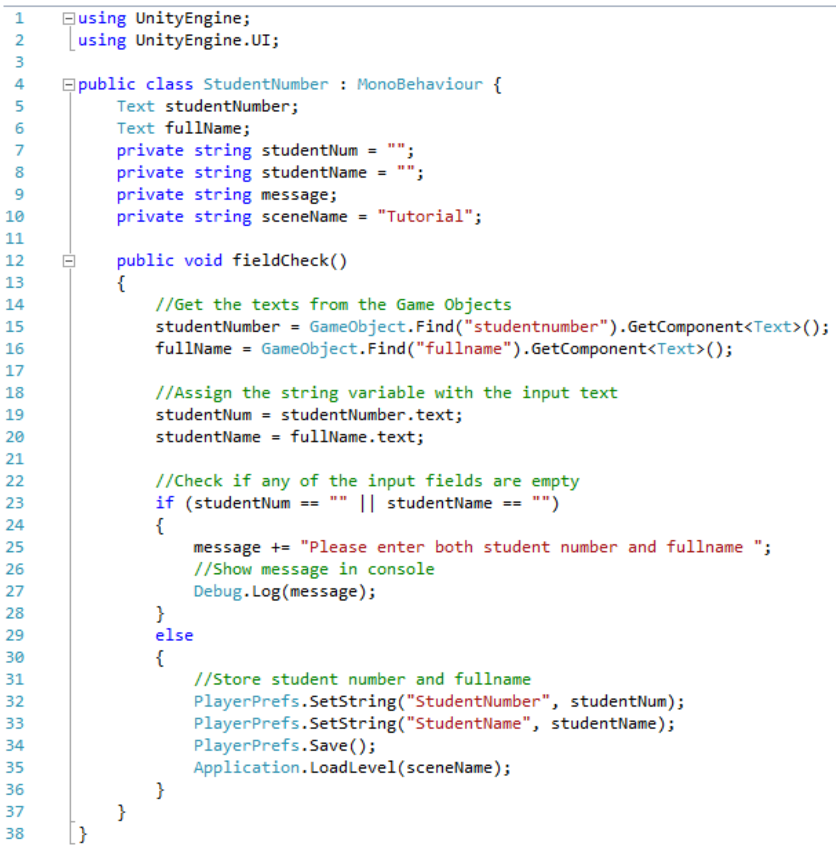

Please enter your student number:
D12345678
Please enter your fullname:
John Smith
Login

FIGURE 5.14: LOGIN SCENE

The class StudentNumber was attached to the "Button" game object, which is the login button. The class is set to run "On Click ()" of the button.



```
using UnityEngine;
using UnityEngine.UI;

public class StudentNumber : MonoBehaviour {
    Text studentNumber;
    Text fullName;
    private string studentNum = "";
    private string studentName = "";
    private string message;
    private string sceneName = "Tutorial";

    public void fieldCheck()
    {
        //Get the texts from the Game Objects
        studentNumber = GameObject.Find("studentnumber").GetComponent<Text>();
        fullName = GameObject.Find("fullname").GetComponent<Text>();

        //Assign the string variable with the input text
        studentNum = studentNumber.text;
        studentName = fullName.text;

        //Check if any of the input fields are empty
        if (studentNum == "" || studentName == "")
        {
            message += "Please enter both student number and fullname ";
            //Show message in console
            Debug.Log(message);
        }
        else
        {
            //Store student number and fullname
            PlayerPrefs.SetString("StudentNumber", studentNum);
            PlayerPrefs.SetString("StudentName", studentName);
            PlayerPrefs.Save();
            Application.LoadLevel(sceneName);
        }
    }
}
```

FIGURE 5.15: STUDENTNUMBER CLASS

PlayerPrefs is a class in UnityEngine. It is used to store and access player preferences at different scenes. When loading the new scene, all current Game Object would be destroyed but thanks to PlayersPrefs, we can store the student number and full name. This information can be access at a different scene when required. On Web players specifically, PlayerPrefs are located in:

Mac OS X: ~/Library/Preferences/Unity/WebPlayerPrefs

Windows: %APPDATA%\Unity\WebPlayerPrefs

They are stored in binary files. There are limitation as, only one preference file per Web player URL is possible and also there is a 1 megabyte file size limit. The table below are the static functions of the PlayerPrefs class. This table is a direct quotation from the documentation obtained by Unity.

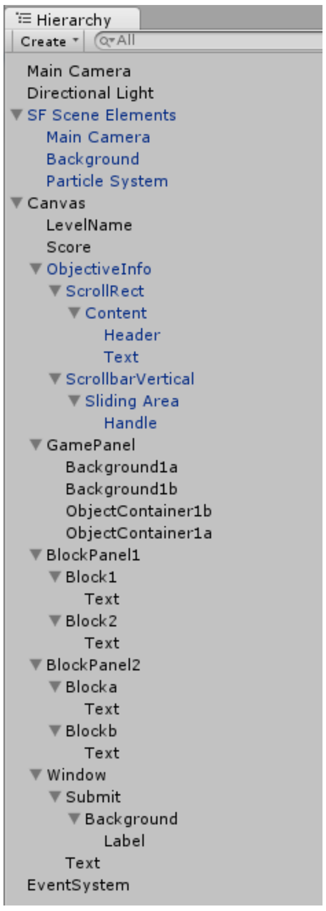
DeleteAll	Removes all keys and values from the preferences. Use with caution.
DeleteKey	Removes key and its corresponding value from the preferences.
GetFloat	Returns the value corresponding to key in the preference file if it exists.
GetInt	Returns the value corresponding to key in the preference file if it exists.
GetString	Returns the value corresponding to key in the preference file if it exists.
HasKey	Returns true if key exists in the preferences.
Save	Writes all modified preferences to disk.
SetFloat	Sets the value of the preference identified by key.
SetInt	Sets the value of the preference identified by key.
SetString	Sets the value of the preference identified by key.

TABLE 5.1: FUNCTIONS OF THE PLAYERPREFS CLASS

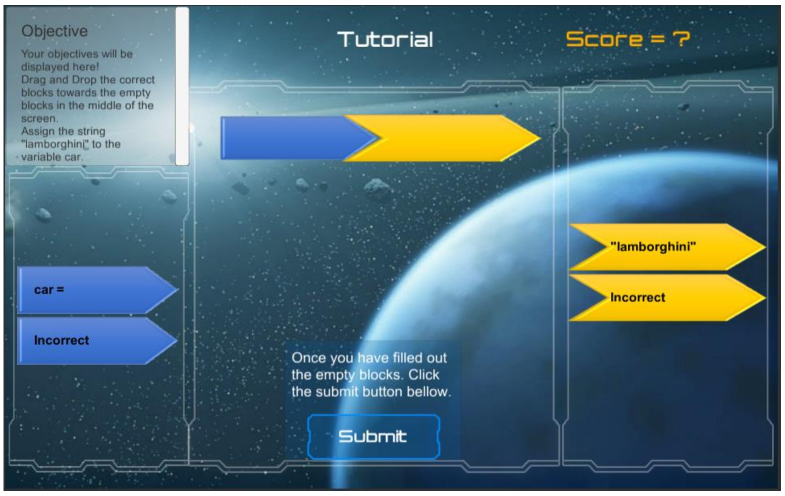
The figure below shows the game view of the tutorial level. This scene loads directly after the student has successfully logged in. The purpose of this scene is for the student to get use to the game mechanics, rules and simply to learn how to play the game. The score are displayed in this level but are not recorded as students were encouraged to make a mistake on purpose so that they can see the outcome. This outcome was a deduction of their score and a red colour mark on the area they got incorrect.

5.5 *Development of Tutorial Level*

The hierarchy is displayed in the figure below to illustrate all the game objects required to develop this level. The "SF Scene Elements" remains unchanged to that of previous scene. The "SF Scene Elements" prefab was simply dragged and dropped from the project window.



```
Main Camera
Directional Light
▼SF Scene Elements
   Main Camera
   Background
   Particle System
▼Canvas
   LevelName
   Score
  ▼ObjectiveInfo
    ▼ScrollRect
      ▼Content
         Header
         Text
    ▼ScrollbarVertical
      ▼Sliding Area
         Handle
  ▼GamePanel
     Background1a
     Background1b
     ObjectContainer1b
     ObjectContainer1a
  ▼BlockPanel1
    ▼Block1
       Text
    ▼Block2
       Text
  ▼BlockPanel2
    ▼Blocka
       Text
    ▼Blockb
       Text
  ▼Window
    ▼Submit
      ▼Background
         Label
     Text
EventSystem
```

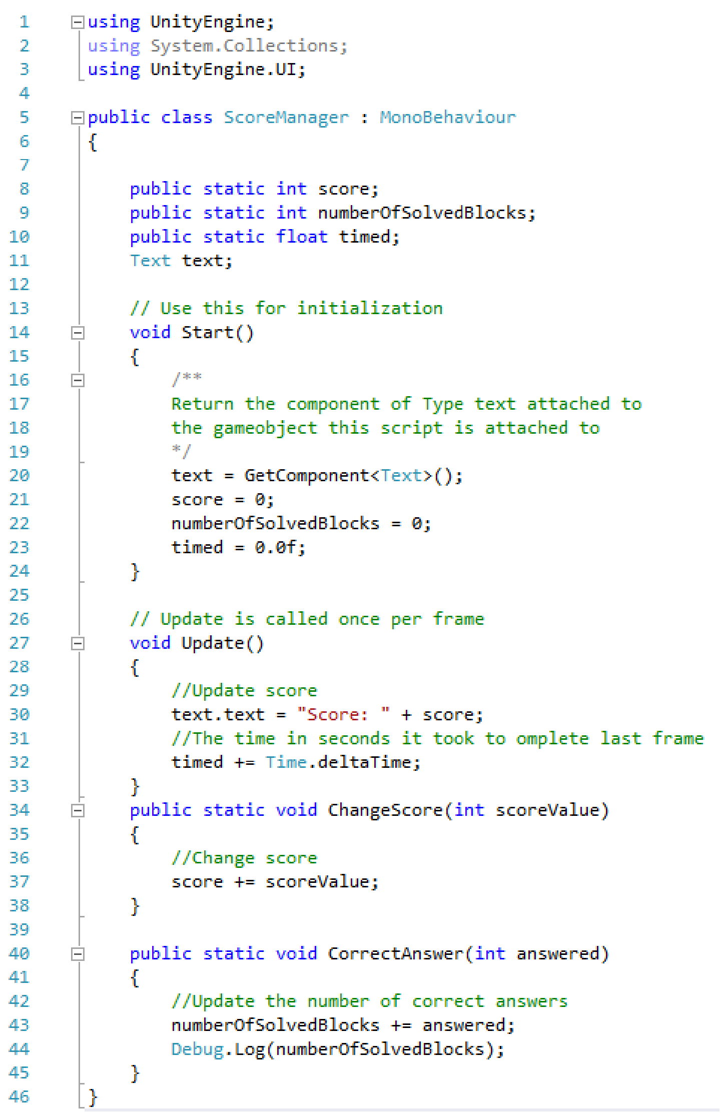


Your objectives will be displayed here!
Drag and Drop the correct blocks towards the empty blocks in the middle of the screen.
Assign the string "lamborghini" to the variable car.

Once you have filled out the empty blocks. Click the submit button bellow.

FIGURE 5.16: TUTORIAL LEVEL AND HIERARCHY WINDOW

A game object canvas called "Canvas" was created first. In the canvas, a UI text game objected named "LevelName" was created. In the inspector of "LevelName", "Tutorial" was written in the text container. The Font was assigned to be "Jupitor" with a normal font style of normal, line spacing of 1 and a font size of 45. The colour of the text was set to be white in order to make it stand out from the background. The UI text game object "Score" was created similar to that of "LevelName" game object. The "Score" object was located beside the "LevelName" game object with an orange colour but the font, font style, font size and line spacing remained the same as that of "LevelName". The "Score" object contained the script "ScoreManager".



```
using UnityEngine;
using System.Collections;
using UnityEngine.UI;

public class ScoreManager : MonoBehaviour
{

    public static int score;
    public static int numberOfSolvedBlocks;
    public static float timed;
    Text text;

    // Use this for initialization
    void Start()
    {
        /**
        Return the component of Type text attached to
        the gameobject this script is attached to
        */
        text = GetComponent<Text>();
        score = 0;
        numberOfSolvedBlocks = 0;
        timed = 0.0f;
    }

    // Update is called once per frame
    void Update()
    {
        //Update score
        text.text = "Score: " + score;
        //The time in seconds it took to omplete last frame
        timed += Time.deltaTime;
    }
    public static void ChangeScore(int scoreValue)
    {
        //Change score
        score += scoreValue;
    }

    public static void CorrectAnswer(int answered)
    {
        //Update the number of correct answers
        numberOfSolvedBlocks += answered;
        Debug.Log(numberOfSolvedBlocks);
    }
}
```

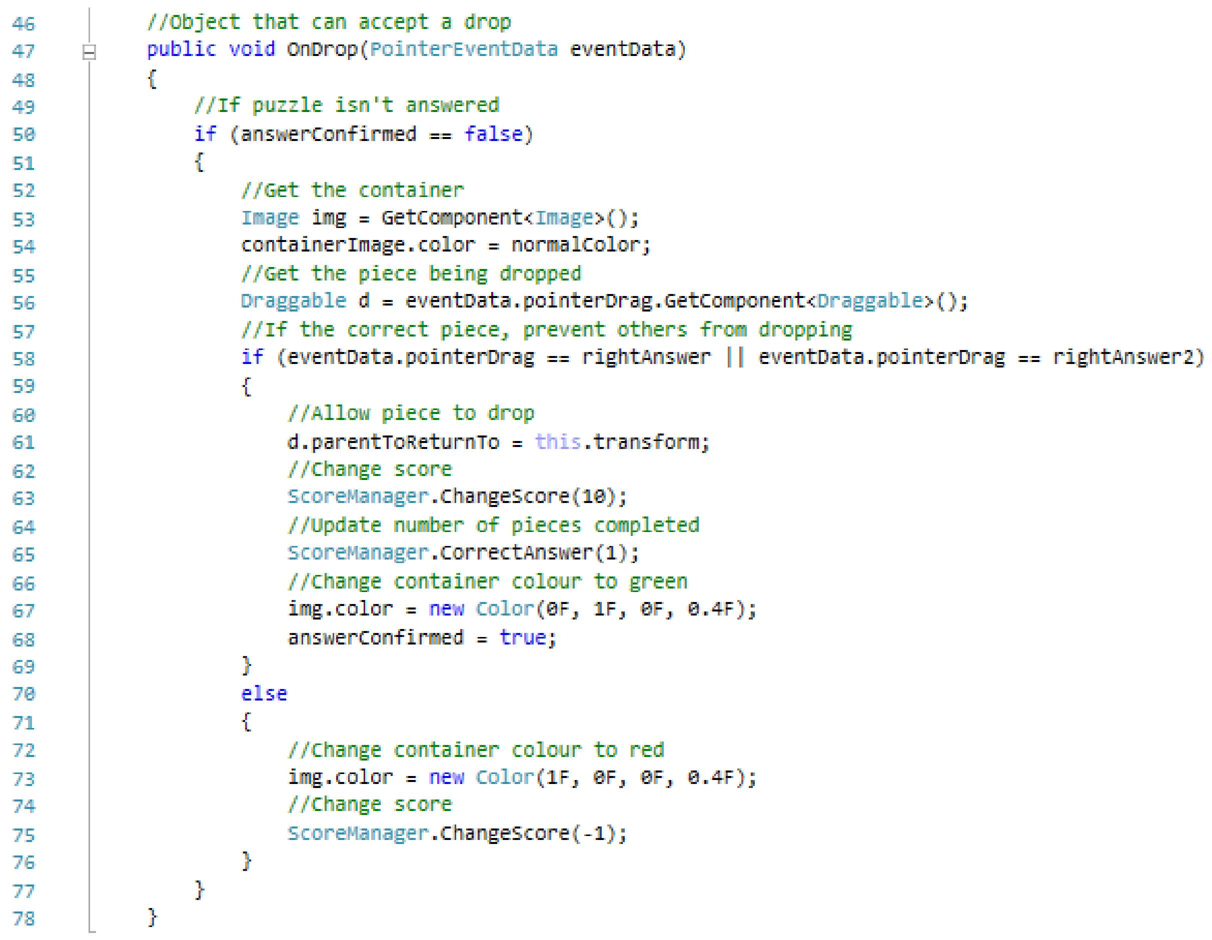
FIGURE 5.17: SCOREMANAGER CLASS

"ObjectiveInfo" is the top right object in the game which provides the player with the objectives of the level. This was created through the UI scroll view game object. The "Scrollbar Horizontal" game object and all of the object it contained were deleted. In the “Content” object two UI text game objects where created, these were “Header” and “text”. The “Header” contained the text “Objective” while the “text” contained details of the objective.

“GamePanel” is a UI panel game object placed in the middle of the scene. It is in this panel where the player drops the puzzle pieces to make a working code. In this panel, two types of game object are located. The first are “Background1a” and “Background1b”, these are images of the blue and yellow blocks. The second are “ObjectContainer1b” and “ObjectContainer1a”, these are panel used to allow the player to drop blocks and also to indicate with colour if the dropped item was correct. It is very important to have the “ObjectContainer1b” to be ahead of “ObjectContainer1a” in the hierarchy, due to the shape of the objects being dropped causing an overlay of the colour used to indicate the answers correctness.

These containers have a component named Grid Layout Group. The Grid Layout Group specifies the cell size to be 200 by 60 for the blue block being dropped and 250 by 60 for the yellow block to be dropped. It was used to also set the start corner to be upper left, start axis to be horizontal, Child Alignment to be middle left and constraint to be flexible. This is important has the dropped block piece will be a child of this panel and we are specifying that the blocked being dropped will have a cell size of 200 by 60 or 250 by 60 and its alignment is middle left. The script “DropManager2” is attached to all object containers.

The IDropHandler is an interface implemented to receive OnDrop callbacks. OnDrop is called by a BaseInputModule on an object that can accept a drop. The DropManager2 class was developed in order for it to be reused at other levels. The inspector window were the script was added is used to specify which blocks can drop.



```
//Object that can accept a drop
public void OnDrop(PointerEventData eventData)
{
    //If puzzle isn't answered
    if (answerConfirmed == false)
    {
        //Get the container
        Image img = GetComponent<Image>();
        containerImage.color = normalColor;
        //Get the piece being dropped
        Draggable d = eventData.pointerDrag.GetComponent<Draggable>();
        //If the correct piece, prevent others from dropping
        if (eventData.pointerDrag == rightAnswer || eventData.pointerDrag == rightAnswer2)
        {
            //Allow piece to drop
            d.parentToReturnTo = this.transform;
            //Change score
            ScoreManager.ChangeScore(10);
            //Update number of pieces completed
            ScoreManager.CorrectAnswer(1);
            //Change container colour to green
            img.color = new Color(0F, 1F, 0F, 0.4F);
            answerConfirmed = true;
        }
        else
        {
            //Change container colour to red
            img.color = new Color(1F, 0F, 0F, 0.4F);
            //Change score
            ScoreManager.ChangeScore(-1);
        }
    }
}
```

FIGURE 5.18: ONDROP FUNCTION IN DROPMANAGER CLASS

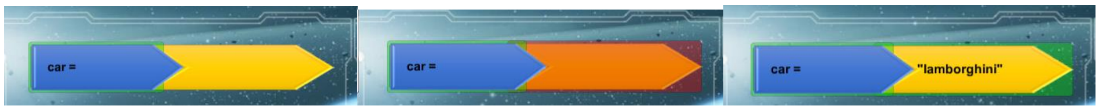
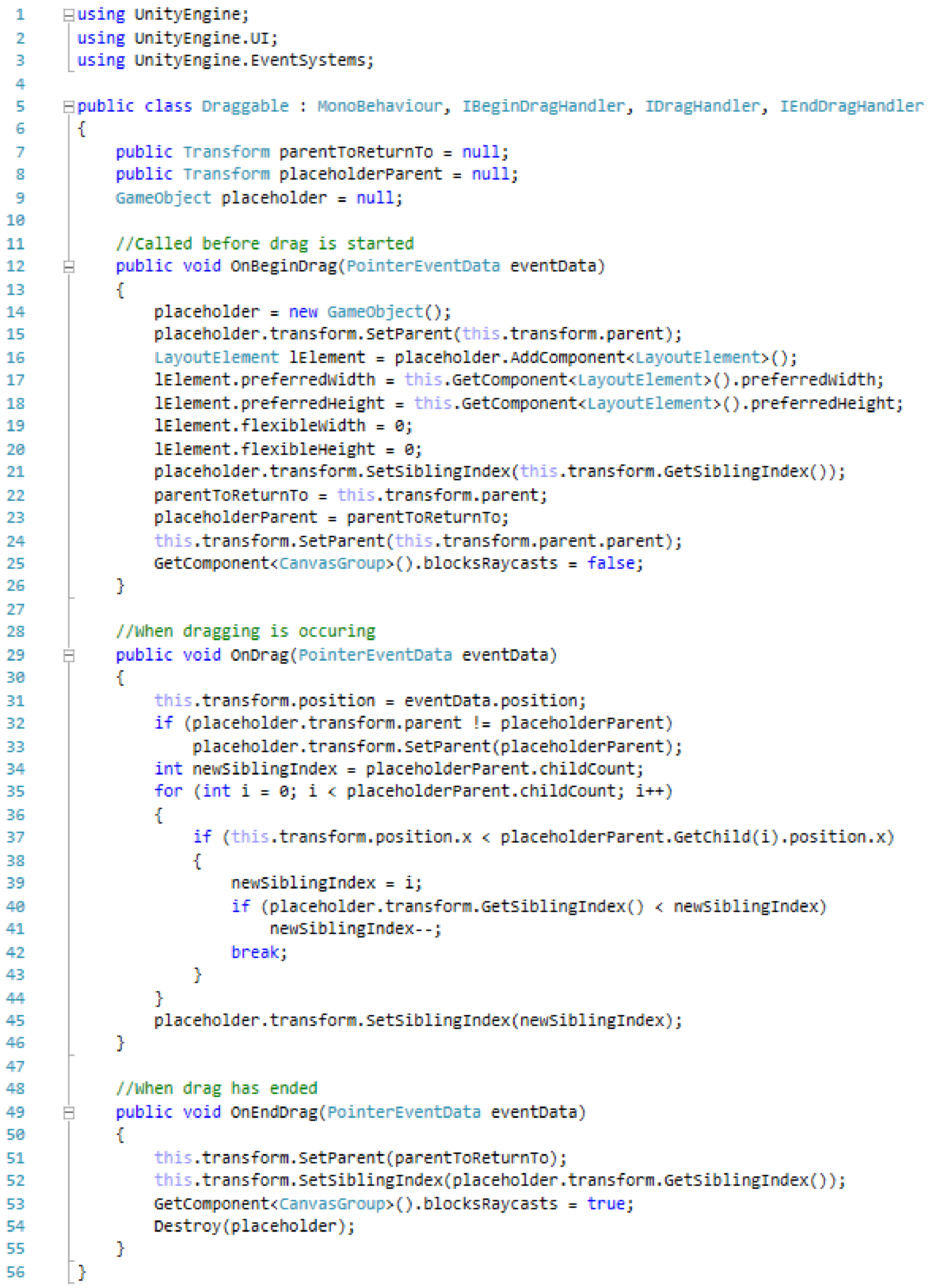


FIGURE 5.19: CHANGE IN BACKGROUND

On the left hand side a UI panel named “BlockPanel1” contains the blue puzzle pieces. This panel contains a Vertical Layout Group component which specifies the spacing to be 3, the child alignment to be middle centre and that the child should not be forced to expand in width or height. This is used to arrange the alignment of the game objects “Block1” and “Block2”. These blocks are UI images. Their source image is the blue block with an image type of simple. The Canvas Group component was assigned to specify that the game object was intractable and that it blocked raycasts.

A Layout Element component was assigned to it to specify that there is a preferred width of 200, height of 60, flexible width of 0 and flexible height of 0. This insures that the element does not change in size when being dragged. The Horizontal Layout Group component was assigned to these objects to indicate that the child object “Text” which is a text UI has a padding of 21 from the left side and that its alignment is in the middle left with indication that it should not expand. The text contained a piece of a code for the game. The script “Draggable” was added to each block. The panel on the left hand side that contained the yellow puzzle pieces “BlockPanel2” was developed the same way as that of the panel on the left hand side “BlockPanel1”. The only major differences between them is the preferred width of the blocks being 250 rather than 200.

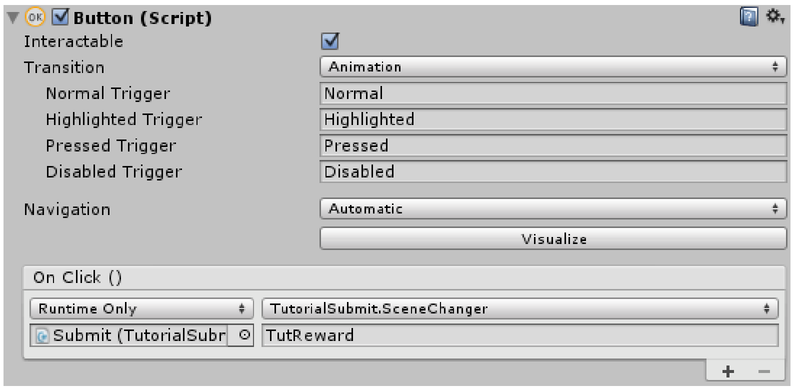


```
using UnityEngine;
using UnityEngine.UI;
using UnityEngine.EventSystems;

public class Draggable : MonoBehaviour, IBeginDragHandler, IDragHandler, IEndDragHandler
{
    public Transform parentToReturnTo = null;
    public Transform placeholderParent = null;
    GameObject placeholder = null;

    //Called before drag is started
    public void OnBeginDrag(PointerEventData eventData)
    {
        placeholder = new GameObject();
        placeholder.transform.SetParent(this.transform.parent);
        LayoutElement lElement = placeholder.AddComponent<LayoutElement>();
        lElement.preferredWidth = this.GetComponent<LayoutElement>().preferredWidth;
        lElement.preferredHeight = this.GetComponent<LayoutElement>().preferredHeight;
        lElement.flexibleWidth = 0;
        lElement.flexibleHeight = 0;
        placeholder.transform.SetSiblingIndex(this.transform.GetSiblingIndex());
        parentToReturnTo = this.transform.parent;
        placeholderParent = parentToReturnTo;
        this.transform.SetParent(this.transform.parent.parent);
        GetComponent<CanvasGroup>().blocksRaycasts = false;
    }

    //When dragging is occuring
    public void OnDrag(PointerEventData eventData)
    {
        this.transform.position = eventData.position;
        if (placeholder.transform.parent != placeholderParent)
            placeholder.transform.SetParent(placeholderParent);
        int newSiblingIndex = placeholderParent.childCount;
        for (int i = 0; i < placeholderParent.childCount; i++)
        {
            if (this.transform.position.x < placeholderParent.GetChild(i).position.x)
            {
                newSiblingIndex = i;
                if (placeholder.transform.GetSiblingIndex() < newSiblingIndex)
                    newSiblingIndex--;
                break;
            }
        }
        placeholder.transform.SetSiblingIndex(newSiblingIndex);
    }

    //When drag has ended
    public void OnEndDrag(PointerEventData eventData)
    {
        this.transform.SetParent(parentToReturnTo);
        this.transform.SetSiblingIndex(placeholder.transform.GetSiblingIndex());
        GetComponent<CanvasGroup>().blocksRaycasts = true;
        Destroy(placeholder);
    }
}
```

FIGURE 5.20: DRAGGABLE CLASS

The IBeginDragHandler is an interface implemented to receive OnBeginDrag callbacks. OnBeginDrag is called by a BaseInputModule before a drag is started. The placeholder for the puzzle piece is specified. This is important later on as, if they object doesn't get dropped in the correct location, it should return back to the parent. The IDragHandler is an interface implemented to receive OnDrag callbacks. Then dragging is occurring OnDrag will be called every time the cursor is moved. For this particular game we indicate the position of the block to be at the cursor so that the players can see the piece they are dragging, so that they can guide it to the drop area. The IEndDragHandler is an interface implemented to receive OnEndDrag callbacks. OnEndDrag is called when a drag has ended.

In the middle bottom, the UI panel named “Window” exists. It contains two objects, one is “Text” and the other is “Submit”. “Text” is an UI text game object that describes what the submit button does. “Submit” is a UI button which also contains an image UI “Background” with a text UI “Label”. The “Submit” game object has an animation component attached to it. AS the “Submit” game object is a UI button, in the inspector we can specify what it does when triggered. The figure bellow shows that the function sceneChanger is called from the TutorialSubmit class. A string variable “TutReward” is sent to that function. This is indicating to the scene changer that the next scene we want if possible is the “TutReward” scene. The script TutorialSubmit is also attached to the “Submit” game object.

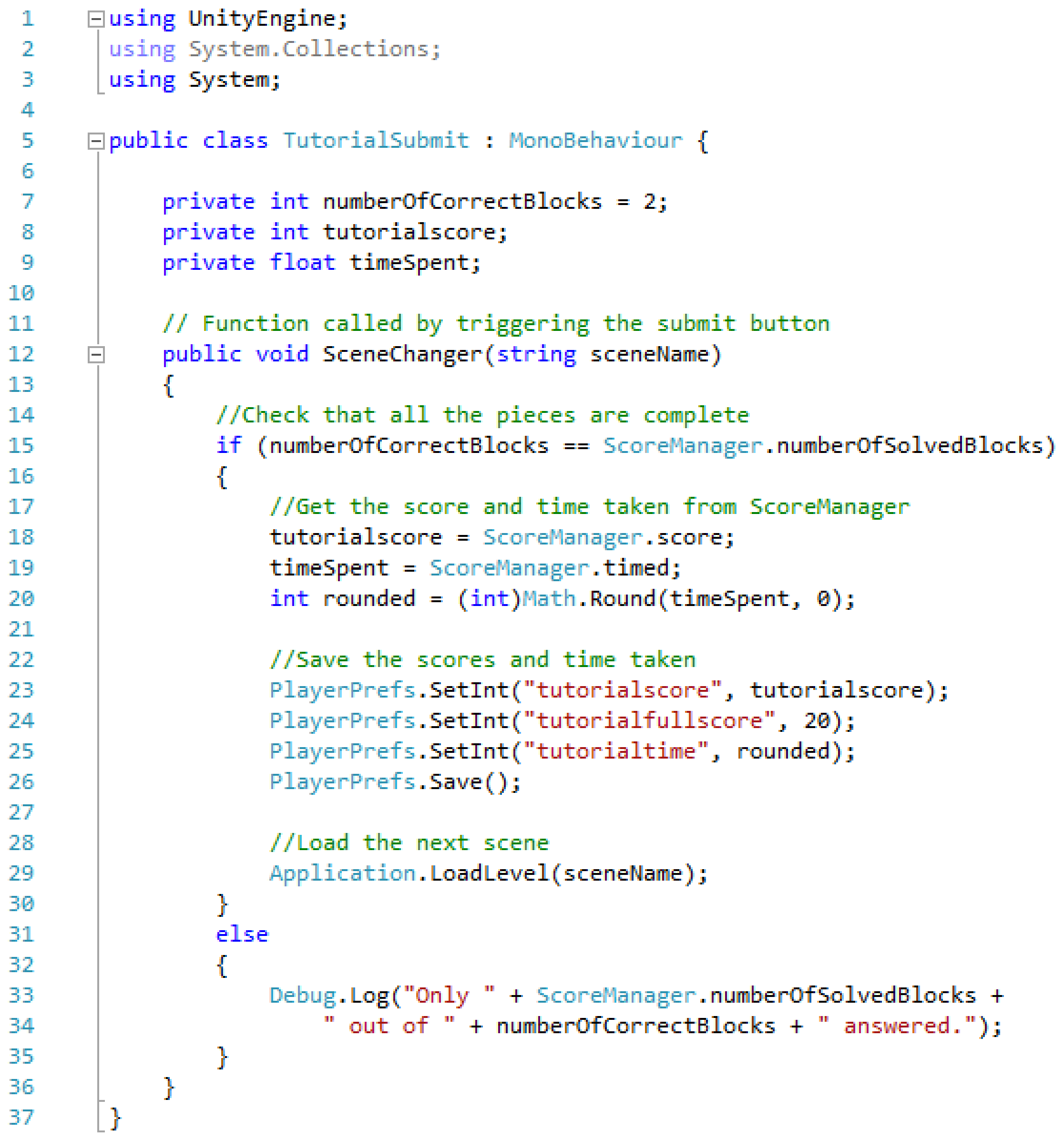

Button (Script)

Interactable	✓
Transition	Animation
Normal Trigger	Normal
Highlighted Trigger	Highlighted
Pressed Trigger	Pressed
Disabled Trigger	Disabled
Navigation	Automatic

On Click ()

Runtime Only	TutorialSubmit.SceneChanger
Submit (TutorialSubr	TutReward

FIGURE 5.21: SUBMIT BUTTON



```
using UnityEngine;
using System.Collections;
using System;

public class TutorialSubmit : MonoBehaviour {

    private int numberOfCorrectBlocks = 2;
    private int tutorialscore;
    private float timeSpent;

    // Function called by triggering the submit button
    public void SceneChanger(string sceneName)
    {
        //Check that all the pieces are complete
        if (numberOfCorrectBlocks == ScoreManager.numberOfSolvedBlocks)
        {
            //Get the score and time taken from ScoreManager
            tutorialscore = ScoreManager.score;
            timeSpent = ScoreManager.timed;
            int rounded = (int)Math.Round(timeSpent, 0);

            //Save the scores and time taken
            PlayerPrefs.SetInt("tutorialscore", tutorialscore);
            PlayerPrefs.SetInt("tutorialfullscore", 20);
            PlayerPrefs.SetInt("tutorialtime", rounded);
            PlayerPrefs.Save();

            //Load the next scene
            Application.LoadLevel(sceneName);
        }
        else
        {
            Debug.Log("Only " + ScoreManager.numberOfSolvedBlocks +
                " out of " + numberOfCorrectBlocks + " answered.");
        }
    }
}
```

FIGURE 5.22: TUTORIALSUBMIT CLASS

5.6 Development of Tutorial Reward Scene

The figure below shows the “TutReward” scene, which is loaded once the student has successfully completed the tutorial level. This scene indicates to the player that they have completed the tutorial level and provides them with a digital badge for their achievement. It also indicates to them, there Score and the time it took them to complete the level.

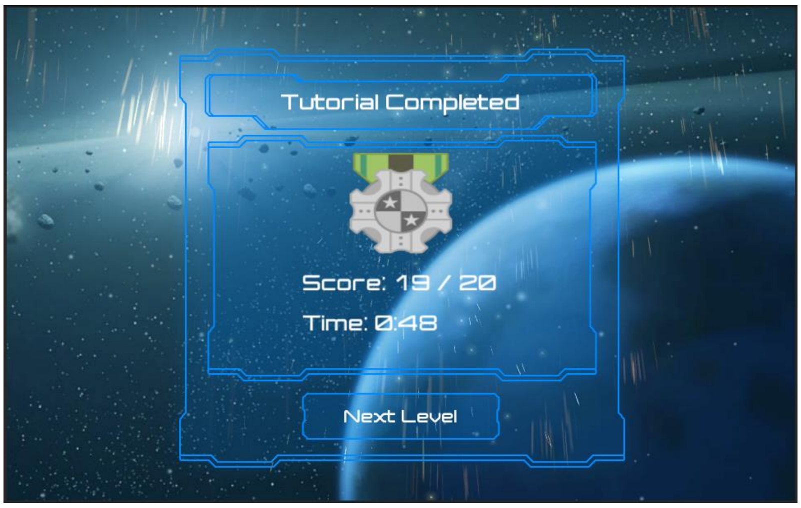
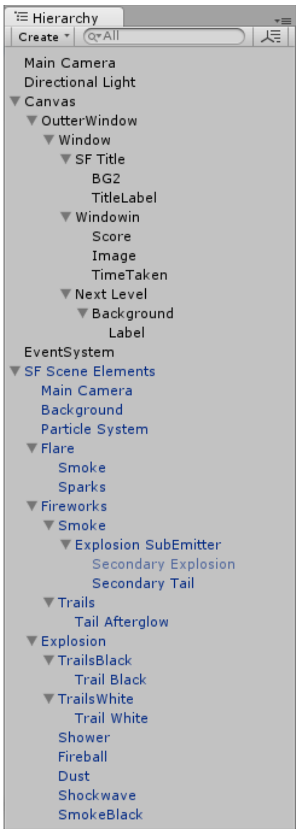


FIGURE 5.23: TUTORIAL REWARD SCENE

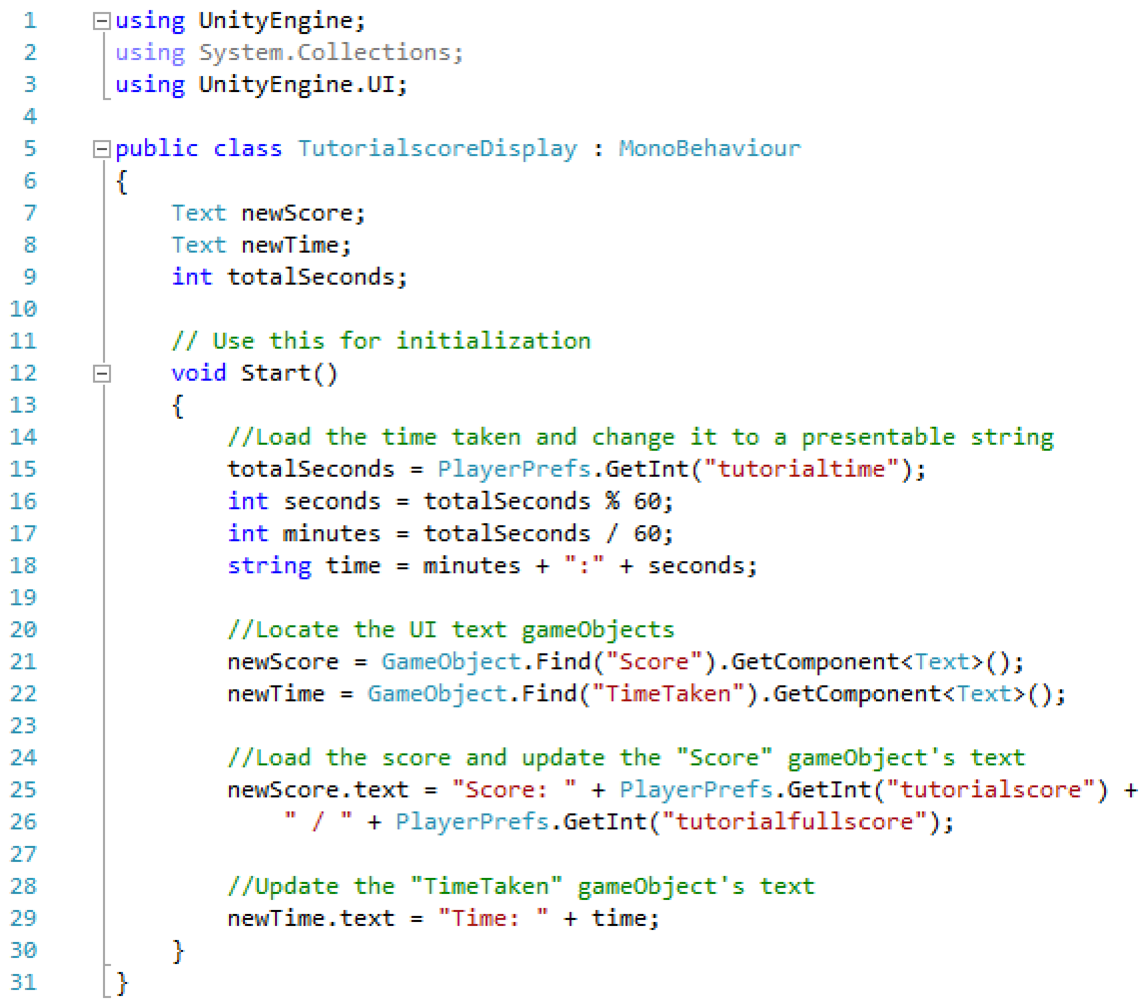
The hierarchy shown in the figure bellow, shows that the same "SF Scene Element" prefab as seen in all other scenes is containing three other objects. These are the particle systems prefabs "Flare", "Firework" and "Explosion" which were obtained from the standard assets located at the project window. The intensity and a suitable location for these particles to commence from was decided for each.



```
Hierarchy
Create ▾  All
Main Camera
Directional Light
▼Canvas
  ▼OutterWindow
    ▼Window
      ▼SF Title
          BG2
          TitleLabel
      ▼Windowin
          Score
          Image
          TimeTaken
      ▼Next Level
        ▼Background
            Label
EventSystem
▼SF Scene Elements
    Main Camera
    Background
    Particle System
  ▼Flare
      Smoke
      Sparks
  ▼Fireworks
    ▼Smoke
      ▼Explosion SubEmitter
          Secondary Explosion
          Secondary Tail
    ▼Trails
        Tail Afterglow
  ▼Explosion
    ▼TrailsBlack
        Trail Black
    ▼TrailsWhite
        Trail White
      Shower
      Fireball
      Dust
      Shockwave
      SmokeBlack
```

FIGURE 5.24: HIERARCHY OF TUTORIAL REWARD SCENE

The “Windowin” game object is a panel that contains the UI texts “Score” and “TimeTaken” game object and the UI image “Image”. This panel also has the script TutorialscoreDisplay attached to it. The “Image” game object contains an image of the digital badge. The “Score” and “TimeTaken” game objects are used to display the performance of the player on their previous level. The script TutorialscoreDisplay provides these UI texts with an updated text of the students score.



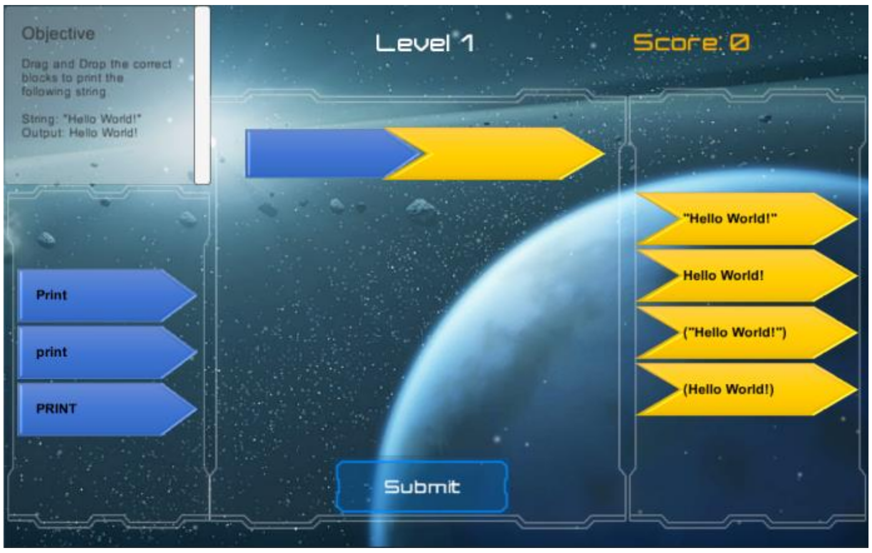
```
using UnityEngine;
using System.Collections;
using UnityEngine.UI;

public class TutorialscoreDisplay : MonoBehaviour
{
    Text newScore;
    Text newTime;
    int totalSeconds;

    // Use this for initialization
    void Start()
    {
        //Load the time taken and change it to a presentable string
        totalSeconds = PlayerPrefs.GetInt("tutorialtime");
        int seconds = totalSeconds % 60;
        int minutes = totalSeconds / 60;
        string time = minutes + ":" + seconds;

        //Locate the UI text gameObjects
        newScore = GameObject.Find("Score").GetComponent<Text>();
        newTime = GameObject.Find("TimeTaken").GetComponent<Text>();

        //Load the score and update the "Score" gameObject's text
        newScore.text = "Score: " + PlayerPrefs.GetInt("tutorialscore") +
            " / " + PlayerPrefs.GetInt("tutorialfullscore");

        //Update the "TimeTaken" gameObject's text
        newTime.text = "Time: " + time;
    }
}
```

FIGURE 5.25: TUTORIALSCOREDISPLAY CLASS

The “Next Level” game object is a UI button similar to the “Sumbit” game object detailed in the tutorial level. The purpose of this button is to change scene to Level 1 of the game.

5.7 Development of Levels and Reward Scenes

In order to simplify the development process for the levels and reward scene that remain to be built, the duplication of scene was found useful. The “Level1” scene is made by duplicating the tutorial level and making adjustments. The text in the “LevelName” game object was changed to “Level 1”. More puzzle pieces were created by duplicating the blocks and changing the text in them. The object containers were specified which of the blocks were able to drop in them, this is the same method as that of the tutorial level.


Objective

Drag and Drop the correct blocks to print the following string

String: "Hello World!"
Output: Hello World!

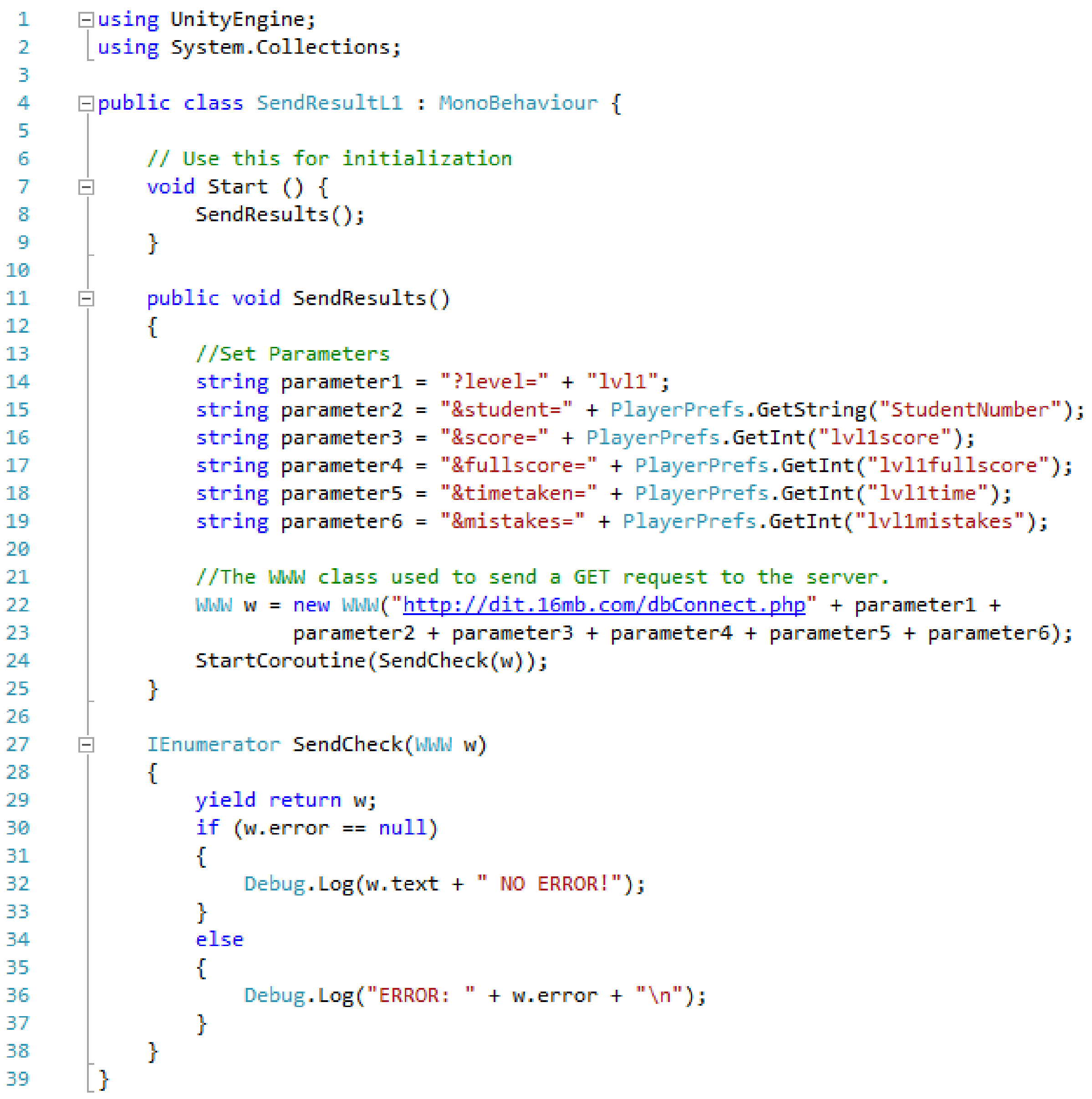
FIGURE 5.26: LEVEL 1 SCENE

The submit button runs the SceneChanger function from the Lvl1Submit class. This script is very similar to that of the tutorial level script TutorialSubmit. The only differences in this script is that, we are also saving the number of mistakes made by the player at that level and also the scene being loaded if the player is successful is the “Lvl1Reward” scene. The scoring system in the ScoreManager class is set to deduct 1 point for each mistake, thus calculating the number of mistake is as simple as deducting their score in that level with what the full score of the level is.

The “Lvl1Reward” scene which is loaded once Level 1 has been completed successfully was created by simply duplicating the “TutorialReward” scene and making adjustment to it. The digital badge was changed by simply attaching a different image to the UI image. The text of “-Tutorial Complete” was changed to “Level 1

Completed”. The Submit button uses the same class and function as that the submit button of the “TutorialReward” scene, but has the string “Level2” being passed on to the function in order to load the next level.

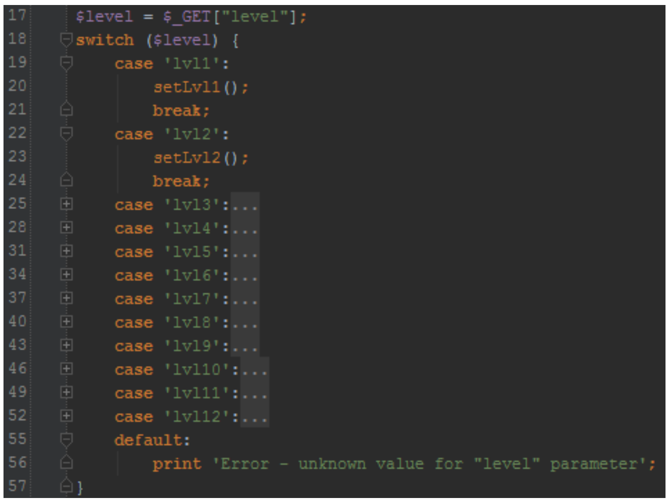
The TutorialscoreDisplay script was removed and a new script called Level1scoreDisplay was created. They are very similar to each other with the only difference being that the information being loaded is related to the scores at Level 1 rather than Tutorial level. The most important change in this “Lvl1Reward” scene is the addition of the script SendResultL1. This script was attached to the “Windowin” game object and was set to run once this scene is loaded. This script send the results of this level by the student to the server.



```csharp
using UnityEngine;
using System.Collections;

public class SendResultL1 : MonoBehaviour {

    // Use this for initialization
    void Start () {
        SendResults();
    }

    public void SendResults()
    {
        //Set Parameters
        string parameter1 = "?level=" + "lvl1";
        string parameter2 = "&student=" + PlayerPrefs.GetString("StudentNumber");
        string parameter3 = "&score=" + PlayerPrefs.GetInt("lvl1score");
        string parameter4 = "&fullscore=" + PlayerPrefs.GetInt("lvl1fullscore");
        string parameter5 = "&timetaken=" + PlayerPrefs.GetInt("lvl1time");
        string parameter6 = "&mistakes=" + PlayerPrefs.GetInt("lvl1mistakes");

        //The WWW class used to send a GET request to the server.
        WWW w = new WWW("http://dit.16mb.com/dbConnect.php" + parameter1 +
                parameter2 + parameter3 + parameter4 + parameter5 + parameter6);
        StartCoroutine(SendCheck(w));
    }

    IEnumerator SendCheck(WWW w)
    {
        yield return w;
        if (w.error == null)
        {
            Debug.Log(w.text + " NO ERROR!");
        }
        else
        {
            Debug.Log("ERROR: " + w.error + "\n");
        }
    }
}
```

FIGURE 5.27: SENDRESULTL1 CLASS

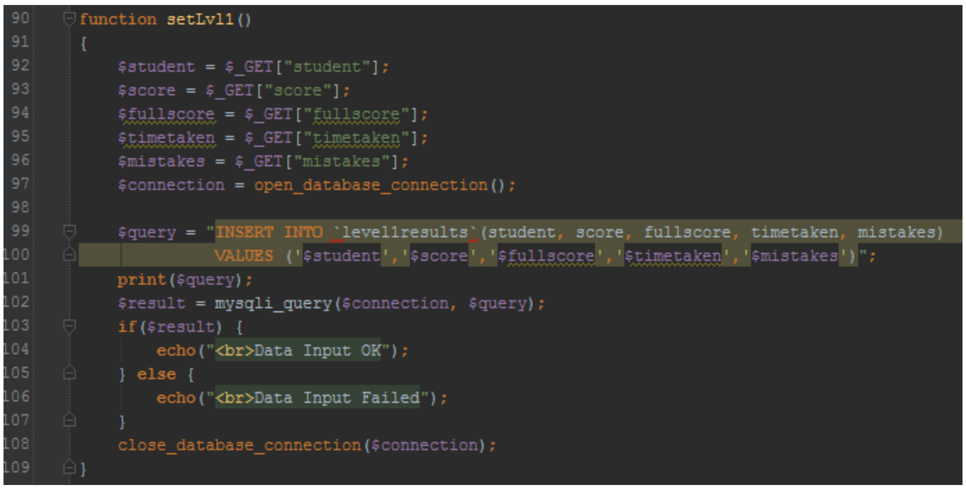
A PHP file named dbConnect.php handled the get request being sent. This PHP file sets the values being sent in the parameter by first identifying what which table it should be set through using the “level” parameter. A switch statement is used to identify which function should execute. Depending on the level parameter, it identifies which table in the database it should write to.



```php
$level = $_GET["level"];
switch ($level) {
    case 'lvl1':
        setLvl1();
        break;
    case 'lvl2':
        setLvl2();
        break;
    case 'lvl3':...
    case 'lvl4':...
    case 'lvl5':...
    case 'lvl6':...
    case 'lvl7':...
    case 'lvl8':...
    case 'lvl9':...
    case 'lvl10':...
    case 'lvl11':...
    case 'lvl12':...
    default:
        print 'Error - unknown value for "level" parameter';
}
```

FIGURE 5.28: SWITCH STATEMENT IN DBCONNECT.PHP

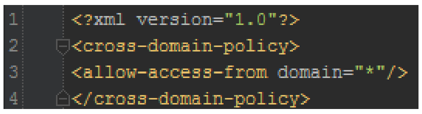
The function establishes a connection to the database, it than inserts the values into the database before closing the connection.



```php
function setLvl1()
{
    $student = $_GET["student"];
    $score = $_GET["score"];
    $fullscore = $_GET["fullscore"];
    $timetaken = $_GET["timetaken"];
    $mistakes = $_GET["mistakes"];
    $connection = open_database_connection();

    $query = "INSERT INTO `level1results`(student, score, fullscore, timetaken, mistakes)
              VALUES ('$student','$score','$fullscore','$timetaken','$mistakes')";
    print($query);
    $result = mysqli_query($connection, $query);
    if($result) {
        echo("<br>Data Input OK");
    } else {
        echo("<br>Data Input Failed");
    }
    close_database_connection($connection);
}
```

FIGURE 5.29: SETLVL1 FUNCTION IN DBCONNECT.PHP

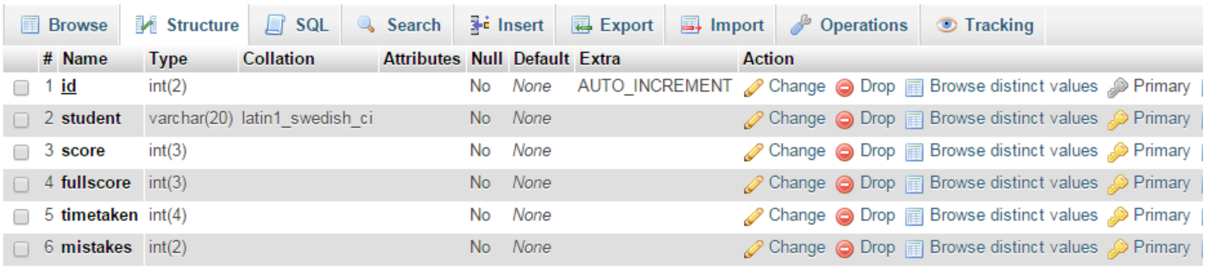
There are implications for using the WWW class in our script. Http server policy named crossdomain.xml is expected to be available on the domain one wants to access with the WWW class by the Unity web player. Although Unity's documentation stated this is not required if what one is accessing is located in the same location as the unity3d file, this was found to be incorrect. Even though the unity3d file and the PHP file where located in the same location, the web player still required crossdomain.xml for the WWW class to work. The crossdomain.xml file was only 4 lines long and was placed in the same location as the unity3d file and the PHP file.



```
<?xml version="1.0"?>
<cross-domain-policy>
<allow-access-from domain="*"/>
</cross-domain-policy>
```

FIGURE 5.30: CROSSDOMAIN.XML

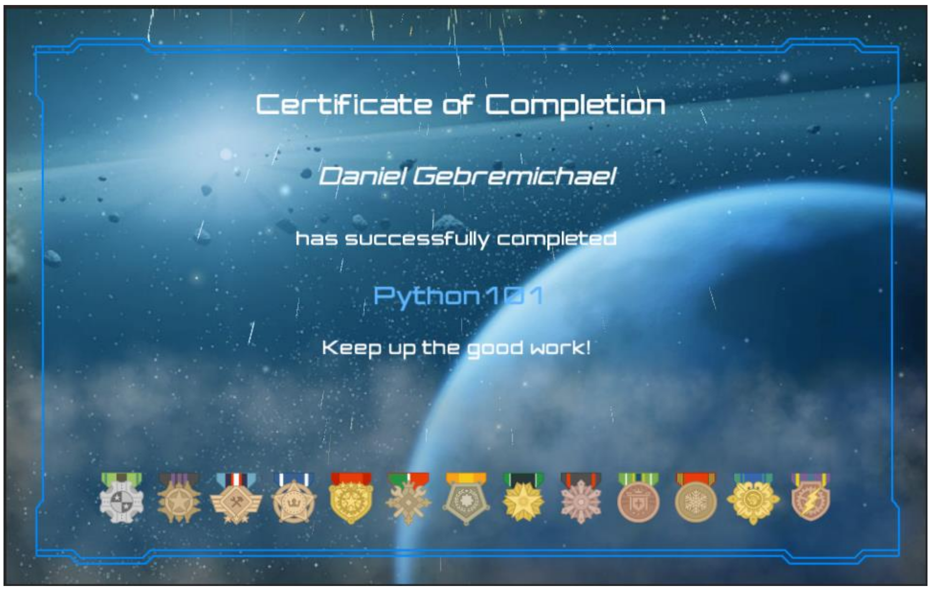
The figure bellow shows the structure of the levels. The primary is the id which is an auto incrementing integer.



#	Name	Type	Collation	Attributes	Null	Default	Extra	Action
1	id	int(2)			No	None	AUTO_INCREMENT	Change Drop Browse distinct values Primary
2	student	varchar(20)	latin1_swedish_ci		No	None		Change Drop Browse distinct values Primary
3	score	int(3)			No	None		Change Drop Browse distinct values Primary
4	fullscore	int(3)			No	None		Change Drop Browse distinct values Primary
5	timetaken	int(4)			No	None		Change Drop Browse distinct values Primary
6	mistakes	int(2)			No	None		Change Drop Browse distinct values Primary

FIGURE 5.31: DATABASE TABLE STRUCTURE

Once the student has finished all the levels, the "Award" scene is loaded by clicking on the finish button in the "Lvl12Reward" scene.



Certificate of Completion

Daniel Gebremichael

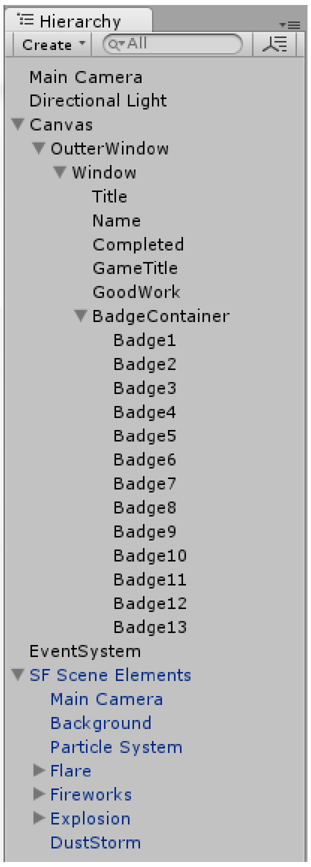
has successfully completed

Python101

Keep up the good work!

FIGURE 5.32: AWARD SCENE

This scene was developed using a duplication of the TutorialReward scene. This was done because this scene used all of the particle system elements observed in the "TutorialReward" scene and one more extra element. This is the dust storm particle effect which can be observed at throughout the bottom of the scene. This was done just to make sure that the badges did not go unnoticed.



```
Main Camera
Directional Light
▼ Canvas
  ▼ OutterWindow
    ▼ Window
        Title
        Name
        Completed
        GameTitle
        GoodWork
      ▼ BadgeContainer
          Badge1
          Badge2
          Badge3
          Badge4
          Badge5
          Badge6
          Badge7
          Badge8
          Badge9
          Badge10
          Badge11
          Badge12
          Badge13
EventSystem
▼ SF Scene Elements
    Main Camera
    Background
    Particle System
  ▶ Flare
  ▶ Fireworks
  ▶ Explosion
    DustStorm
```

FIGURE 5.33: HIERARCHY DISPLAYING ALL GAME OBJECTS

In the “window” panel, the “Title”, “Completed”, “GameTitle” and “GoodWork” UI text game objects exist. These simply contain text in a variety of manner. The “Name” UI text game object is slightly different, as it text depends on the script attached to it called “awardDisplay”. This script displays the name of the student.

There are 12 level in this game with a unique badge award for the level completion, which would be 13 badges in total when the tutorial level completion badge is included. All 13 of these badges are displayed in the scene. This was done by having a UI panel “BadgeContainer” which had a specific and important component added to it through the inspector. This component is the Grid Layout Group. This component was used to specify the child objects in this panel would adhere to the following rules; there size would be 60 by 60, there would be no spacing or padding between them, they are aligned horizontally and there alignment is middle centre.

The sequence of all the scenes is organised in the file -> build settings window. From the option of platforms to build for, Web Player was selected. Once this was done, it just required a single click on the build button and the location of where it should save the build to complete the development game.

5.8 Conclusions

This chapter details the environment setup that was required to develop the game for this project, which could also be used as reference for setting up an environment for other similar games. The detailed description of the use of Unity throughout the development process allows for those who are unaware of Unity and how it is used, to discover the process and become familiar to it. The development of the tutorial scene was so efficient that it allows for it be duplicated and modified to create levels 1 to 12. This was also the same for the tutorial reward scene and the reward scenes from all the levels. Unlike the tutorial reward scenes, the level rewards scenes were responsible for sending students results to the database and all the processes required to do so were detailed.

6 EXPERIMENTATION AND EVALUATION

6.1 Introduction

The pervious chapter detailed the development of the game, while this chapter discuss how the game was used to assess students learning. The use of the game developed to assess the students' learning allows for the research questions to be addressed. This chapter looks at the preparation of the experiments which included the game, survey and written tests. All the results obtained from the experiments are displayed and analysed. Interviews were also conducted once all results were obtained to address any remaining questions from the findings.

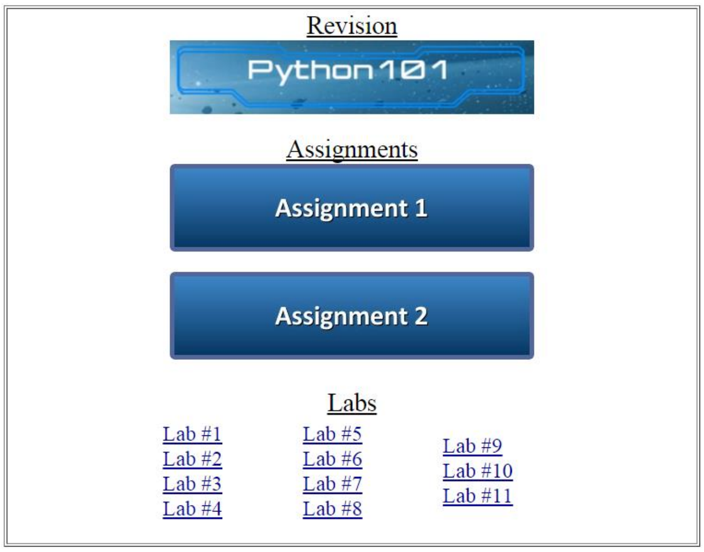
6.2 Experiment Preparation

A variety of websites that could be used to host the game were investigated, and after trial and error a host that could run the game with no expense for the website was identified. The host was Hostinger which provided free hosting with PHP and MySQL. The host doesn't apply banners or ads to the website. Unlike other free hosting services investigated, Hostinger allows unity3d file to be downloaded to the server and also allows information to be sent to the server via GET method.

The Unity build contained a unity3d file and html file which plays the game. This html file was renamed to index.html and extra code was added to it. This extra code was a link underneath the web player that open a survey in a new tab. This survey was developed to get feedback from the students about them game. The survey consisted of 10 questions and was created using Survey Monkey.

The experiment was planned to be carried after students have completed their normal laboratory work. It experiment was discussed with the laboratory lecturer and 30 minutes for the students to play the game and complete the survey at the end of their lecturer on the 23 of November was agreed. The laboratory computers were investigated to check if the game would work. Using the Firefox browser, the plugin for the web player was downloaded to run the game. This indicated that the laboratory computers could be used to run the game for the students.

In order to give a feeling that this game was part of their module, the lecturer kindly agreed to linking the website for the game, at the assignment section of the website where students get there lecturers and laboratory notes. The link of the game was placed under the title Revision.


Revision

Python101

Assignments

Assignment 1

Assignment 2

Labs

Lab #1	Lab #5	Lab #9
Lab #2	Lab #6	Lab #10
Lab #3	Lab #7	Lab #11
Lab #4	Lab #8	

FIGURE 6.1: LINK FOR THE GAME

On the day of the experiment at 5.30pm on 23th of November students were asked to participate in playing a game. All students present in the laboratory were willing to participate and informed consent was sought and given. Students were advised to use the Firefox browser to view the assignment section of the website where they obtain their notes. Students were allowed to use their laptops if they wished to do so and had Firefox installed. Students were instructed to click on the Python101 game link afterwards. All students were required to download a plugin for the web player. Once all students had the game running, they were told to finish the game and to complete the survey that is linked bellow the game afterwards. All students were capable of playing the tutorial level and the rest of the game without any guidance. The database was checked to make sure all students had competed all the levels.

6.2.1 Survey

The survey was developed based on a number of papers from the existing literature, with special focus on (Sweetser and Wyeth, 2005) paper.

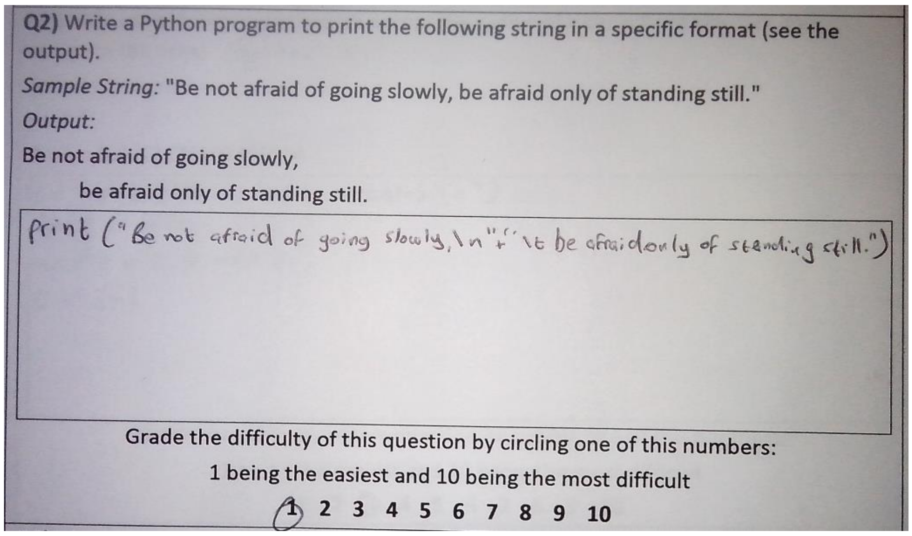
The questions asked in the survey are as follows;

1. Did you like the game?

2. Did you like the visuals in this game?
3. Did you find the tutorial level to be adequate enough to understand the concept of the game? If not, explain?
4. How did you find the difficulty of the levels?
5. Was there anything about the game you didn't like?
6. Compared to programming in class and written tests, do you believe games such as this can be a good alternative?
7. Compared to programming in class and written tests, did you find the game to be less stressful?
8. Would you think a game such as this, is a fair tool to test students on their learning?
9. Can you see a game like this being a beneficial revision tool?
10. Would you ever use a game like this for revision?

6.2.2 Written Test

On the 30th of November at approximately 5:30pm students were presented with question sheets. These questions were identical to that of the question sheets provided during the design stage, with only one change made to them. This change was the inclusion of equations, the students would require to write code for two questions. The reason for the inclusion of the equations are discussed in the design chapter. The questions can available in the design chapter.



Q2) Write a Python program to print the following string in a specific format (see the output).

Sample String: "Be not afraid of going slowly, be afraid only of standing still."

Output:

```
Be not afraid of going slowly,
        be afraid only of standing still.
```

```
print ("Be not afraid of going slowly,\n"+"\t be afraidonly of standing still.")
```

Grade the difficulty of this question by circling one of this numbers:

1 being the easiest and 10 being the most difficult

[✓ 1] 2 3 4 5 6 7 8 9 10

FIGURE 6.2: PART OF A FILLED OUT SHEET

6.3 Evaluation of Results

The results obtained from the experiment were analysed. The analysis of the results are aimed to address the research questions.

6.3.1 Game Results Evaluation

In order to assess students' learning with the game, two very important sets of information were required. The first was to have a graph that shows students number of mistakes at each level. Many information can be obtained through the number of mistakes students are making. It allows the lecturer to identify which student struggles with which question and the average level of students' knowledge with certain concepts. Creating a graph to display these results can help to view many trends. As the levels were created in sets, which means every two questions are similar to each other, we can see whether the students learn from their mistakes and improve their results in a more difficult question. The graph below shows the average number of mistakes made by students at each level.

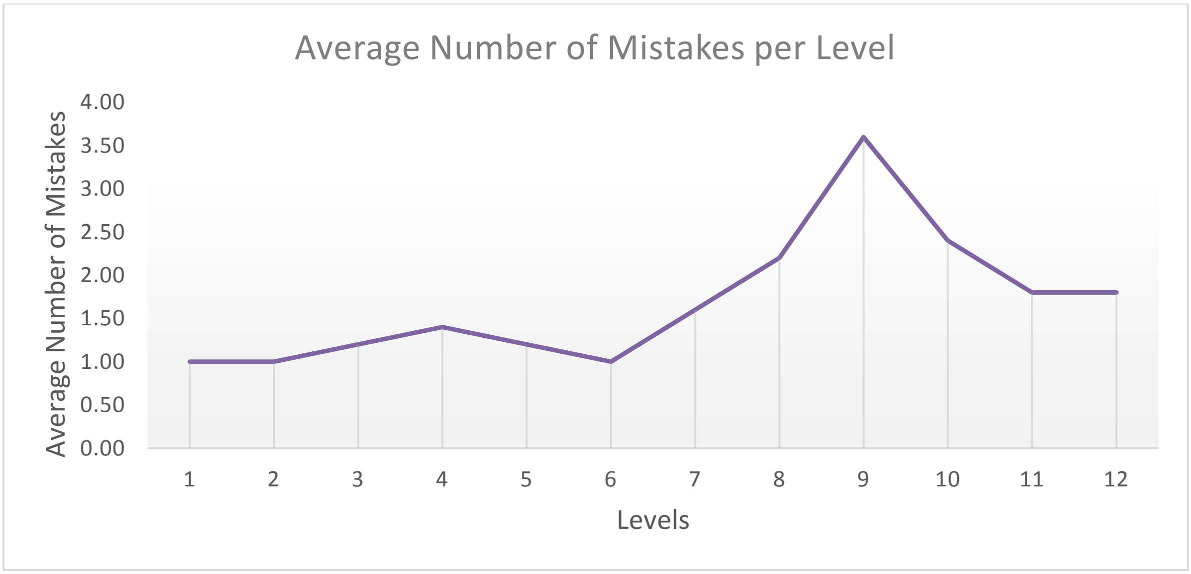


FIGURE 6.3: GRAPH OF THE AVERAGE NUMBER OF MISTAKES PER LEVEL

As already mentioned, due to the design of the levels every two levels have similar question with the second one being more difficult. For example, level 1 and level 2 are similar but level 2 is more difficult, this is the same for level 3 and 4 and so on. Students are expected to make more mistakes in the more difficult level of each set but this is not the case from our results. Our results show a decrease in mistakes from level

5 to 6 and from 9 to 10. In order to simplify the result from the graph to a more meaningful result we address the trends of 2 sets of levels.

There was an increase in the number of mistakes going from level 3 to 4 and level 7 to 8. When investigating these levels in order to make sense of the result, an interesting find was made. Level 3 and 4 had the same question but different ways to answer them and 7 and 8 had the same question as each other but also answered in a different way.

Level 3 and 4 had the same objective of;

Drag and Drop the correct blocks to print the following string.

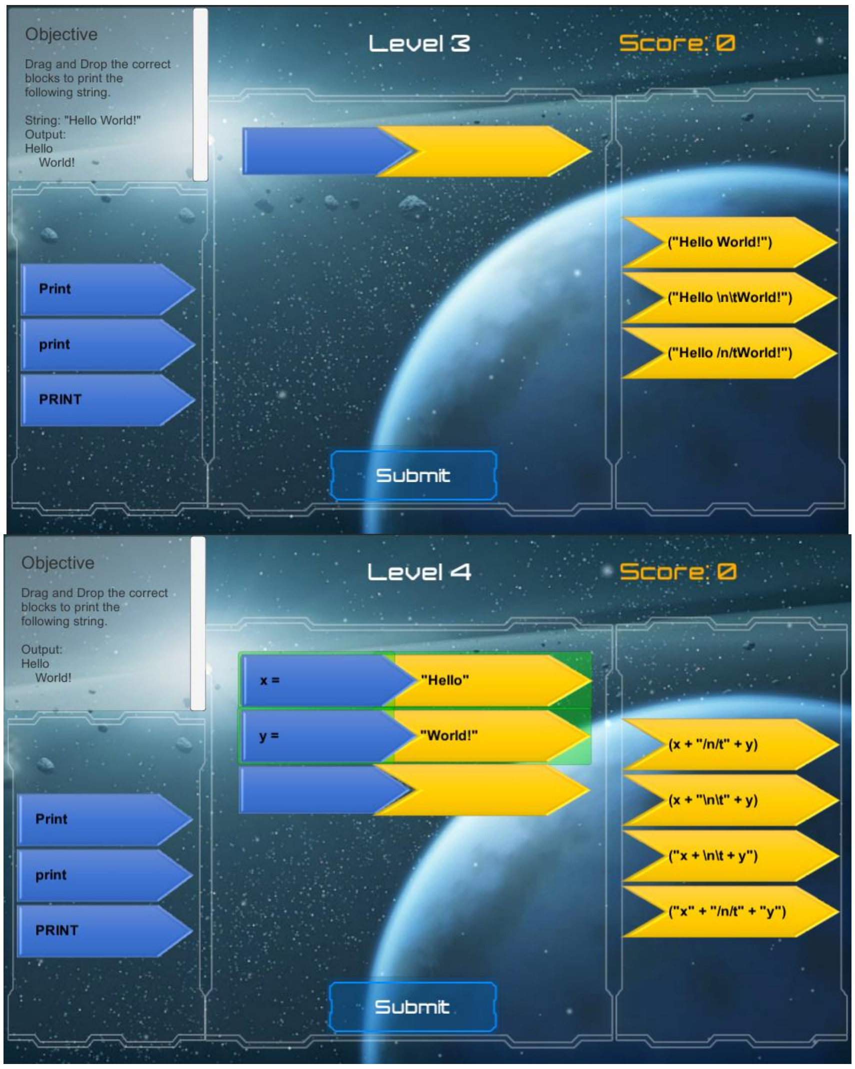
String: "Hello World!"

Output:

Hello

        World!

The reason for the increase in the number of mistakes at level 4 is due to the fact that the answer to level 4 was a bit more complicated than level 3. There was an indication of students who made mistakes at level 3, making mistakes on level 4 too. It is as if they have forgotten the answer they gave just a few seconds ago. This could be to the students' lack of attention and difficulties they have with variables.


Objective

Drag and Drop the correct blocks to print the following string.

String: "Hello World!"
Output:
Hello
    World!

Objective

Drag and Drop the correct blocks to print the following string.

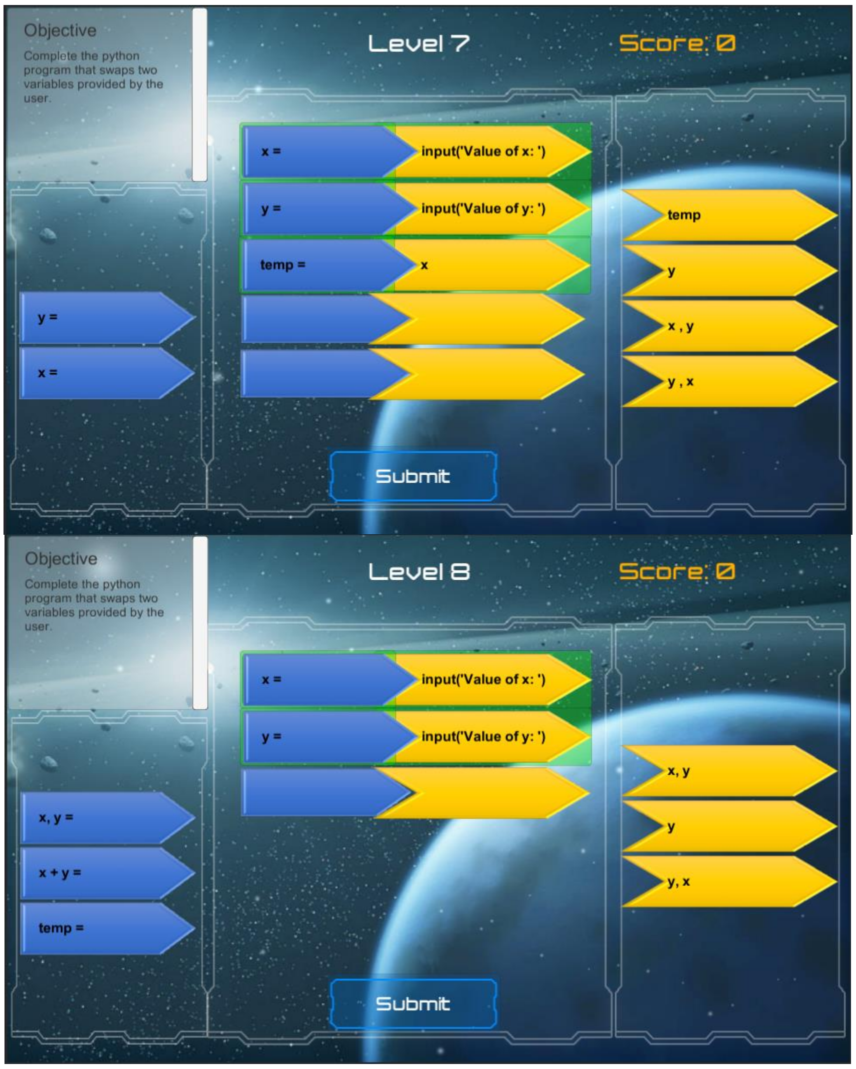
Output:
Hello
    World!

FIGURE 6.4: GAME SCENES OF LEVEL 3 AND LEVEL 4

The Objective of both question 7 and 8 was;

Complete the python program that swaps two variables provided by the user.

The actual puzzle and puzzle pieces vary which shows the students a variety of ways of answering questions. Students were expected to get less number of mistakes at level 8 when compared to level 7. Some of the reasons for this increase in the number of mistakes were observed during the experiment. This was due to the influence of a single student. The student repeatedly chose the same incorrect piece, thus making many mistakes due to the belief that student had, which was that the student believed there was a mistake with the game. This issue was solved with the student by indicating to the student that the game was in fact working fine. It is important to note that, even if the results of that particular student was removed from the result, the average number of mistakes for level 8 would still remain higher than level 7. Thus, it wouldn't have changed the fact that an average number of more mistakes were made in level 8, but would have had almost the same average number of mistakes as level 7.


Objective

Complete the python program that swaps two variables provided by the user.

```
x =     input('Value of x: ')
y =     input('Value of y: ')
temp =  x
```

Objective

Complete the python program that swaps two variables provided by the user.

```
x =     input('Value of x: ')
y =     input('Value of y: ')
```

FIGURE 6.5: GAME SCENES OF LEVEL 7 AND LEVEL 8

Number of mistakes remained unchanged for level 1 to 2, and for level 11 to 12. Students that did not made mistakes in level 1 did not make mistakes in level 2. The remainder of the students, ether made more mistakes or less mistakes from level 1 to 2 and no pattern was found. Level 11 and 12 may have the same average number of mistakes but when inspecting individual mistakes, it was apparent that the number of mistakes by some students were very high while the many of the students made only one mistake. Although no student got level 12 completed without a mistake, the same high number of mistakes by individuals as observed in level 11 was not found.

Decrease in the average number of mistakes was found for level 5 to 6 and level 9 to 10. There were more students in level 6 that got the question correct without a mistake than in level 5. Results show that some students learnt from the mistakes they made in level 5. Level 9 had the highest average number of mistakes when compared to all other questions. Students found this level to be the most difficult. Level 10 was very similar to question of level 9 and had the second highest average number of mistakes. The reason to why there was a decrease in the number of mistakes from level 9 to 10 is simply due to some students identifying where they went wrong in level 9. This still did not change the fact that the students with the largest number of mistakes in level 9, making the same high number of mistakes in level 10.

The second piece of information one can also get from this game is, how long it took students to complete each level. It is important to note that this type of information is difficult to work with as some students may be slower readers than others and the number of puzzle pieces vary through the levels. When viewing the average time taken for students to complete the level with the average number of mistakes per level, there are some noticeable features between them. The level with the most mistakes had the longest average time for that level to be complete. Level 5 had a longer question than its previous levels and required more puzzle pieces than the previous levels. Although level 8 had the third highest average number of mistakes, due to the fact that it only had two puzzle pieces to be completed meant it required less time to complete. Level 10 was the level with the second highest average number of mistakes, yet it had a very low average of time taken for students to complete that level. This was not due to the number of puzzle pieces as it had more puzzle pieces. This is an indication that the students understood the question must faster than as it had similarities to the previous level and made decisions must faster than in level 9.

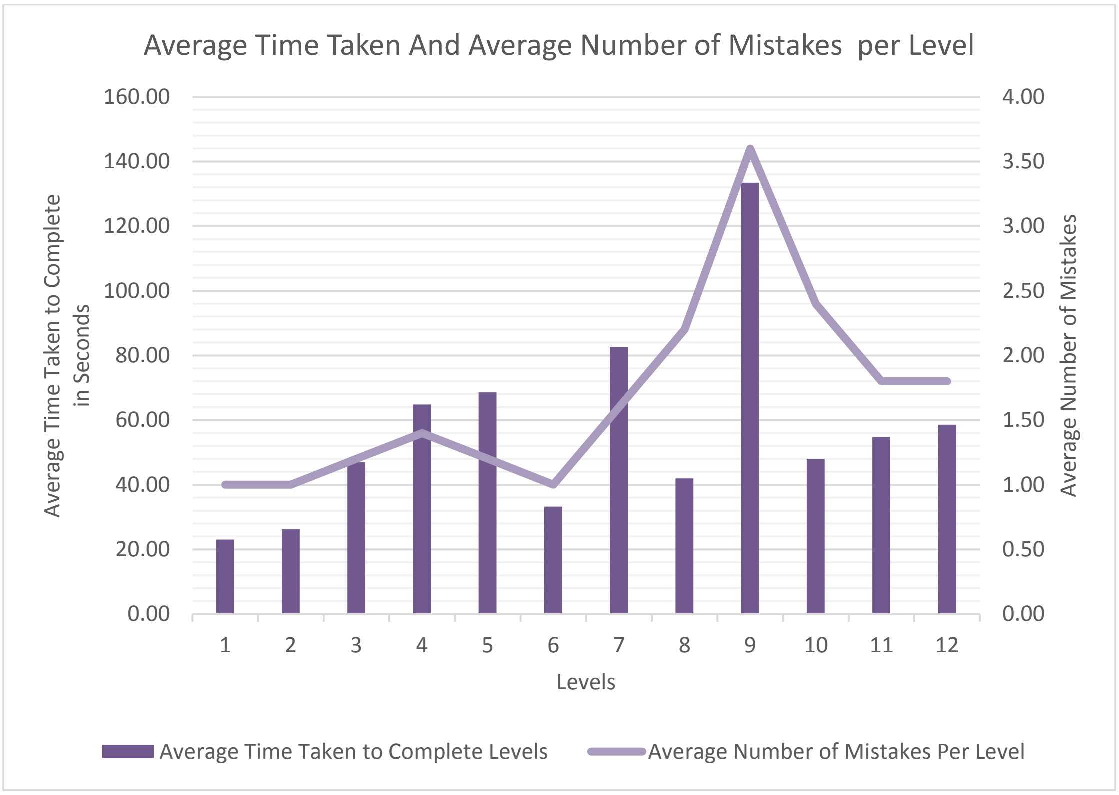


FIGURE 6.6: COMBINATION GRAPH OF THE AVERAGE TIME TAKEN TO COMPLETE A LEVEL AND AVERAGE NUMBER OF MISTAKES PER LEVEL

6.3.2 Post-Game Survey Evaluation

80% of the students who participated in playing the game completed the survey presented to them. All these students did so, as soon as they completed the game. The percentage of students who completed the survey is very high as students were not being pressured to complete the survey, they did so simply to help with this research. The responses from the students on the 10 survey questions are examined one by one.

Question 1: Did you like the game?

The students where given three choices of answers. They were yes, no and partially. 75% of the students responded with yes and 25% respondanded with partially. This indicated that no student disliked the game. It is also indicates that 25% student liked some aspects of the game. The fact that the majority of students like the game indicates that the game was well received by the students. The 25% of the students that like the game partially, is also seen as a possitive feedback as it may identify issues students may have with the game.

Question 2: Did you like the visuals in this game?

All the students like the visuals in the game.

Question 3: Did you find the tutorial level to be adequate enough to understand the concept of the game? If not, explain?

All students believed that the tutorial level was adequate enough to understand the concept of the game. One student responded with "real good and funny".

Question 4: How did you find the difficulty of the levels?

All students indicated that the difficulty of the game where fine. No negative feedback of the difficulty of the levels were made. An example of the student response is as follows "They were good and got reasonably more difficult as the game progressed but stayed understandable".

Question 5: Was there anything about the game you didn't like?

No student indicated that there were anything about the game that they didn't like. Although one student out of context noted that they didn't like the time of day they played this game. This is important to note, even though it was not the question asked. This identifies that the time of day the game is played may have some sort of a negative effect on the student.

Question 6: Compared to programming in class and written tests, do you believe games such as this can be a good alternative?

Majority of the students indicated that they do believe games such as this could be a good alternative than to programming in class or written tests. Some of the responds are "yes I do, indeed I will be playing it at home" and "yes as it was more hands on". Although there were no students who completely disagreed, there was indications that the students enjoyed their programming in class with their lecturer as indicated by this student, "I don't know but Damien Gordon is an awesome teacher".

Question 7: Compared to programming in class and written tests, did you find the game to be less stressful?

All students believed that the game was less stressful than programming in class or written test and one student even went further and said “it was good fun too” in their response.

Question 8: Would you think a game such as this, is a fair tool to test students on their learning?

This was an important question and the student responses were similar. All students believed that this game and others similar where a fair tool to test their learning. Responses such as “Yes! Definitely! 100%!” and “yeah, I think if you get games that are suited to the level you're at it, it is a good and fair test” indicate that students are willing to be assessed through games.

Question 9: Can you see a game like this being a beneficial revision tool?

All students believed that this game and others like it can be a beneficial revision tool. No student went into detail to why but simply responded with yeses.

Question 10: Would you ever use a game like this for revision?

When it came to the students themselves using a game such as this for a revision tool, not all students agreed. 75% of the students would use a game like this as a revision tool. 25% of the students were unsure. No student disagreed with the notion of a game such as this being used as a revision tool.

6.3.3 Post-game written test evaluation

Students were given a post-game written test, with the same questions as the pre-game written test. This was done to investigate, if any change has occurred in terms of students’ assessment of the difficulty of these questions and if the students’ scores changed in anyway. The question sheets contained 7 question where they were expected to write python code, and the students provided how difficult they found each question by grading the question’s difficult with a number between and including 1 and 10, where 10 indicated the questions to being most difficult and 1 being the most easiest.

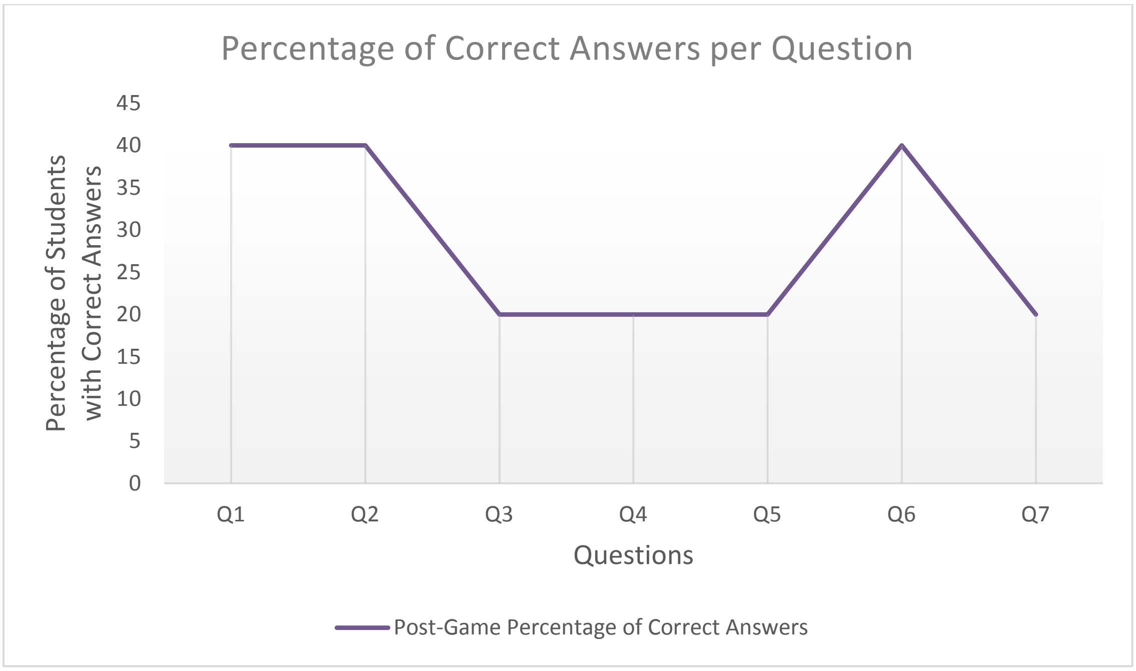


FIGURE 6.7: GRAPH OF THE PERCENTAGE OF CORRECT ANSWERS PER QUESTION

40% of the students were able to answer question 1 and 2 correctly. The percentage of correct answers would have doubled to 80%, if students didn't not make a mistake of not including parentheses. There seems to be an issue with students not using parentheses in other questions as well. Students found questions 3, 4 and 5 difficult. This is due to the combination of lack of parentheses and logic error. There were cases, where students wrote compiling code but had logic error as the output were undesirable and did not answer the question.

Question 6 had significantly more correct answers than question 5, this was unexpected. It would not have been surprising if question 6 had a small increase of correct answers, as questions 5 and 6 where similar questions but question 6 was less complicated. There were indications that students struggle with squaring a value in question 5, and the equation for question 6 did not require any number to be squared. Question 7 was expected to be the most difficult for students to do, as it was the most difficult question out of these 7 questions. 20% of the students got this question correct. The same mistakes from students were observed, these where lack of parenthesises and logic errors. Overall, 20% of the students were able to answer all questions.

The final page of the question sheets contained a method for the students to grade their knowledge in 4 different concepts in programming. This was done by students grading their knowledge of each concept from 1 to 10. 1 indicating the least knowledge and 10 indicating the highest knowledge.

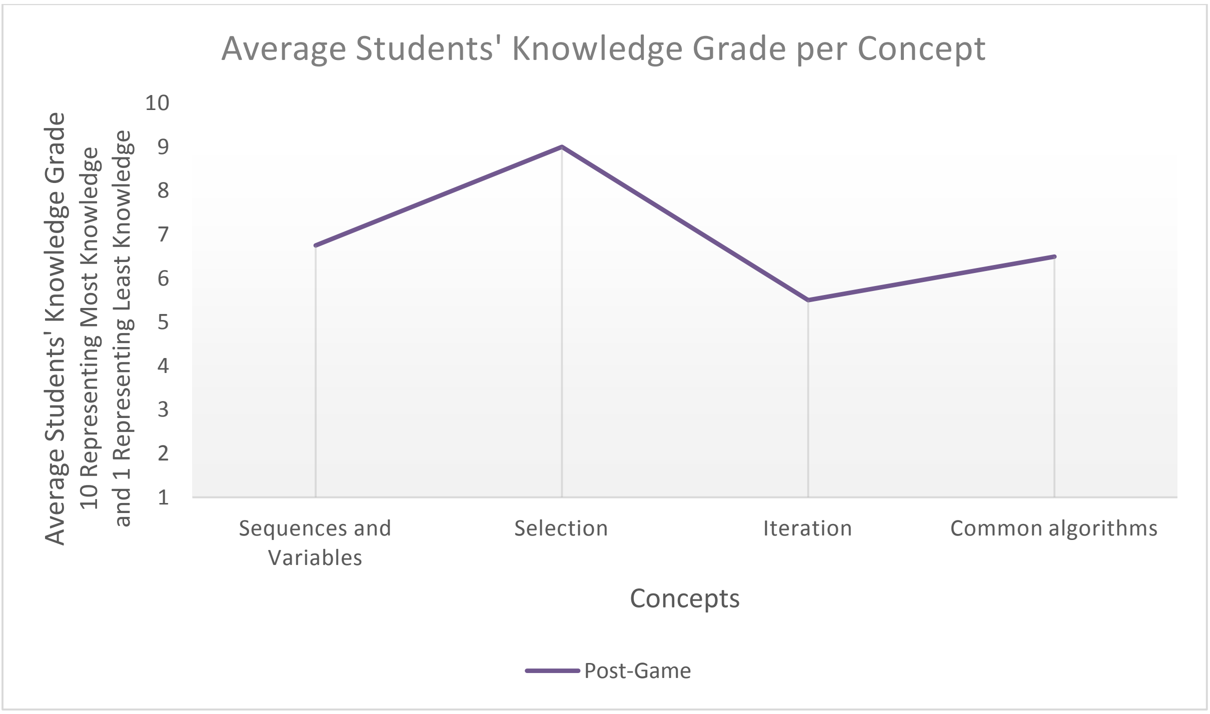


FIGURE 6.8: GRAPH OF THE AVERAGE STUDENTS' KNOWLEDGE GRADE PER CONCEPT

The students' average grade in their assessment of their knowledge indicated that they have most knowledge in Selection. This means that students believed they had very good knowledge of IF statements, CASE statements and Boolean logic. This indicated that students believed they had more knowledge in Selection than in Sequences and Variables. This is an expected result as one would have to have a good knowledge of Sequences and Variables before they can have good knowledge in Selection. Common algorithms which include Prime Numbers, Fibonacci Numbers and Compression had a higher average of knowledge than that of Iteration, which is yet again an unexpected result. This is because Common algorithms contain Iteration. The reason for Common algorithms having a higher average could be due to students simply remembering the code.

Bloom's Taxonomy on the cognitive domain from (Bloom et al, 1956) which involved *"knowledge and the development of intellectual skills"*, which (Krathwohl, 2002) made a modification to may help to understand at what stage of the cognitive domain students are at.

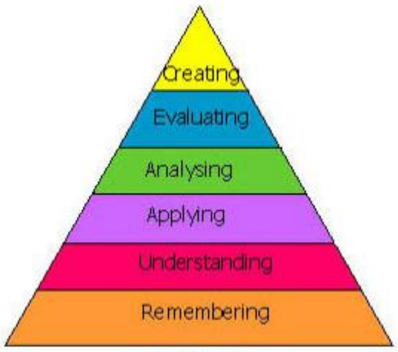


FIGURE 6.9: BLOOM'S TAXONOMY ON THE COGNITIVE DOMAIN REVISED (KRATHWOHL, 2002)

As 20% of the students were capable of answering all the questions, it can indicate that at least 20% of the students are at the applying stage. This means that they remember, understand and apply their knowledge to answer the 7 questions. Majority of students understood what they needed to do to answer the question but failed in the applying stage by making mistakes in their code.

In terms of the four concepts students were asked to grade their knowledge on. Students may be indicating that they understand concepts such as Common algorithms by giving it a high grade. Student may be giving the Iteration a lower grade than Common algorithms even though some Common algorithms contain Iterations in them because students may find it difficult to apply or analyse their knowledge of iteration. It is difficult to understand why Selection has a higher grade of understanding than that of Sequence and Variables, this is could be due to some behaviourism related reason, because it was the first programming concept they have learnt and they may be grading it as difficult due to their past experience rather than current. It could also be due to the game having an effect on the students' opinion on their knowledge, this is discussed in more detail in the research question section in this chapter.

6.3.4 Addressing Research Questions

Research Question 1

Using a written test that contained a method for students to grade their knowledge of different programming concepts, would the students' assessment of their knowledge differ significantly after they have participated in playing the game developed to assess them?

First hypothesis 1 – The game developed to assess the students' knowledge would not change the students' opinion on their level of knowledge of different programming concepts.

Second hypothesis 1 – The students would assess their knowledge of different programming concepts differently after they have participated in playing the game developed to assess their knowledge.

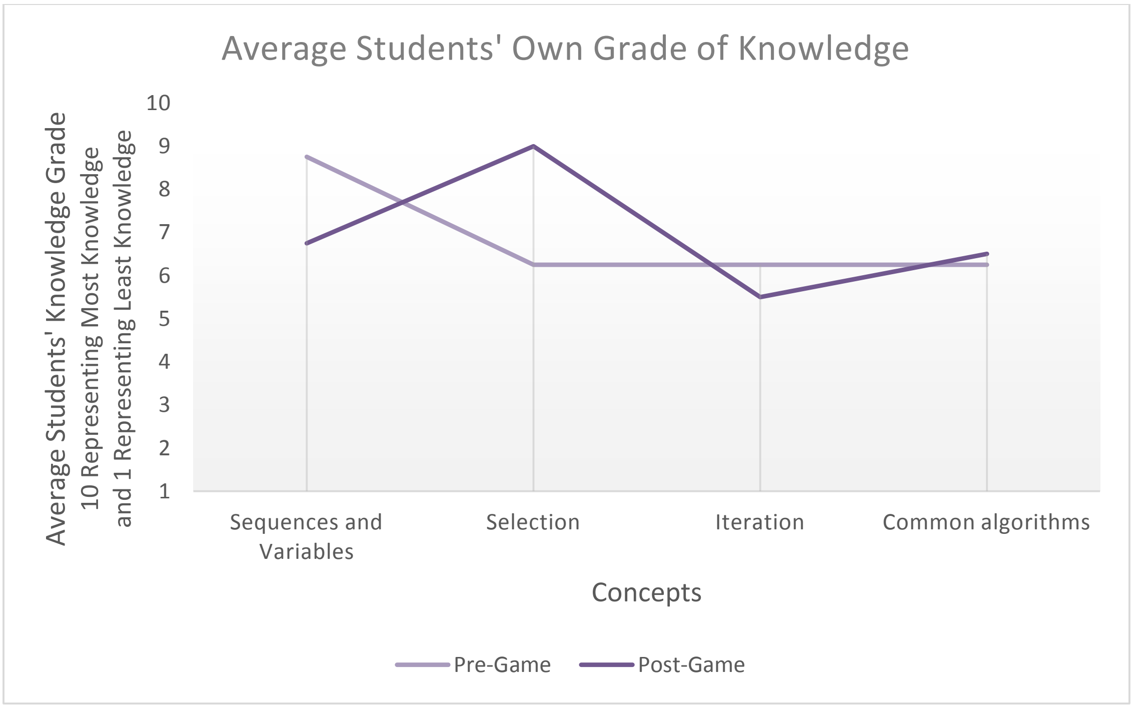


FIGURE 6.10: GRAPH OF AVERAGE STUDENTS' OWN GRADE OF KNOWLEDGE BEFORE AND AFTER PLAYING THE GAME

When comparing the students' own assessment of their knowledge in four programming concepts before and after they participated in playing the game, a significant difference was identified. This verifies the second hypotheses of a change of students’ own assessment of their knowledge to be noticeably different.

Students are indicating that they have less knowledge in Sequences and Variables after they have played the game. This could be due to the students’ realisation of their lack of knowledge after they have played the game. Students who may have graded their knowledge of Sequences and Variables high before the game, may have made mistakes in the game, leading them to noticing that their high grade in this concept may have been incorrect.

Students graded their knowledge in Selection significantly higher after the game. There may be many various reason for this, but the fact that more students got questions that contained Selection concepts correct and close to being correct could be the major factor.

The grade for student’s knowledge in Iteration decreased slightly. The reason for this remains unclear, but one hypotheses for this change is that, students’ realisation of their lack of knowledge. Students’ grade on their knowledge in Common algorithms increases but the change in grade was very small, and not enough to indicate a noticeable increase in students understanding in Common algorithms.

Research Question 2

Using a written test before and after the students have participated in playing the game developed to assess their knowledge, would the average score of the students’ written test differ significantly?

First hypothesis 2 - Using a written test before and after the students have participated in playing the game developed to assess their knowledge, the average score of the students’ written test would have no to little variation.

Second hypothesis 2 - Using a written test before and after the students have participated in playing the game developed to assess their knowledge, the average score of the students’ written test would change significantly.

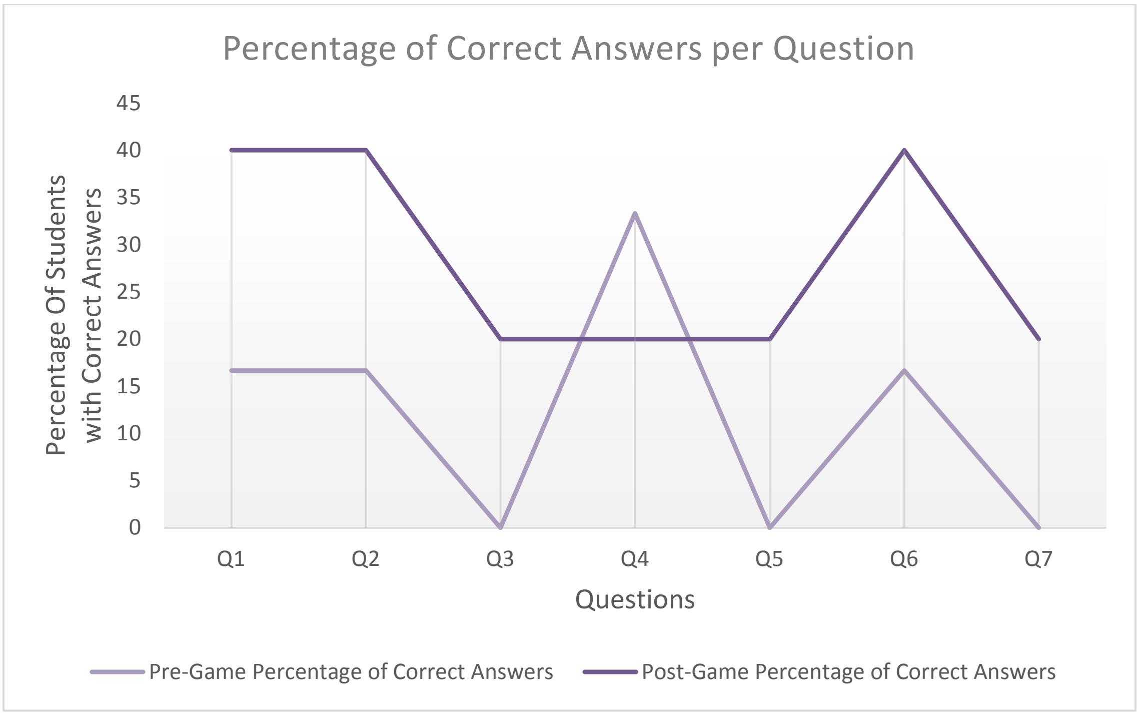


FIGURE 6.11: GRAPH OF PERCENTAGE OF CORRECT ANSWERS PER QUESTION BEFORE AND AFTER PLAYING THE GAME

Students answered more questions correctly in the post-game test than the pre-game test. This verifies the second hypotheses of student being able to answer a significant amount more questions correctly in their post-game written test. The percentage of correct answers increased in all questions but question 4. Surprisingly, question 4 had less correct answers in the post-game test. This clearly indicates that, students who were capable of getting question 4 correct, got it incorrect.

The growth of the percentage of students who got the questions correct shows an equal percentage of increased correctness for questions 1, 2 and 6. This is also true for questions 3, 5 and 7. This is a positive result as, questions 3 and 5 had no correct answers in the pre-game test. In order to identify, the reasons to why there was a decline of correct answers for question 4 were examined. There were indications that some students were very close to getting the answers correct. There code would have compiled but would not have given the correct answer. On the other hand, some students were unable to attempt this question.

Research Question 3

Using a game developed to assess students' learning, would students prefer to be assess through games rather than programming in class or written test?

First hypothesis 3 - Using a game developed to assess students' learning, students would not prefer to be assessed through games.

Second hypothesis 3 - Using a game developed to assess students' learning, they see some benefits of being assessed through games.

Two questions from the post-game survey addressed this research question. These questions were;

Compared to programming in class and written tests, did you find the game to be less stressful?

Would you think a game such as this, is a fair tool to test students on their learning?

From the answers to the first question, all students claimed that the game was less stressful than other methods of assessment with some even noting that they found it fun. This indicates that students have a more positive view of games being used for assessment than that of other methods mentioned. The second question directly asks the students the research question. It asked the students opinion of the game being used as method to assess their learning while comparing it to current methods used. All students believed that this game and others similar to it where a fair tool to test their learning. Responses such as "Yes! Definitely! 100%!" and "yeah, I think if you get games that are suited to the level you're at it, it is a good and fair test" indicate that students are willing to be assessed through games. This confirmed the second hypothesis of students being able to see some benefits of games being used to assess learning.

Research Question 4

Using a game developed to assess students' learning, would students like the game even though it is used to assess their knowledge?

First hypothesis 4 - Using a game developed to assess students' learning, students would not show signs of liking the game.

Second hypothesis 4 - Using a game developed to assess students' learning, students would show some signs of liking the game even if it was assessing their learnings.

The second hypothesis was verified to be correct due to two post-game surveys that addressed this question. These questions were the first two questions in the survey, the first asked if the students liked the game and the second if the liked the visuals in the game. The responds from the students indicated that majority of the students enjoyed the game. No student disliked the game and 25% of the students partially liked the game. As all the students indicated that they liked the visuals, the reason why 25% of the students partially liked the game was not due to visual or graphic related reasons. Another survey question that asked if the tutorial level to be adequate enough to understand the concept of the game, and showed that all students believed the tutorial level was adequate enough. This means that the student who partially liked the game was not due to the lack of understanding the game mechanics.

Students already indicated in the survey that they found the levels of the game reasonable and that it increased in difficult as they progressed in the game. This means that we can rule out that the game being difficult was not the reason students partially liked the game. It can be assumed that there might have been something in the game that might have caused the students to dislike parts of the game, but a survey question that asked if the students disliked anything about the game showed that all student had no negative feedback. This makes it difficult to understand to why 100% of the student did not like the game fully when they all indicate that they found the levels adequate, game mechanism easy to understand due to the tutorial level and even indicating that there were nothing about the game they disliked. The reason for this could be due to the 25% of the students actually liking the game but may have wanted more features as all the features in it were not disliked. This may mean that the game was liked by all students and 25% of the students saw ways to improve the game without removing any existing components.

Research Question 5

Using a written test, would students find the questions in the written test to be less difficult after playing a game developed to assess the students learning?

First hypothesis 5 - Using a written test, students will not find the questions in the written test to be less difficult after playing a game developed to assess the students learning.

Second hypothesis 5 - Using a written test, students will find some or all questions in the written test to be less difficult after playing a game developed to assess the students learning.

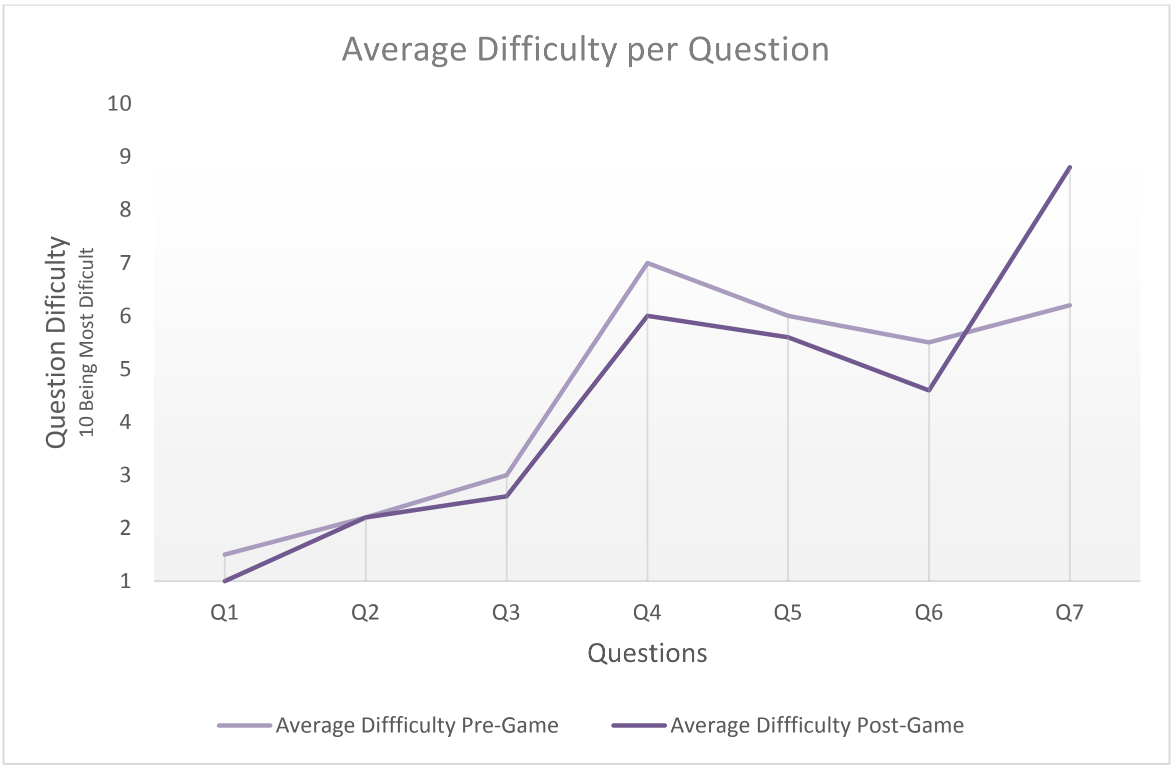


FIGURE 6.12: GRAPH OF AVERAGE DIFFICULTY PER QUESTION BEFORE AND AFTER PLAYING THE GAME

The students graded the difficulty level of all questions in the question sheet, in the same manner as the pre-game test. All students gave the lowest difficulty level of 1 for question 1. The difficulty of this question has decreased since the pre-game test. There were no changes in the difficulty of question 2. Questions 3, 4, 5 and 6 were given a lower difficulty level in the post-game test. Out of these four question, questions 4 and 6 has there difficulty level significantly decreased.

Unexpectedly, question 7's difficulty level had increased dramatically in the post-game test. It had the biggest change in difficulty than all other questions asked. All questions except for question 7 had an equal or decreased level of difficulty in the post-game test. The reason for the significant increase in question 7's difficulty could be due to students understanding increasing, and realising that the question was in fact difficult. As more students got question 7 correct in the post-game written test, this may indicate that students might have given question 7 a lower grade in difficulty in the pre-game written test due to their lack of understanding the difficult of the question.

The second hypothesis was verified as 5 out of the 7 questions were indicated by the students to be less difficult in their post-game written test.

Research Question 6

Using interviews and surveys, can a game developed to assess students also be used for revision purposes?

First hypothesis 6 - Using interviews and surveys, the game developed to assess students will be identified to not be useful as a revision tool.

Second hypothesis 6 - Using interviews and surveys, the game developed to assess students will be identified to have some useful beneficiaries as a revision tool.

This research question was address by two methods. One was a post-game survey asking students if they could see a game such as the one they played could be a beneficial revision tool and another question which asked if they would use a game such as the one they played for revision purposes. The second method was to identify if lecturers that have or are currently teaching programming to first year computer science students in third level education could see a potential for games to be used as revision tools, by using a semi-structured interview.

The survey of the students identified that all students believe that games such as the one they played would be a beneficial revision tool. When students were asked if they would use it as a revision tool themselves, the results were totally one sided. Although no student indicated that they wouldn't use it as a revision tool, 25% of the students remained unsure if they would use it. This uncertainty could be totally eradicated if a lecturer indicated to the students that the revision tool was useful to them. As these students already indicated that the game could be seen as a beneficial revision tool, just how beneficial it was to them main have remained uncertain. There are two factors that may entice all students to use this game and other similar to it, to be used as revision tools by the students. One is the increase in level and material and the freedom for students two choose the levels they want to answer rather than the sequential way it currently is and, two is to get support from the lecturer to indicate how beneficial the game is to the students. There is a huge potential for this game in terms of it being a revision tool as, if students could be assessed while having fun, than there is no noticeable reason to why students wouldn't have fun revising.

6.3.5 Lecturer Perspectives

The first lecturer provided a long and informative answer. The lecturer first sentence was "*I have always believed students should be asked to write code in exams, as it's the only way to believe that it is them doing it*". An indication of issues of plagiarism was indicated with the following quote "*The problem with continuous assessment in general is that there is a worry that there could be some other students doing it for them*" and explained that there are ways to go around that by asking students to modify their work and show it in class. In terms of the exams themselves, the lecturer indicates that there are no concerns about plagiarism as "*It's a tried and trusted method is to have the students do it in exam*". The lecturer explained the change of opinion experienced recently in the matter of asking students to write code in a written exam.

"*As a consequence of this question you are asking me, I am actually changing my mind on that now. Even though I can't change the exam paper for their exam this semester,* *for the repeat exam paper I am going to change it and take out any requirement to produce code. I am still happy enough to ask them to write an algorithm in Pseudocode because I think it's important and even demonstrate a kind of basic grasp on Python syntax but I think I have definitely revised my opinion in terms of the importance of their ability to write in Python successfully. I have never thought of it before even though I have been programming for 32 years. I have never seen it in this way before. I know I come from a tradition of writing code on paper.*"

The lecturer later on continued to say "*The kinds of activity I am asking them to do is no difference to the pseudocode really*" and "*If I am asking them to code I am really basically taking marks away from them on the basics of their syntax and things like that*". "*This question and possibly the outcome of this experiment has really changed what I think I am teaching the students as well, fundamentally I am not teaching them how to understand python, I am teaching them how to use Python in IDLE.*"

The second lecturer indicated that the answer given by the students in an exam does not need to be compile able. The following quotation explained the reasons, "*I am looking for key specific parts of the code, that's where I allocate the marks*", "*I am not looking for codes that are compile able*" and "*Under pressure of the exam, students are going to make mistakes*". The lecturer believed that "*Student should sit in exam that has some written component for coding*".

The third lecturer interviewed explained that "*it depends on what you are assessing*". The point of asking for students to write code in a written exam is to see if the students can represent a solution to the problem. Through this method "*you want to see they know they need certain things to solve the problems*". The lecturer indicated that students shouldn't be assessed on missing semicolons. The lecturer indicated that it wouldn't be different for the students, if they were in front of the compiler as the compiler would not tell them what they need to do to solve the problem.

The second and third lecturer shared a lot of similar opinions. They both believe that the codes written by students does not need to be compile able and that they show leniency with syntax errors. They both believe that students should be asked to write code in written exams. The opinions of the first lecturer is at variance to the other two lecturers. The first lecturer remains indecisive about the matter of asking students to write code in written exams. The lecture is considering removing the requirement for students to write code in exams. The lecturer's belief is that students should not be judged by syntax errors and that pseudocode can be a better method of assessing students' knowledge. It can be concluded that the lecturers don't focus on the syntax mistakes but rather if the students understand what steps they need to conduct to solve the problem. The approach on how a student solves a problem is where all lecturers interviewed wanted to focus on.

The second question asked the opinions of the lecturers on students' struggle with equations. It asked whether they provided equations for their students when giving them problems, whether they see the need to do so and how important it is to teach students maths such as Prime Numbers and Fibonacci.

The first lecturer said "*In exams I always gave them an English expression, a near in between pseudo like equation and the equation*" which was similar to what the second lecturer said "*I always give them the formula because it is outside the context of the course"* and *"This is a computer science course not physics course*". The third lecturer responded with *"You would expect them to know the equations",* and *"there is no problem with providing it to them"*. The lecturer still expects students to of capable of understanding and knowing equations taught in first year computer science modules.

The third and final question related to students' feedback indicating that they would have gotten the questions there were assessed on in the written exam correct if they were able to compile and test it. The third lecturer noted that the remark of students saying they would have provided the correct solution to a question if they were in front of a compiler is not rare. The lecturer indicted that compiler doesn't tell the students how to solve or approach the problem. Similar views were made by all the other lecturers. Lecturers' opinion on students' reliance on sample solution and the matter of students copying and pasting from their own past work was asked. All lecturers indicated that they have no problem with students copying codes that the students have made in the past, as long as they understand what they are copying. The first lecturer even when on to saying that this in fact could lead the student in view their past work in a different light.

6.4 Conclusions

This chapter details the preparation required to carry out the experiment to address this project's research questions. The results of the game were analysed separately to each other before comparing the results together to see the effects they have on each other. Observing the results in sets of two levels allowed for a greater understanding of the results as the levels we developed in sets of two. Knowing that there are 12 levels, with each two levels being similar to each other but designed to be increasing in difficulty as the levels increase, three types of information on students' performance were identified. These were students performing equally well in a similar but more difficult level, students performing better in a more difficult level and students making more mistakes in a more difficult level. This chapter was able to identify the reasons for these 3 variations. Students' performance in a written test showed a large difference of improvement after the students' were assessed through a game and this finding was analysed.

7 CONCLUSIONS AND FUTURE WORK

7.1 *Introduction*

The analysis of the results of this research suggests that first year computer science students studying a module in programming were assessed effectively through gamification. The increase of students' knowledge in Python after they were assessed through the game was recorded through the two-stage experimental method devised.

Taking this into consideration, a conclusive summarization of this research is made in this chapter. This is done by bearing in mind the overview of the research and definition of the problems, experimentation and evaluation of results, and its contribution to the body of knowledge involved with the research. Future works and recommendation in the areas of the research as also available in this chapter.

7.2 *Research Overview and Problem Definition*

The main focus of this project was to research and evaluate the use of gamification on assessing students' learning of a first year computer science module in programming. This involved the development of a game. There were many other research themes carried out also. They are as follows;

- To examine educational games in general and in the field of computer science with focus on gamification.
- To investigate current views in game design.
- To review appropriate research related to tools for assessing student learning.
- To identify the tools to be used for developing the game.
- To design experiment to assess student learning before, during and after gamification.
- To record all the information gather through all experimentation and analyse the findings
- To provide recommendations in the areas of this research for future research

7.3 Experimentation and Evaluation

The research focused on the assessment of 22 students studying a first year module in programming. The experiment was divided into two main stages, a pre-game and post-game stage. The pre-game stage focus on evaluating students' current knowledge and identifying the requirements for developing the game to be used to assess students. The pre-game stage assessed students without gamification in order to meet the objectives stated.

The second stage of the experiment was the post-game stage. This stage involved the actual use of the game developed to assess the students' knowledge of the programming language Python. The surveys conducted on students about the game used to assess their knowledge immediately after they were assessed allowed for a direct feedback which contributed answering some of the research questions.

The second stage also included the use of the same none game method of assessing students' knowledge. The same questions were asked in order to identify if the game used for assessing students learning ended up with improving their performance due to the game's ability to provide immediate feedback on student mistakes.

The experiment led to the identification of issues with the current method of assessing students' knowledge in a programming module with a written exam. Interviews conducted in both the pre-game and post-game stages of the experiment, identified the issues and difficulties both students and lecturers face.

7.4 Contributions to Body of Knowledge

The contributions to the body of knowledge made by this research is as follows;

- A two stage method for the development of a game to assess students, which included a method for assessing students' knowledge through a written test.
- A game developed to assess students' knowledge of the syntax and semantics of the Python programming language.
- A game that can identify to students the mistakes they are making, thus allowing for students to learn from their mistakes. This was shown to cause an improvement in the students' performance and knowledge by conducting an

experiment that compared the students' knowledge before and after the game through written tests.

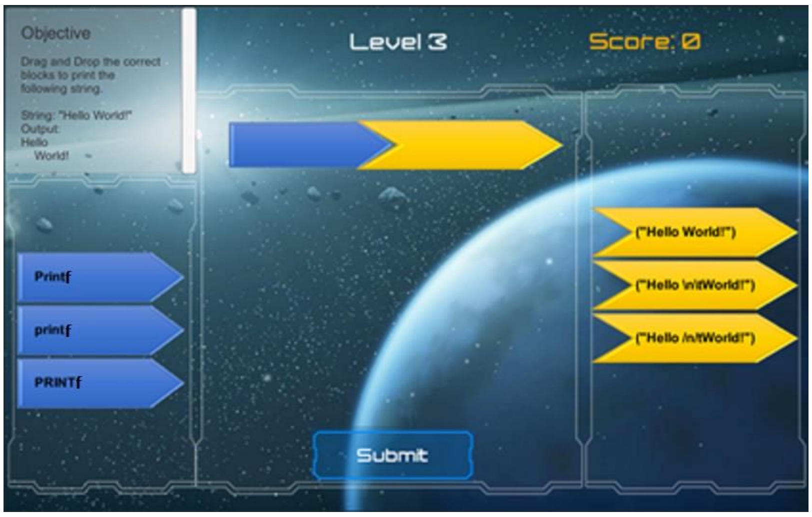
- A game that was developed for assessing students to be also able, to be used as a revision tool for themselves for exam preparation.
- The identification and use of techniques included to assess students in their first programming language with students enjoying the assessment process.
- The revelation of students struggling to write code on paper, unveiling the unexpected mistakes students were making. Indications of students mixing their knowledge of Pseudocode with Python. Indication of students obtaining information of an outdated version of Python online, rather than the version they are taught by their lecturer.

7.5 Future Work and Recommendations

There are a variety of further research that can be conducted using the findings of this project as a source. Recommendations of future research includes;

Expanding the tool to include Pseudocode as well as Python, this might be beneficial to determine if there are differences in the way the students conceptualise the design elements of programming (Pseudocode) and the development elements of programming (Python, in this case).

Testing a similar gamified tool that uses a different programming language, to determine if there are any characteristics of Python that makes it either easier or harder to use in this context than another 3GL programming language (e.g. C, C++, or Java).


Objective

Drag and Drop the correct blocks to print the following string.

String: "Hello World!"
Output:
Hello
    World!

FIGURE 7.1: PYTHON101

Evaluation of the use and effectivity of gamification as a tool for students to use as revision before programming. This could be done using a Control Group approach, where the class was divided in half and the Control Group did not use the tool, and the Experimental Group did use if for revision. The students would be them assessed quantitatively and qualitatively to determine any differences.

Consideration of using games as part of a continuous assessment and designing a suitable marking scheme. Given that the game is presented in several levels, this would be very easy to achieve and might have a different impact on the students if the game is played over time, as opposed to being played in one sitting.

An investigation and comparison of gamification through various game genres. This could include using existing games, like CodeCombat (CodeCombat, 2016), or modifying RPG-style games like Spent (Playspent.org, 2016) or FloodSim (Playgen.com, 2012).

FIGURE 7.2: CODECOMBAT THE GAME

An investigation on the effect of incorporating a multiplayer platform on a gamified assessment tool. This research could identify the benefits and issues that arise with involving several students to a gamified assessment. The findings could be compared to a single player approach in order to identify if incorporating a multiplayer platform is worth the work required to do so.

8 REFERENCES

Abramovich, S., Schunn, C., & Higashi, R. M. (2013). Are badges useful in education?: it depends upon the type of badge and expertise of learner. *Educational Technology Research and Development*, *61*, 217-232.

Adams, E. (2014). *Fundamentals of game design*. Pearson Education.

Adams, E. (2014). *Fundamentals of Puzzle and Casual Game Design*. Pearson Education.

Anderson, T., & Kanuka, H. (1999). Using constructivism in technology-mediated learning: Constructing order out of the chaos in the literature.

Angelo, T. A., & Cross, K. P. (1993). Classroom assessment techniques.

Biggs, J. B., & Collis, K. F. (2014). *Evaluating the quality of learning: The SOLO taxonomy (Structure of the Observed Learning Outcome)*. Academic Press.

Bloom, B. S. (1956). *Taxonomy of Educational Objectives: The Classification of Education Goals. Cognitive Domain. Handbook 1*. Longman

Code.org. (2015). *Every child deserves opportunity*. Retrieved 4 October 2015, from http://code.org

CodeCombat. (2016). *CodeCombat: Learn to Code by Playing a Game*. Retrieved 2 January 2016, from https://codecombat.com

Connolly, T. M., Boyle, E. A., MacArthur, E., Hainey, T., & Boyle, J. M. (2012). A systematic literature review of empirical evidence on computer games and serious games. *Computers & Education*, *59*(2), 661-686.

Deterding, S. (2012). Gamification: designing for motivation. *Interactions, 19*(4), 14-17.

Deterding, S., Dixon, D., Khaled, R., & Nacke, L. (2011, September). From game design elements to gamefulness: defining gamification. In *Proceedings of the 15th International Academic MindTrek Conference: Envisioning Future Media Environments* (pp. 9-15). ACM.

Dickey, M. D. (2006, July). Ninja Looting for instructional design: the design challenges of creating a game-based learning environment. In *ACM SIGGRAPH 2006 Educators program* (p. 17). ACM.

Dondlinger, M. J. (2007). Educational video game design: A review of the literature. *Journal of applied educational technology*, *4*(1), 21-31.

Eastin, M. S., & Griffiths, R. P. (2006). Beyond the shooter game examining presence and hostile outcomes among male game players. *Communication Research*, *33*(6), 448-466.

Ferrera, J. (2012). Playful Design: Creating Game Experiences in Everyday Interfaces. *Rosenfeld Media*.

Fisch, S. M. (2005, June). Making educational computer games educational. In *Proceedings of the 2005 conference on Interaction design and children* (pp. 56-61). ACM.

Fischer, G. (1995, October). Distributed cognition, learning webs, and domain-oriented design environments. In *The first international conference on Computer support for collaborative learning* (pp. 125-129). L. Erlbaum Associates Inc.

Forehand, M. (2010). Bloom's taxonomy. *Emerging perspectives on learning, teaching, and technology*, 41-47.

Gagne, Robert M., Briggs, Leslie, J., Wager, Walter, F. (1985). *Principles of Instructional Design*, Wadsworth, ISBN 0030347572

Gibson, D. (Ed.). (2006). *Games and Simulations in Online Learning: Research and Development Frameworks: Research and Development Frameworks*. IGI Global.

Hofer, B. K., & Pintrich, P. R. (1997). The development of epistemological theories: Beliefs about knowledge and knowing and their relation to learning. *Review of educational research*, *67*(1), 88-140.

Indraprastha, A., & Shinozaki, M. (2009). The investigation on using Unity3D game engine in urban design study. *Journal of ICT Research and Applications*, *3*(1), 1-18.

Johnson, W. L., Vilhjálmsson, H. H., & Marsella, S. (2005, May). Serious games for language learning: How much game, how much AI? In *AIED* (Vol. 125, pp. 306-313).

Jonasson, D. H. (1996). Learning from, learning about, and learning with computing: A rationale for mindtools. *Computers in the Classroom: Mindtools for critical thinking*, 1-22.

Krathwohl, D. R. (2002). A revision of Bloom's taxonomy: An overview. *Theory into practice*, *41*(4), 212-218.

Lee, J., Luchini, K., Michael, B., Norris, C., & Soloway, E. (2004, April). 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* (pp. 1375-1378). ACM.

Merrill, M. D., Drake, L., Lacy, M. J., Pratt, J., & ID2 Research Group. (1996). Reclaiming instructional design. *Educational Technology, 36*(5), 5-7.

Milne, I., & Rowe, G. (2002). Difficulties in learning and teaching programming—views of students and tutors. *Education and Information technologies*, *7*(1), 55-66.

Mislevy, R. J., Behrens, J. T., Dicerbo, K. E., & Levy, R. (2012). Design and discovery in educational assessment: evidence-centered design, psychometrics, and educational data mining. *JEDM-Journal of Educational Data Mining*, *4*(1), 11-48.

Muratet, M., Torguet, P., Jessel, J. P., & Viallet, F. (2009). Towards a serious game to help students learn computer programming. *International Journal of Computer Games Technology*, *2009*, 3.

Nagarajan, P., & Wiselin, J. G. (2010). Online educational system (e-learning). *International Journal of u-and e-Service, Science and Technology, 3* (4), 37-48.

Norton, T. (2013). *Learning C# by Developing Games with Unity 3D*. Packt Publishing Ltd.4

Nte, S., & Stephens, R. (2008). Videogame aesthetics and e-learning: a retro-looking computer game to explain the normal distribution in statistics teaching. In *2nd European Conference on Games-Bases Learning. Barcelona, Spain* (pp. 341-8).

Oak, J. W., & Bae, J. H. (2014). Development of Smart Multiplatform Game App using UNITY3D Engine for CPR Education. *International Journal of Multimedia and Ubiquitous Engineering, 9*(7), 263-268.

O'Donovan, S., Gain, J., & Marais, P. (2013, October). 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* (pp. 242-251). ACM.

Pavlov, I. P. (1897). Lektsii o rabote glavnykh pishchevaritel'nykh zhelez [Lectures on the work of the principal digestive glands]. *St. Petersburg, Russia: Typografiia Minislerstva Putei Soobsheniia*.

Pears, A., Seidman, S., Malmi, L., Mannila, L., Adams, E., Bennedsen, J., & Paterson, J. (2007). A survey of literature on the teaching of introductory programming. *ACM SIGCSE Bulletin*, *39*(4), 204-223.

Playgen.com. (2012). *FloodSim | PlayGen*. Retrieved 2 January 2016, from http://playgen.com/play/floodsim

Playspent.org. (2016). *SPENT*. Retrieved 2 January 2016, from http://playspent.org

Reigeluth, C. M. (1999). The elaboration theory: Guidance for scope and sequence decisions. *Instructional design theories and models: A new paradigm of instructional theory*, *2*, 425-453.

Rossiou, E., & Papadakis, S. (2008). Applying Online Multiplayer Educational Games based on Generic Shells to Enhance Learning of Recursive Algorithms: Students' Preliminary Results. In *Proceedings of the 2nd European Conference on Games Based Learning* (p. 373). Academic Conferences Limited.

Scratch.mit.edu. (2015). *Scratch - Imagine, Program, Share*. Retrieved 4 October 2015, from https://scratch.mit.edu

Siegall, M. (1988). The simplistic five: An integrative framework for teaching motivation. *Journal of Management Education*, *12*(4), 141-143.

Smith, M. K. (1999). The behaviourist orientation to learning. *The encyclopaedia of informal education*.

Susi, T., Johannesson, M., & Backlund, P. (2007). Serious games: An overview.

Swartout, W., & van Lent, M. (2003). Making a game of system design. Communications of the ACM, 46(7), 32-39.

Sweetser, P., & Wyeth, P. (2005). "GameFlow: A Model for Evaluating Player Enjoyment in Games", *Computers in Entertainment (CIE)*, 3(3), 3-8.

Unity3d.com. (2015). *Unity - Game Engine*. Retrieved 6 October 2015, from http://unity3d.com

Vahed, A. (2008). The tooth morphology board game: an innovative strategy in tutoring dental technology learners in combating rote learning. In *Proceedings of the 2nd European Conference on Games Based Learning* (p. 467). Academic Conferences Limited.

Wang, H. (2014). The Comparative Analysis of Micro-reading and Traditional Reading Based on Schema Theory. *International Journal of Social, Management, Economics and Business Engineering*, *8*(6), 1888-1890.

Waraich, A. (2004). Using narrative as a motivating device to teach binary arithmetic and logic gates. *ACM SIGCSE Bulletin*, *36*(3), 97-101.

Watson, J. B., & Rayner, R. (1920). Conditioned emotional reactions. *Journal of experimental Psychology*, *3*(1), 1.

Yip, F. W., & Kwan, A. C. (2006). Online vocabulary games as a tool for teaching and learning English vocabulary. *Educational media international*, *43*(3), 233-249.

Zagal, J. P., Nussbaum, M., & Rosas, R. (2000). A model to support the design of multiplayer games. *Presence: Teleoperators and Virtual Environments*, *9*(5), 448-462.