# Administration Centre for Customizability of Game-Based Learning Material

**140012021 Christopher A Fleming**

University of
St Andrews

16th June 2018

Supervisor: **Ruth Letham**

## 1 Abstract

Extensive research has gone into the area of using games to aid in the educational process. The field extends across different fields of study from History to Computing. Game Based Learning (GBL) techniques increase the motivation and results gained when compared to traditional teaching. However, different demographics can succeed more than others when playing these games, with the majority of games designed for a younger age group. Many GBL systems do not allow creation, modification, or customisation of game levels. Those that do, do not provide a means to plan or structure the games into planned lessons.

The aim of this project is to create an Admin Centre which a teacher can log into, where there will be tools and options to customise the appearance of each level to be tailored to a user, which would increase the beneficiary factors they could obtain from the system.

The Admin Centre will contain a section to register and create accounts for both teachers and students. There will be an area for teachers to create and assign levels to students, customising the levels to suit the student it is assigned to. These levels will be created for the students to play when they log in. A grid based system will be used, which is similar to several existing GBL systems that focuses on teaching Computing, with a new input system developed to be used alongside it.

## 2 Declaration

I declare that the material submitted for assessment is my own work except where credit is explicitly given to others by citation or acknowledgement. This work was performed during the current academic year except where otherwise stated.

The main text of this project report is 16,078 words long, including project specification and plan.

## Contents

## List of Figures

## 3 Introduction

The main aim for this project is to create an Admin Centre which can be used by a teacher to produce customised levels for different students to increase their learning benefits.

The ideologies of Game Based Learning (GBL) is to increase the motivation and engagement levels of users, through using games to educate instead of traditional sources. In 2001 Marc Prensky published a paper that discussed the concept of Digital Game-Based Learning, identifying that there was a generational shift in how people are learning. When talking about GBL he states "it represents one of the first effective and doable means to alter the learning process in a way that appeals to, and excites, people from the 'games generations'"[16]. Within Game Based Learning there are two distinct benefits. In various experiments that have been published, its shown that Game Based Learning can increase the motivation and engagement of participants.

There were concerns over the use of Game Base Learning due to different demographics reacting differently, which in a classroom environment at any stage of the educational system would cause issues with only certain students benefiting from the use of GBL. This could be for example age or gender. A study was carried out using a gender neutral game to see if there was any difference in the learning of girls and boys[13]. In order to conduct this study a gender neutral game was created, instead of the previous games that could be considered male-oriented. The issue then arises that the game might engage both of the demographics equally, but not as well as if they have games that were designed for their particular demographic. However, this is not a solution either, as demographic information does not directly relate to likes and dislikes which could increase the engagement and motivation factors.

The project focuses on creating an environment where the teacher can apply a skin to each level when assigning it to a student, which would alter the visual aspects of a level to suit that student. The structure of the level would be the same, but instead of targeting one type of person, or no types of people, the system would allow the teacher to tailor the learning experience to each individual participant. This would result in an increase in motivation and engagement from each student.

In order to create an admin centre, an educational game must be developed. The game is designed on a grid-based system that teaches students the concept of loops. It teaches the efficiency of loops and how to indent loop structures. The game has different input techniques as well as skins to aid the user's experience through their preferences and current ability.

One of the secondary aims of the project is to create a piece of software that allows a teacher to create new levels for the systems so students of different abilities within the same class could be assigned different levels that would allow them all to benefit from the lesson.

## 4 Objectives

The objectives of this project as set out at the beginning of the project after discussion with the project's supervisor (Sept 2017):

### 4.1 Primary Objectives

P1: Create an admin centre for users to log in to where they can build their own GBL activities.

P2: Create a base game for GBL that will be used in the admin centre.

P3: Create levels of different difficulties to be used in the game.

P4: Have different skins for the game based on target audience of users.

### 4.2 Secondary Objectives

S1: Design a ratings system to give an accurate representation of the difficulty of the game.

S2: Allow users to create their own levels of the game for other users in the admin centre.

S3: Have a automatic moderation system to keep the admin centre clean and fit for use in classes.

As the project continued the scope of it shifted over time. There were Primary objectives that needed to be broken down into multiple objectives due to the size of the problem. Because of this all of the initial Primary goals were met, with one of the secondary goals partially met from the initial list.

## 5 Requirements

This section details the requirements list that was developed in the initial requirements capture phase of the project. They were agreed with the customer during the weekly meetings and were used to assess whether project was running to schedule. The MoSCoW method was used to produce a list of requirements for the system. Must have is something the project guarantees to deliver, Should have is important but vital to the system, Could have is desirable to the system and Won't is something that will be useful for the system to have in future.

| ID | Requirements | Priority | Dependencies |
|---|---|---|---|
| 1 | Teacher **must** be able to log in | High | None |
| 2 | Teacher **must** be able to create Levels | High | None |
| 3 | Teacher **must** be able to assign Student and Tileset to a Level | High | 2 |
| 4 | Teachers **could** be able to bulk assign Students and Tilesets to a Level | Medium | 2 |
| 5 | Students **must** be able to log in | High | None |
| 6 | The system **must** contain a game to play Levels | High | None |
| 7 | Students **should** be able to play levels assigned to them | Medium | 2,3,6 |
| 8 | The system **should** have a professional look and feel | Medium | None |
| 9 | The system **must** have security protocols in place for passwords | High | None |
| 10 | There **should** be a wide range of Tilesets available | Medium | None |
| 11 | There **could** be a difficulty rating that scales to user completion | Low | None |
| 12 | The system **could** have a chat system | Low | None |
| 13 | The system **could** have moderation on the chat system | Low | 12 |
| 14 | The Student **could** be allowed to submit levels | Low | None |
| 15 | There **won't** be a section showing solution as coded example | N/A | None |

Each of these requirements is linked to one or more of the project objectives.

- By implementing requirements 1-4, objective P1 will be satisfied
- By implementing requirement 6, objective P2 will be satisfied

- By implementing requirements 1-4, objective P3 will be satisfied
- By implementing requirement 10, objective P4 will be satisfied
- By implementing requirement 11, objective S1 will be satisfied
- By implementing requirements 1-5, objective S2 will be satisfied
- By implementing requirements 12-13, objective S3 will be satisfied

# 6 Context Survey

This section describes the effectiveness of game-based learning in Computer Science as well as other academic areas. It will also analyse the game-based systems that are currently in use, looking at what the systems are missing. Finally, it will look at how visual programming languages function and how they can be used for this project.

## 6.1 Game-Based Learning

Game-based learning is seen as a way to revolutionise teaching methods, being able to "help to establish dialogue and break social and cultural boundaries"[15]. To see the effectiveness of game-based learning different fields must also be looked at with the core ideas of motivation and engagement which are not limited to the field of Computer Science.

Huizwnga investigated the use of Game-Based learning to teach the history of medieval Amsterdam to secondary school children[9]. The study contained a large participation group of 458 pupils, with half learning using traditional methods and half using the game-based approach, which would remove any unwanted results that could be skewed with a small data set. After the pupils used the system it was found that there was no notable difference in the motivation of the groups, but there was a significant increase in the knowledge gained from pupils who used the game-based system. The mean of the scores using normal techniques showed 36%, but the mean of the scores with the system showed 60%, showing the advantages of the game-based system.

Another example that supports game-based learning in a non computer science environment was a study that look at how a game could be used to teach instructions to a user [7]. It compared teaching instructions for a quiz with a traditional method and using game-based learning. The traditional method was the more successful method in the first experiment. In the second experiment, constructive feedback on the user's performance was added to both methods and the game-based method became the more successful option. The feedback to show the user is carrying out the task correctly suits the game-based learning system.

One of the types of games that is used to teach Computer science is moving around a grid-based system. Hewijin describes in her paper a strategy game where the user must move around collecting resources and returning to their castle. This type of thinking is similar to that of the user in this project's game. "The simulation environment allows the students to monitor the execution of the game so that they can understand how to adjust their strategies to improve the game results. Therefore, the students have an increased motivation to modify their game strategies by refining their source programs".[10] By showing where a user has gone wrong, they increase the motivation in the user to improve their solution.

Tobias gave a summation of the empirical evidence supporting the effectiveness of game-based learning. He covers the analysis of a variety of topics, "external tasks, enhancing cognitive processes, guidance and animated agents, playing time and integration with curricular objectives, effects on game players, attitudes toward games, cost-effectiveness, and finally, the use of games for evaluation"[**18**]. He individually summarises each of the types throughout his analysis, showing how each can be beneficial. At the end he states "The research reviewed above indicates that games hold promise as instructional delivery systems".

## 6.2 Game-Based Learning Systems

There are a variety of game-based systems currently available for teachers and students to use. Robozzle[12] is a puzzle game that teaches conditional statements and logic to users. The game is targeted at an older audience, with the high complexity of problems (see figure 1).

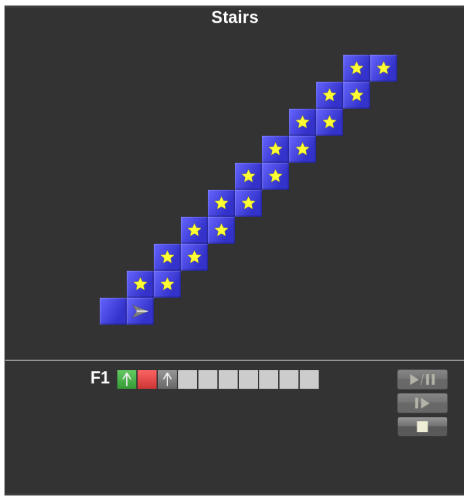
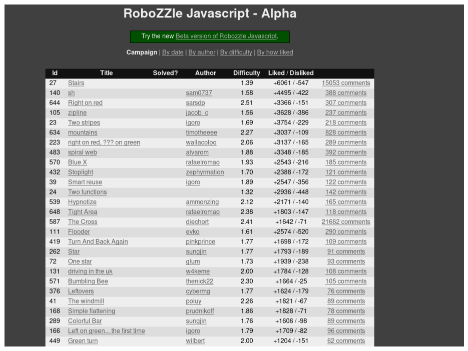


Figure 1: Example of level in Robozzle

The graphics are sharp and generally dark, leading itself to an older audience. The main menu in Robozzle is hard to navigate with no instructions on how to complete the levels, and the input is not very intuitive to use (see figure 2).



RoboZZle Javascript - Alpha

Try the new Beta version of Robozzle Javascript.

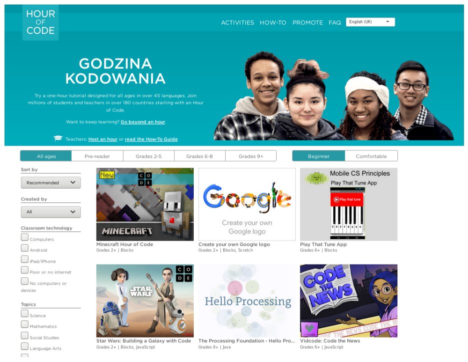
Campaign | By date | By author | By difficulty | By how liked

| Id | Title | Solved? | Author | Difficulty | Liked / Disliked | |
|---|---|---|---|---|---|---|
| 27 | Stairs | | | 1.39 | +6061 / -547 | 15053 comments |
| 140 | sh | | sam0737 | 1.58 | +4495 / -422 | 388 comments |
| 644 | Right on red | | sarsdp | 2.51 | +3366 / -151 | 307 comments |
| 105 | zipline | | jacob_c | 1.56 | +3628 / -386 | 237 comments |
| 23 | Two stripes | | igoro | 1.69 | +3754 / -229 | 218 comments |
| 634 | mountains | | timotheeee | 2.27 | +3037 / -109 | 828 comments |
| 223 | right on red, ??? on green | | wallacoloo | 2.06 | +3137 / -165 | 289 comments |
| 483 | spiral web | | alvarom | 1.88 | +3348 / -185 | 392 comments |
| 570 | Blue X | | rafaelromao | 1.93 | +2543 / -216 | 185 comments |
| 432 | Stoplight | | zephyrmation | 1.70 | +2388 / -172 | 121 comments |
| 39 | Smart reuse | | igoro | 1.89 | +2547 / -356 | 122 comments |
| 24 | Two functions | | | 1.32 | +2936 / -448 | 142 comments |
| 539 | Hypnotize | | ammonzing | 2.12 | +2171 / -140 | 165 comments |
| 648 | Tight Area | | rafaelromao | 2.38 | +1803 / -147 | 118 comments |
| 587 | The Cross | | diechort | 2.41 | +1642 / -71 | 21662 comments |
| 111 | Flooder | | evko | 1.61 | +2574 / -520 | 290 comments |
| 419 | Turn And Back Again | | pinkprince | 1.77 | +1698 / -172 | 109 comments |
| 262 | Star | | sungjin | 1.77 | +1793 / -189 | 91 comments |
| 72 | One star | | glum | 1.73 | +1939 / -238 | 93 comments |
| 131 | driving in the uk | | w4keme | 2.00 | +1784 / -128 | 108 comments |
| 571 | Bumbling Bee | | thenick22 | 2.30 | +1664 / -25 | 105 comments |
| 376 | Leftovers | | cybermg | 1.77 | +1624 / -179 | 76 comments |
| 41 | The windmill | | poiuy | 2.26 | +1821 / -67 | 89 comments |
| 168 | Simple flattening | | prudnikoff | 1.86 | +1828 / -71 | 78 comments |
| 289 | Colorful Bar | | sungjin | 1.76 | +1606 / -98 | 89 comments |
| 166 | Left on green... the first time | | igoro | 1.79 | +1709 / -82 | 96 comments |
| 449 | Green turn | | wilbert | 2.00 | +1204 / -151 | 62 comments |

Figure 2: Main menu of Robozzle

Robozzle contains a comment section that is meant to be used by users to discuss level solutions, but does not seem to be moderated and is open to spam and language unsuitable for young children. Robozzle does allow you to create your own level which will be added to a central area where all the levels are contained, you can filter these by author, difficulty and how liked a level is, but there is no feature for setting up a lesson, which makes it more difficult to lead a student through a lesson using the system.

Another example of a site that provides game-based learning environments is Hour of Code[14], which is a collection of different games that teach a wide variety of different topics (see figure 3).



Try a one-hour tutorial designed for all ages in over 45 languages. Join millions of students and teachers in over 180 countries starting with an Hour of Code.

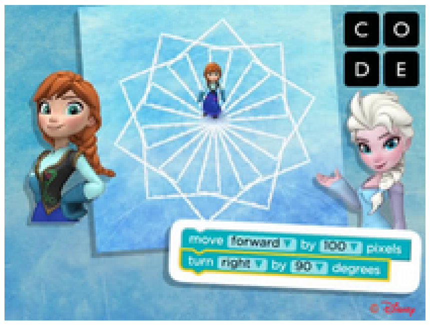
Want to keep learning? Go beyond an hour

Teachers: Host an hour or read the How-To Guide

Figure 3: Main menu of Hour of Code

Unlike Robozzle, Hour of code offers lots of short tutorials for users to get used to the environment. The issues that lie with Hour of Code is that it targets demographics with certain games. With Robozzle targeted at older users, Hour of Code tends to be targeted at younger audience with little experience. This limits the the system to be used with younger classes and with users who have no current experience. The styling of the games means the system excludes older users who are at a beginners level and could learn well from the functionality of the game, but will not due to the appearance of the game.

Hour of Code also has games that can be targeted at genders, such as a game that is based on the movie Frozen (see figure 4).



Code with Anna and Elsa

Grades 2+ | Blocks

Figure 4: Example of a game that can be found on Hour of Code.

The game has functionality that could greatly benefit users, but the game could alienate some of them due to its appearance. Hour of Code does not offer features for creating levels, due to it being a collection of different games. There is no central area for a teacher to create a lesson for a student.

Robocode is a programming game that teaches users how to program, which is a large part of Computer Science. In the Robocode system they state "there are few good Problem-based Learning (PBL) problems available to assist new practitioners with implementation" [11]. They discuss the fact that learning becomes easier when there is a good problem for the user to solve inside the game. They use a competition where users compete against each other to gain this affect. This is a good solution, but the system they have created is not similar to project, meaning their methods of using competition is not applicable.

The concept of the project is to have an admin centre where the teacher can create a lesson plan for the student to work through. The teacher can set different lessons to different students so they have a lesson plan that suits their ability. With the teacher also able to change the styling of the page to suit each student, they will be engaged and alienated.

## 6.3 Visual Programming Languages

A Visual Programming Langauge (VPL) is a programming language contains graphical elements which are used to create a program. "By manipulating graphical elements, using visual expressions, spatial arrangements and relations of text and graphic symbols" [2] a user can program with the traditional methods. The interactions for creating the program involve dragging and dropping the graphical elements into certain positions that will dictate how they preform. Many VPL have the idea of boxes and arrow that give a data flow and show relations between objects.

An example of a VPL that is used as an educational tool for children is Scratch, which is a tool that allows you to program "interactive stories, games and animations" [21]. In Scratch blocks are used to represent code, which children will drag and drop into the block area to build up a script. They can then run the script, with the results being shown on the stage (see figure 5).

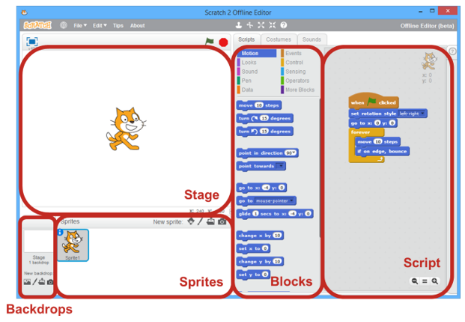


Figure 5: Screen shot showing the layout of Scratch.

This style of layout is common amongst similar systems. Google and the BBC both produce systems called Blockly[8] and BBC micro:bits[3] (see figure 6) respectively that have the same type of layout, with there being an input area for the objects representing code and a stage area showing the results of running the code.

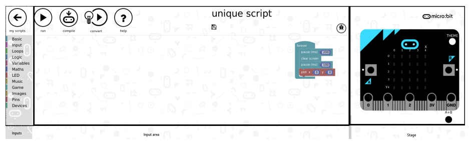


Figure 6: Screen shot showing the layout of BBC Microbit.

Scratch is becoming more widely used in primary schools as an introductory path into Computer Science. However, Scratch does not use game-based learning, making it different from this project. In Scratch there is a teachers account where they can add students accounts and manage student projects, which is aided through a strong developer community. Concepts from Scratch are used in the system for the project, interleaving it with game-based learning concepts.

## 7 Software Engineering Process

There are many different project management and software development processes, each with their own benefits and drawbacks. For this project, a largely agile approach was taken, with frequent customer involvement, feature prioritisation, and the ability to adjust scope[6].

This project was conducted over a 9 month period from September to June. In the initial stages of the project over the first month, the majority of the work was defining the details of the project. The project was very open ended, with there not being an example of the proposed system. To make sure the best design decisions could be taken, extensive research was conducted into the field of game-based learning and the current systems available.

After the initial research phase of the project, the objectives and description of the project were defined. The information from here was complied into a DOER, which also state if there was any ethical approval required for the project. Due to the time scale of the project, tasks would be broken down into smaller tasks that could be completed within a week. This is sprinting, a practise used in agile development, primarily in Scrum[18] to keep efficiency high and stop other work becoming a distraction to the project.

Towards the end of the first semester of the project, myself and my supervisor realised some of the secondary objectives were large pieces of work in themselves and would be hard to complete. Therefore the Primary objectives were targeted instead and additional features would then be added around them later.

Throughout the time spent on the project, weekly meetings were conducted with my supervisor/customer, to initiate the start of a sprint. These meetings would first act as a retrospective, going over the work that had been completed since the last meeting, discussing improvements and alternative solutions to the method used. Then the rest of the time of the meeting would be used as a brain storming session where ideas would be discussed for the next week's work. If I was not aware of how a possible method functioned, my supervisor would introduce me to the concept before allowing me to go experiment with the method in the problem space.

That iterative process of meeting every week helped establish the strategy of breaking down tasks into smaller goals. The benefit of this process being that progress can be seen from a early stage in the project.

It was discussed what issues were more important to the project, having the admin centre and database functioning better and having a more professional aesthetic or whether to create more skins for the game. Due to the large time constraints in making a new skin for the game, after five were created it was deemed that the concept had been proven and creating additional ones would not add to the project in the same way, whereas developing the admin centre more would.

For the implementation of the project version control was used to back up the code of the project, also allowing the project to be reverted to the previous week if it had skewed in direction from the previous meeting.

For the coding section of the project, each page had the entirety of the code in its own file to allow the rapid development without the worry of affecting previously completed functionality. After the functionality of a page was completed, refactoring was performed on the page to improve the codes, reusability and consistency. This process is called rapid deployment and is used to "demonstrate system aspects critical to the user"[17]. In a research project with a time limit, similar to this project, it is important to show the prove of concept as early as possible.

## 8 Ethics

When the project was first proposed there was a possibility of empirical analysis being carried out on the system, but due to the fact of requiring children to participate in the study, it was deemed there was not enough time to get get approval and conduct the study.

There were no other ethical concerns with the project that do not appear in the ethics form submitted earlier in the project. The form can be seen in the appendix.

# 9 Design

## 9.1 System Features and Flow

### 9.1.1 Activities and System Flow

After the Objectives and Requirements have been produced a clear flow for the system is needed. By designing this it would become clearer how the implementation would be carried out. When designing a system it is useful to know the different activities that could be completed when using the system, the objectives and requirement lists were used to help produce the following list of activities.

- Create Account
- Teacher assigns a level
- Teacher creates a level
- Student participates in lesson

An activity diagram is UML diagram that describes the dynamic aspects of a system [1]. The diagrams produced would be useful when programming the flow of data through the system.

Creating an account is a straight forward process, where a new user must enter a unique username along with a password (see figure 7). The main implementation challenge will be writing the information to the database.

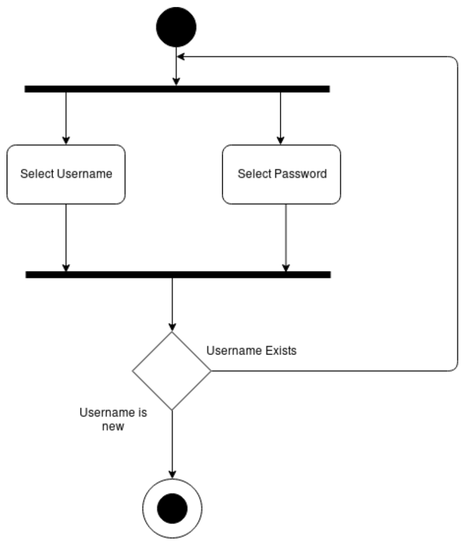


Figure 7: Activity Diagram for Registering Student/Teacher

There are two ways ways for a teacher to reach the area for assigning a level (see figure 8). The two paths that can be followed are assigning a preexisting level or creating a new level. The forked symbol on the create Level symbol represents a subactivity. Each subactivity will have its own diagram. Both routes through the activity, end at the same location, so handling the information up to this point will have to be carefully looked at.

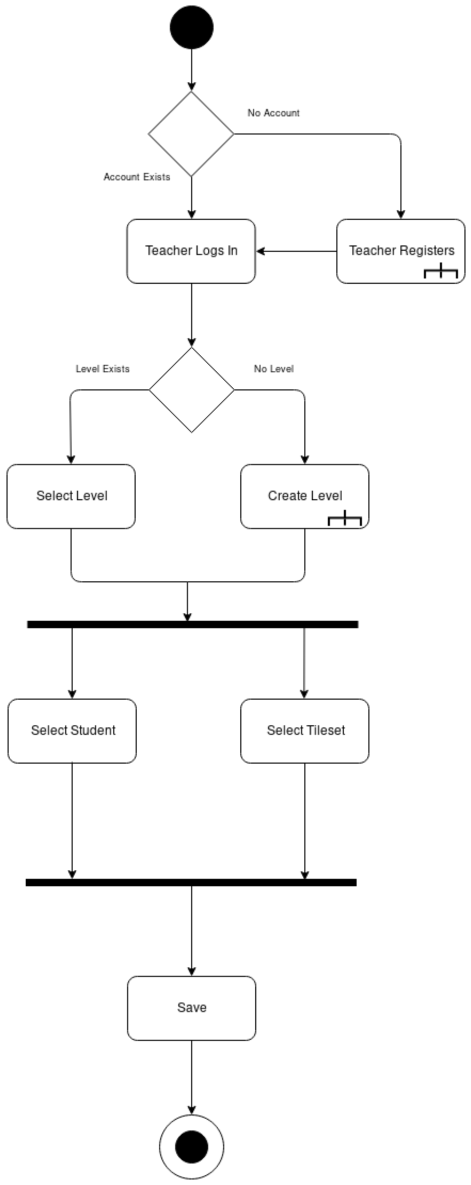


Figure 8: Activity Diagram for teacher assigning level

When a teacher wishes to create a level, the decision at the beginning (see figure 9), on whether the level name exists or not is a consideration that will be taken into account when the implementation process begins. There are two main stages for the teacher entering information so when it comes to developing this stage, two separate areas are required.

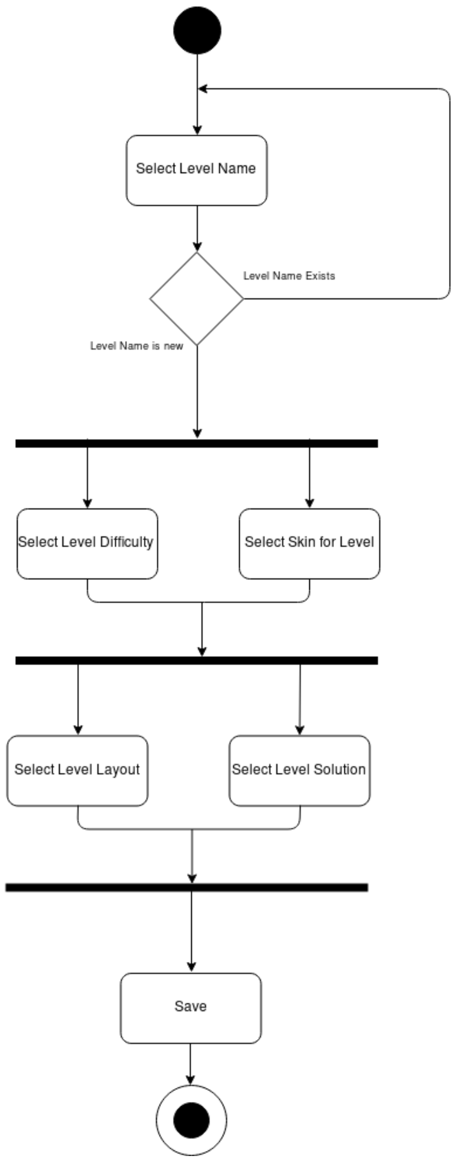


Figure 9: Activity Diagram for teacher creating a level

The start of the activity for a student playing a game (see figure 10), starts with the same login procedure as figure 7. After the student has successfully logged in they can enter a solution to the game and run it. If the solution is incorrect they must enter a new solution. If the solution was correct and the user does not wish to reset the level, a new level is loaded or they reach the end of their lesson. At the end of their lessons if they have improved any of their solutions, it will be stored.

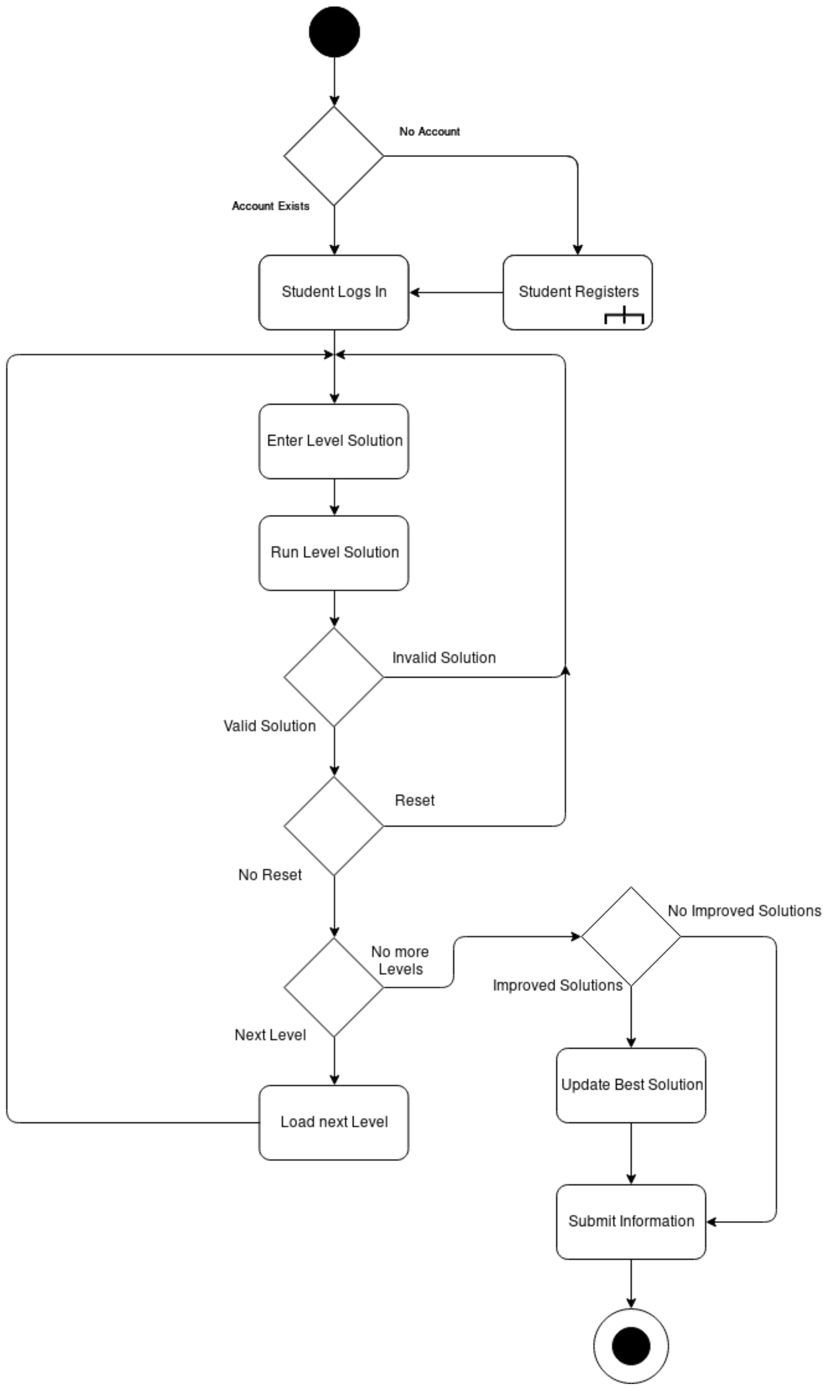


Figure 10: Activity Diagram for student playing the game

Once the flow of activities had been established, consideration could be given to how this information would be structured and presented to the user and this would be in the form of a System map. Instead of going through the high level actions and decisions the user would make while using the system, it would show how all the different pages in the system would link together (see figure 11).

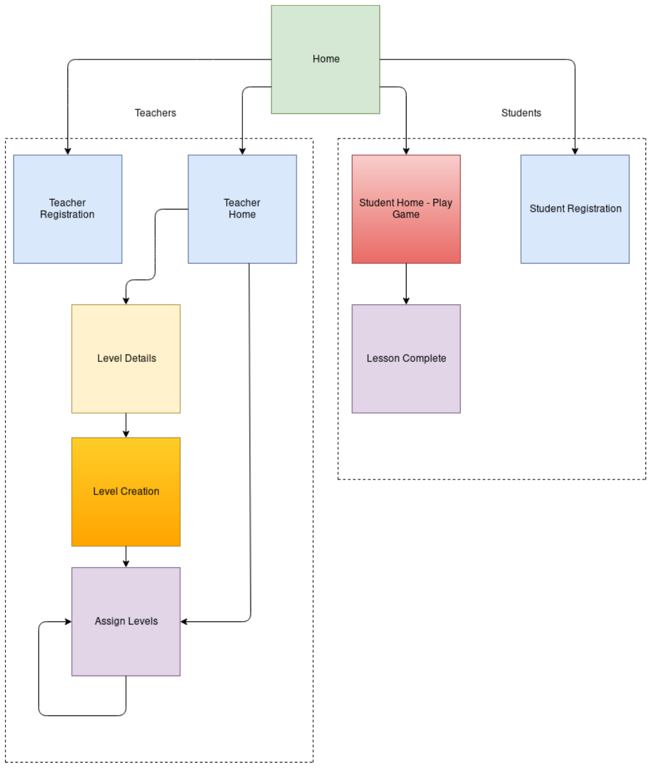


Figure 11: Screenshot of systems map diagram.

A System map diagram is used to find out what a client wants from a system. It aids in gaining functional requirements as it highlights the scope of the problem. After creating the System map diagram it was clear that the project would be in two sections, one for the teacher and one for the student. This was backed up by the activity diagrams that were also drawn. The System map diagram also showed that there could be multiple routes through the system to get to the same location. This would have to be considered when handling values during the implementation.

## 9.2 Game Design

As seen in the Context Survey cross, previous educational games that were centred around teaching programming used a grid based system where inputs would move something around it. As the main objective of the project was to create an admin centre around a game, it was decided that creating a whole new concept for a game would not be within the scope of the project. So inspiration was taken from Robozzle and Hour of Code to create a grid based system chosen for the game.

The design of the game itself fell into two sections: Game Map Components and Instruction sets.

The Game Map Components would be for describing the different tiles that could be selected and the Instruction set, describing the rules around the user input of the game.

### 9.2.1 Game Map Components

- Character: The character tile would be moved through the level towards the goal.
- Goal: The tile that the user would try to get their character to. Once the character reaches the goal tile, the user completes the level.
- Terrain: The character is able to move over this tile.
- Obstacle: The character is unable to move over this tile.

As the game is a single player game, only one character tile can be in the game at one time. If a character attempts to step over an obstacle tile then the level is failed and the level is reset. As the user can see where the character has failed the level, resetting the level allows the student to attempt the level again.

### 9.2.2 Instruction sets

By creating a centre for customising different levels to suit different users one of the aspects that could be customised would be the method a user could enter inputs with, from simple input techniques to more advanced ones. Designing lots of different input techniques would again be very time consuming and detract from creating the admin centre. It was decided that creating one new input technique would be a useful concept and experiment. However, the way the input is parsed means another input technique could be placed in without changing the rest of the system.

Most of the concepts online were simplistic, so a more complex input that introduced the concept of indention was designed. The input system teaches loop, indentations and iterations. This is done using the following symbols;

- Up Arrow: Moves the character forward one tile in the direction they are facing.
- Clockwise Rotation: Rotates the direction of the character 90 degrees clockwise
- Anti-Clockwise Rotation: Rotates the direction of the character 90 degrees anti-clockwise
- Start: Designates the start of a loop
- End: Designates the end of a loop
- Numbers: There are nine separate number symbols each representing an iteration value.

The following rules are put in place in the implementation stage to help users learn the mentioned concepts.

- The up arrow and rotation symbols can be placed anywhere in the input area.
- The start symbol can also be placed anywhere.
- The end symbol must be placed in the same column as a start symbol, that does not already have an end symbol placed in line with it.
- A iteration number can only be placed one to the right of a start symbol
- Any symbols that appear inside a loop must be indented one column to the right of the iteration number.

To calculate the best solution of the game the number of up arrows and rotation symbols are counted in the user's solution. This would teach students that loops are the better and more efficient solutions.

A solution that uses eight up arrows to reach its solution will have a score of eight. A solution that uses a loop with 8 iterations, which contains a single up arrow will be given a score of one. So the new system teaches how to create loops and how they can produce a more efficient solution.

## 9.3 System aesthetics

Before designing the individual elements on each page it was important to create what the overall feel of each page would be and to do this a Wireframe diagram was used. This diagram represents the skeletal framework of a page, showing where elements will appear on the page. Two Wireframe diagrams were produced as a mock up of what the pages would look like.

In figure 12 the student game area is represented and so is the level creation area the teacher will use. At the very top of the page there would be a home button in the top right hand side, which would link the user back to the home page of the system. There then would be four clear sections to the page. The furthest left block is called the *tile pallet*. It would contain the selection of tiles that would be used to create the level. This is similar to the how the inputs are displayed on the BBC Micro:bit (see figure 6). The next block would be the *map area*, which would house the the Tilemap that would display the game and the area the teacher would place tiles in to create a map. The *map area* would act how the stage acts in Scratch (see figure 5). The *input pallet* would contain the input objects the student would drag and drop to enter this input to the game. The final box would be the *solution area* where the user would place the inputs from the *input pallet*. The idea of dragging and dropping graphics was taken from the Visual Programming technique in section 6.3. By taking inspiration from other current systems, the project would be able to fit into the wider ecosystem of programming education.

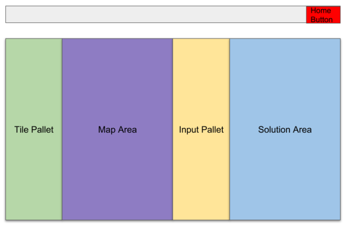


Figure 12: Wireframe diagram for level creation and game play area

The Wireframe diagram in figure 13 shows the page where the teacher can assign levels to different students. The three boxes in the middle of the diagram represent the table that will show the current students who are assigned to a level, also showing the tileset they use and whether they have solved the level or not. The table will appear black if there are no students currently assigned to it. At the bottom of the diagram there are two boxes that represent the menus that will have student and tilesets selected from to add to the table.

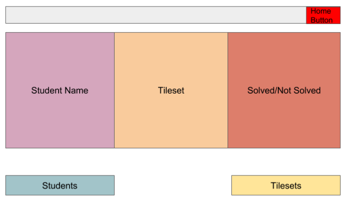


Figure 13: Screenshot of level being created.

After producing the Wireframe diagrams, each page was designed in greater detail using Ben Shneiderman eight golden rules in user interface design[19]. These rules were derived heuristically from experience and are applicable across most systems, therefore it would be useful for designing the project.

When designing the admin centre a professional feel is required throughout the system, where the same themes will be used. This would be to follow the first golden rule of consistency. Schneiderman first rule states that similar terminology should be used in menus and for navigation. To apply that to the system, drop down menus were used throughout the system because they hide the information in a smaller space when it is not interacted with, which would keep simplistic styling of the page in tact. Also, a drop down menu works best when there are between 15 and 25 items. For a single teacher this would be around the number of students and levels in these drop-down menus. The menus were also all given the same colouring and styling, and the colour chosen was the same colour as the blue from the University badge and was obtained using an eyedropper tool. When hovered over all the menus would turn purple, as would any button in the system to again keep a consistency to the design of the website.

Another feature that is deemed to be useful from the rules is Enabling frequent users to use shortcuts. In order to allow users to navigate back to the home page of the system quicker than clicking back in their browser, a home button was placed in the top right of pages, which would return he user to the home page when clicked on. This shortcut allows users with more experience to navigate through the system quicker.

By offering informative feedback Schneiderman says there should be feedback relating to the size of an action in the system. When the user completes a level they are given an alert showing this, and the same occurs when their solution is unsuccessful. These are small alerts that just need clicked off to remove them. When a teacher creates a level, a student finishes a lesson or an account is created a larger notification is given, where a whole page is used to tell the user. When a student completes their lesson or a teacher finishes creating a level and assigning it, they are given congratulations message to show they have reached the end of the activity they were completing. This is another golden rule, "Design dialog to yield closure", where it is important to give the user informative feedback which will give the user the satisfaction of accomplishment.

It is important in the system to permit easy reversal of actions which then allow the user to explore the system more. When creating a level a teacher is placing different tiles from the tile pallet into the map area they are experimenting what they wish the level to look like. Once a tile is placed it can be written over easily with another tile. Another example of this rule being applied to the system is the reset function in the game. A user can reset a level even after they reach a goal. This allows the student to correct their solution if they wish too.

Another of Shneiderman's rules is to support internal locus of control. This concept is that the operators of an interface feel they are in charge of the system and the system responds to their actions, not the other way around. In the teacher section of the web page, the teacher can either create a new level or assign a previous level, they are not forced into a choice leaving them in control of their actions. In the student side of the system they are given the option whether to go to the next level or stay to improve their solution, they are also allowed to solve the level using their own choices from the input pallet.

Reducing the short-term memory load is also a rule to consider. He Shneiderman that "The limitation of human information processing in short-term memory requires that displays be kept simple"[20]. The entire of the system, especially the teacher side of the system, was given a simple design that also allowed it to obtain a professional feel. The University logo was placed in the admin centre to give depth to the page and not leave the pages as empty. The colour scheme used throughout the centre was the same colour to give a consistent flow to the pages. A white background was used to stop the page looking cluttered with unnecessary pictures distracting the teacher from the system's menus. The colour scheme for the home icon was black and white, which allowed it to appear clearly to the user on the white background

## 9.4 Database Design

The Activity Diagrams, user System Maps, and Game Design combined to give a picture of what information needed to be stored and highlighted how that information needed to relate. The information collected shows highly structured data that is relatively static, which is suited to a relational database. Primary and foreign keys will be chosen in a way that would not allow information to be altered by an action in another table.

With there being two distinct sections to the project, the teacher and student area, there would be a table for both the student and for the teacher, each containing a username and a password. The primary key for both of these tables would be their respective username fields. This was to stop users having the same name or they could end up with levels assigned to them that they were not supposed to have.

The next information required to be stored would be the information required for a level to be stored. The name of the level would be the primary key as it would be challenging to assign levels if they shared names. The best solution would be stored, which would be a 1D array turned into a string. It would also contain the difficultly of the level as an integer. A default skin is stored so there will always be a skin that can be used to display a level. The teacher who created the level will also be in the table, in order to help display all the levels created by the teacher. Finally the layout of the level is stored. The layout of the level would be where each type of tile is placed and the skin would be how this is visually represented to the user. It is stored as a blob and is a textual representation of a 2D array.

It was decided that the studentLevel table would have the student's name and the level's name stored as primary keys that would reference the student and level table. This allows the student to be linked to a level and have it personalised to them, to gain the necessary information for loading the level information about the layout and then how to personalise it to the student in the table.

The storage for the layout in the implementation would be a 2D array, so several options were considered using Entity–relationship (ER) diagrams to discover the best solution.In the first ER diagram (figure 14) the layout is a single field in the table. The 2D array would be stored as a string in the table.

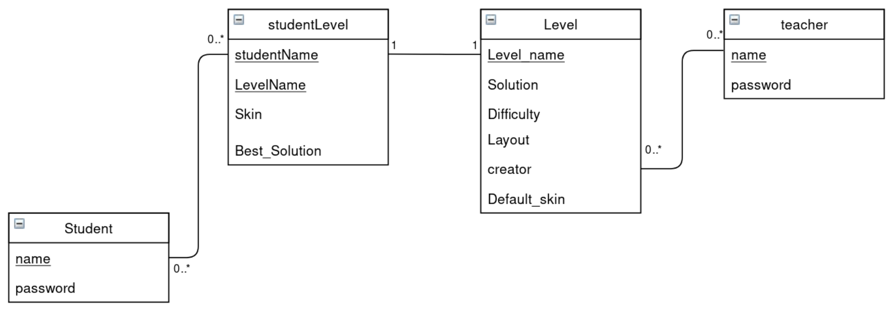


Figure 14: Entity–relationship diagram with layout as a field

Normally in databases each entry in a field should be atomic, i.e. the values should be in the simplest form, using joins and other operations to combine values. By having a single field for layout which is representing a complex object (the map), this design is not the best practise, so other options were investigated.

In figure 15 the layout is represented with a table of Rows. A layout is made up an nXm grid, so there are m columns in the table as fields and there is a 1 to n relationship from the level table to the row table. As mentioned above the table's fields should be atomic in nature and by having the row table as its own table you can allow each tile to be an entry on its own. There are some issues with this design, one being there can be a lot of null values in the table, which would lead to the code handling the database to require special handling of data. The number of columns in the layout, (i.e the size of the map) would be very hard to change as every query to the Row table (including select, update  insert) would need to be updated as well as potentially the code and logic related to those queries.

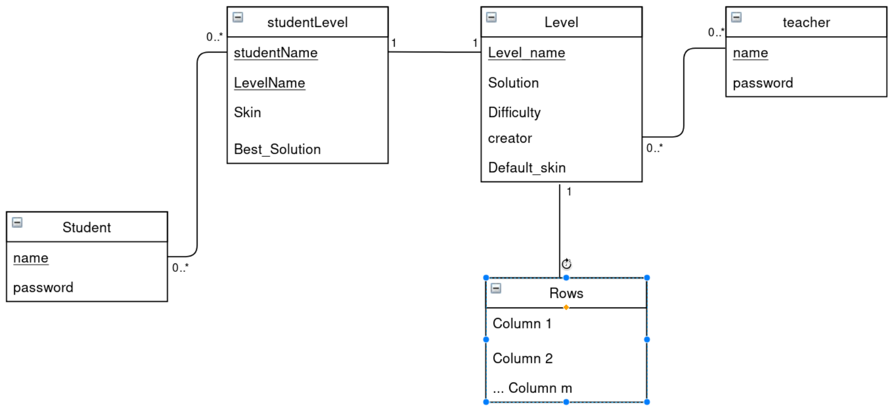


Figure 15: Entity–relationship diagram showing Row as a table

In figure 16 the tiles that make up the map are stored in a separate table. This provides atomicity as a tile is an object in itself and each tile should be stored in the database. Fewer null values would appear in this database compared to the second ER diagram, but it would be harder to work with blank values in the layout, as it would be hard to tell the position of the missing tile in the layout. There would be a massive increase in the number of rows in total in the database making a database based on this design potentially very slow to work with.

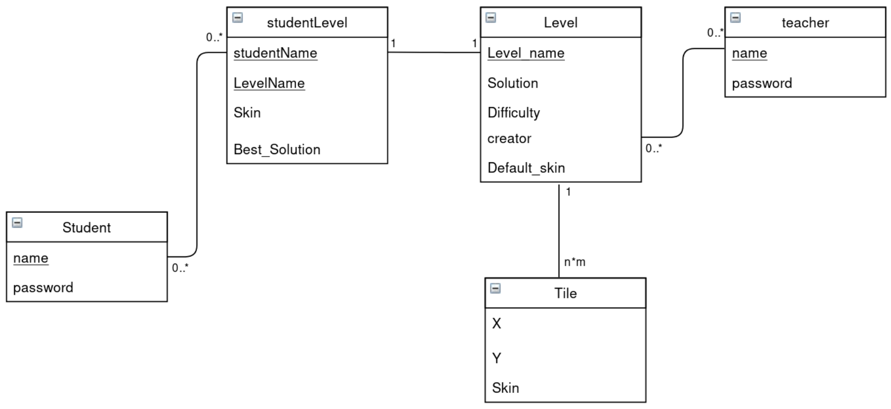


Figure 16: Entity–relationship diagram with layout as a field

A join is an expensive procedure and can slow a system down. There are very few joins in the first ER diagram. The second ER diagram has an increase in joins, then there is a massive increase in the number of joins in the third diagram. These could cause the second and third options to run much slower than the first option.

The first ER diagram was chosen as the structure of the database. Even though it will store the layout as a string, this is much easier to work with than many null values. It also makes it easier to handle missing values and means there does not need to be any alterations to the database if there is a change to the map. In addition, this structure will be quicker to process and access. All of these benefits makes it worth using a string to store layout.

### 9.4.1 Relational Schema

From the ER diagram, a relational schema could be derived to show how each field would store its information.

*Level* = (<u>Level_name</u>: varchar(20), Difficulty: integer, layout: blob, creator*: varchar(20), default_skin: varchar(20))

*student* = (<u>name</u>*: varchar(25), password: varchar(20))

*studentLevel* = (<u>student_name</u>: varchar(25), <u>Levelname</u>*: varchar(20), skin: varchar(20), best$_solution : varchar(20), best\_solution : varchar(20)$)

$teacher = (\underline{name}* : varchar(20), password : varchar(20))$

# 10 Implementation

## 10.1 Technologies Used

For the project, it was necessary for many students to be able to easily log into the system at the same time, and also to deploy the project in schools. In order for this to be possible, a web based solution was deemed the most appropriate solution. Despite Java being my most proficient language, due to web technologies being more inline with the project it was chosen. For this solution a client server architecture would be required (see figure 17).

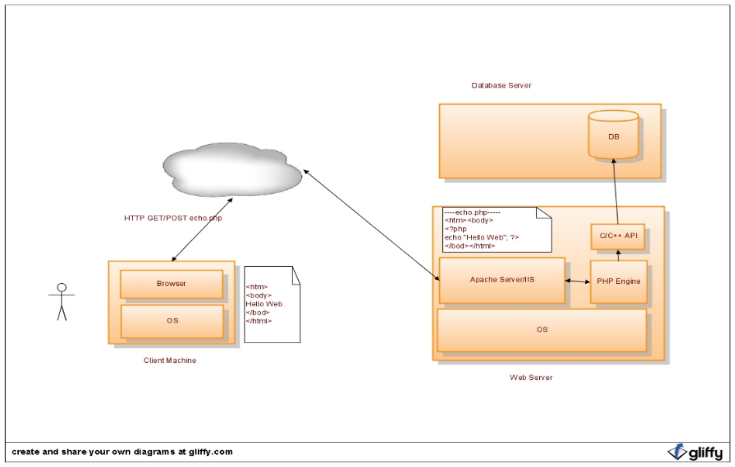


Figure 17: Diagram showing a client server architecture for PHP[23]

The client side of a system is viewed with a web browser, using HTML, CSS and javascript to display content on the page. The client makes a request to the web server, which will then route the request to the correct server side script. This is handled by Java, PHP, Python etc, which can access the database writing changes. The result of the server side script is then sent back to the client side, which then visually shows the user the result.

### 10.1.1 Database

There are different types of databases available to use including hierarchical, Object-oriented (OO) and relational. A hierarhcical database is a fast and simple with the access and update functions being quick. Records contain information about groups of parent/child relationships. There will not be any parent child relationships in the system so was not the most appropriate choice to use.

OO databases provide full feature database programming capability, while keeping the compatibility of the languages its built upon. As the database and program handling the data have the same methods for handling data, the system is consistent. However they can be more expensive to develop and there are less tools available to aid in developing with it.

A relational database is made up of tables which have fields(columns) and records (rows). Each table has a key filed that uniquely indicate each row and the key fields can be used to combine table together. Relational databases force data to be stored in a strict structure, which I had prior experience with. The other options did not offer anything that would make them beneficial over relational. MariaDB, a community-developed fork of the MySQL relational database, is supported on the accounts within the school of Computer Science was chosen.

#### 10.1.2 Server-Side

There are various server side languages to choose from. General programming languages such as Java and Python can be used. These can be used with a Common Gateway Interface (CGI) which is an external gateway for programs to interface with information servers. General purpose languages generally have server-side scripting packages that aid the user in writing the script. Despite having a familiarity with Java the server-side scripting language PHP was chosen as I have some experience using it in this scenario. PHP also has the advantage of being able to easily create pages for the system.[23]

#### 10.1.3 Client-Side

The document structure of the web pages would be achieved using HTML5, with CSS2 adding the styling to the web page. All the dynamic content of the page is produced using JavaScript, along with Phaser which is a 2D game framework for HTML5 and is written using JavaScript[4].

Phaser works by first starting with a Phaser Game object, which is the heart of the program, allowing access to common functions and other objects. Within the Game object the size of the Phaser Canvas is defined. The Canvas is what all objects are displayed on. The Game object also defines the functions that create the structure of the program; preload, create and update function. The preload function will happen when the code first runs, and will load in all the assets that will be used in the program. The create function defines and creates the tile maps, images and other objects that will be used. Finally the update function runs continuously through out the running of the program, updating the state of the game and the object.

## 10.2 TileMaps

A technique was needed to display this grid. Using a 2D array was investigated, but involved a lot more processing and had greater limitations as there would be no built-in functions to aid in development. After Phaser had been chosen as the framework for the project, the different features of Phaser were looked at. Phaser has an object called a Tilemap. A Tilemap is made up of Tiles, and each Tile has a size and position. A Tilemap is generated from either a JSON file or a CSV file, so in order to create one of these files without coding the entire file by hand an additional piece of software was needed.

A program called Tiled was used (see figure 18). The process for creating a Tilemap using Tiled was that after setting the size of the Tilemap and the size of the tiles, set the images that would be used to create the Tiles. The tool that was used to select sections of an image was the same size as the Tiles. A design decision was taken to make the images that make up the Tiles from scratch. This would mean the system could be made public for use after the project easier and that images could be made the same size as the tiles in the Tilemap, which would stop any cropping or scaling affecting the images.

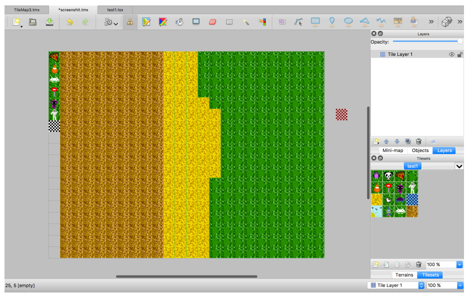


Figure 18: Screen shot of *Tiled*

The images for the Tilemap were created using a website called Piskel. The website is used to create pixel artwork with the images being able to be exported in a png format. Multiple images that would be individually used are combined within the one png image. The reason for this is that once the images have been used to create a Tilemap it needs to be embedded into the Tilemap. Once an image is embedded, it appears inside the JSON file. This links the JSON file, the Tilemap and the png file together to create the Tilemap to be used when implementing the Tilemap for the web page.

The concept of the system was to have different skins to change the way the game is displayed to aid in the learning of different people. To have this different images needed to be drawn. Drawing and creating Tilemaps was a long process, so five Tilemaps would be designed to show the concept of the system. Inside these Tilemaps there were images ranging from animals to inanimate objects to food, which would give the idea of what could be used for the Tilemaps in future.

## 10.3 Code Structure

In order to pass information through the system and between pages, HTML forms were used. They can hold information through different fields, which can obtain information from the user using text fields. A form can also have hidden elements that allow information to be passed easily through multiple pages Forms have an action filed that links a PHP file that will be launched when the submit button is pressed (see figure 19).

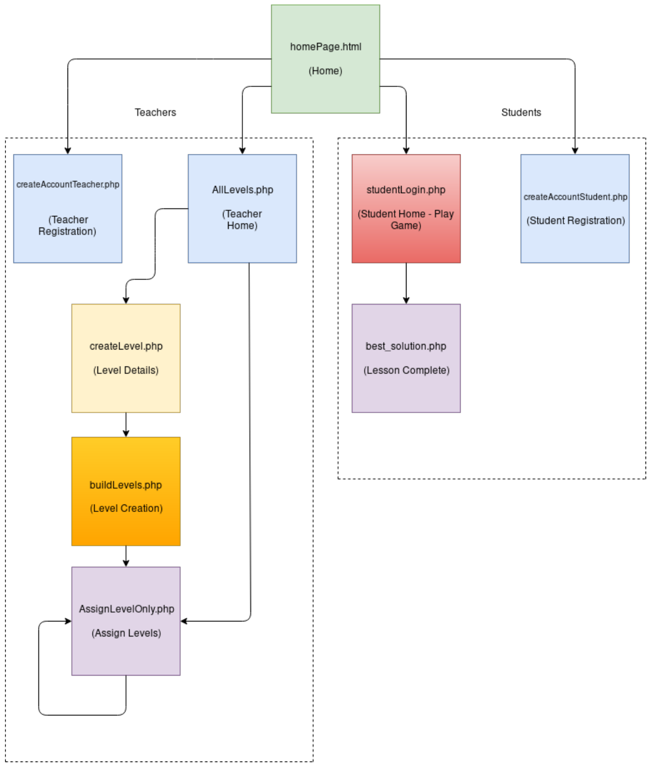


Figure 19: Screenshot of systems map diagram, showing web pages involved.

There are two options when passing information from a form to a PHP file. GET Requests include all the data from the form in the URL of the page that was launched from the form. The parameters are featured in the browser history, which means the page can be bookmarked for later use. When using POST Requests, information that was passed from the form is not visible in the URL, meaning the page cannot be bookmarked. The advantages of using a POST Request are security based: because the information is not visible or storable, a user's account details can remain secure. As the request being handled would contain details of different accounts, POST was chosen to be used throughout the project.

When choosing the how to access the database through PHP, the MySQLi extension was used, as the MySQL extension for PHP has now depreciated. MySQLi is the recommended extension to be used for PHP [22]. MySQLi is still being actively developed, which means any new features will be released for it and not another extension, which means the system can be kept up to date with PHP.

## 10.4 Login and Registration

The home page of the admin centre (see figure 20) contains the following: student login; creating Student accounts; Teacher login; and creating Teacher accounts. The username boxes are input fields with a type of *text*. The passwords have input fields with the *password* type, to guarantee the security of users' passwords when logging into the system, as the environment the users would be working in would most likely be a classroom environment with close proximity to other users. The username and password for each item is contained inside a Form, where the field auto-complete is set to false in order not to reveal users' details. Therefore there are four forms in total, each of which submits to a separate page.

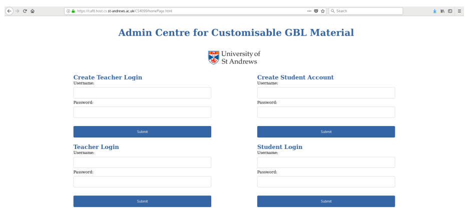


Figure 20: Screen shot of main menu of admin centre

When registration/login pages are launched the information of a username and a password are passed into the file using POST, as this will be needed to create/check an account. All the PHP files that were launched will write data to the database. In order for this to be possible, the files first need to be connected to the database. The name of the database, the username attached to the database and the password to the database are stored in PHP variables at the beginning of the pages, all of which is required when connecting to a MySQL database. Registration follows a fairly standard procedure. with a check to see if the username exists. If the username is valid. it is inserted into the relevant table and the user is notified about their account being successfully created. For logging in, the username is checked to see if it exists, if it does the user is allowed to the systems next area. There is currently no authorisation procedure for whether someone can be a teacher or not. This will be discussed further in section 12.2 Future Works.

The information passed in is written to the database, after which a conditional notifies the user on whether the query was successful or not.

When a teacher or a student logs in the validity of their credentials is first checked. To check if an account exists or not. The following is the SQL command is executed for the teacher's login check.

```
SELECT * FROM teacher
    WHERE name='".$username."' AND password='".$password."'"
```

The student account check would be similar, but with different names. The command checks if there is entry in the database that matches the details passed on by the user. If a result was found then the account has been verified and the page is displayed.

## 10.5 Level Creation

Level Creation is split across two files, representing two pages. When the user arrives on the "CreateLevel.php" (see figure 21) page, the teacher has three values that need to be decided, which are the name of the level, difficulty of the level and the tileset that will be used when building the level. The auto-complete is turned off for the levels value as they cannot share a name.

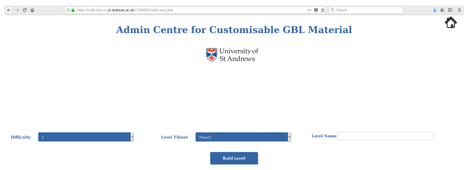

https://caf8.host.cs.st-andrews.ac.uk/CS4099/CreateLevel.php
Search
Admin Centre for Customisable GBL Material
University of St Andrews
Difficulty
1
Level Tileset
Tileset1
Level Name
Build Level!

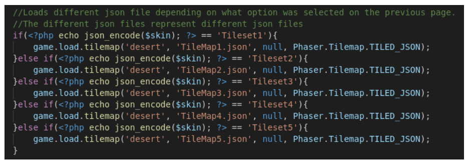
Figure 21: CreateLevel.php web page

The Difficulty of a level is a whole number between 1 and 5, with 1 being very easy and 5 being very hard. The values are pre-coded as to keep the levels consistent, the same difficulty system needs to be used across all levels. The Tileset selects what the skin the build level tool would have for the teacher when they would be designing the level for a user. The different Tilesets that appeared in the list are the pre-made Tileset created for this project. If a teacher would like additional Tilesets to choose from, they can be added by a developer once they have been created using *Tiled*. This information is passed through to the buildLevels.php and stored in hidden elements to be used later while inserting into the Level table.

The level creation area follows the Wireframe diagram from figure 12. There is four areas to the page, the tile pallet, map area, input pallet and solution area. The map area will have a underlying 2D array that will store the structure of the tile map its displaying. This structure will be stored in the database and used to load the levels in the game section of the system. The best solution obtained from the solution area, once parsed is a 1D array that is stored in the database and later used to compare user's solutions against. As described in section 10.1.3, Phaser provides three methods to load, create and update the Phaser Game object. These are customised to build the level creation area.

### 10.5.1 Preload

The *preload* function of the buildLevel.php page contains a conditional structure that will decide which JSON file will be loaded in containing information that will be used to create the Tilemap. The conditional performed over the variable containing the name of the Tileset that was set on the previous page. The chosen tilemap is loaded to make it available to the create function (see figure 22).



```
//Loads different json file depending on what option was selected on the previous page.
//The different json files represent different json files
if(<?php echo json_encode($skin); ?> == 'Tileset1'){
    game.load.tilemap('desert', 'TileMap1.json', null, Phaser.Tilemap.TILED_JSON);
}else if(<?php echo json_encode($skin); ?> == 'Tileset2'){
    game.load.tilemap('desert', 'TileMap2.json', null, Phaser.Tilemap.TILED_JSON);
}else if(<?php echo json_encode($skin); ?> == 'Tileset3'){
    game.load.tilemap('desert', 'TileMap3.json', null, Phaser.Tilemap.TILED_JSON);
}else if(<?php echo json_encode($skin); ?> == 'Tileset4'){
    game.load.tilemap('desert', 'TileMap4.json', null, Phaser.Tilemap.TILED_JSON);
}else if(<?php echo json_encode($skin); ?> == 'Tileset5'){
    game.load.tilemap('desert', 'TileMap5.json', null, Phaser.Tilemap.TILED_JSON);
}
```

Figure 22: Screen shot of code loading tilemap into game

### 10.5.2 Create

Inside the *create* function a Tilemap object is added to the Phaser game, then the conditional structure that appeared in the preload section also occurs, but this time loads in the images for the previously chosen tileset. These images will be the assets used to display the tilemap in the *map area* of the screen. They also will be the assets that make up the *tile pallet*, changing what the teacher can populate the tile map with.

For the teacher to be able to select Tiles to place into the work space, they need to be able to see the Tiles that are being selected. A rectangle with no fill was added to the game using the *graphics* function Phaser contains. The rectangle has the same dimensions as the tiles on the screen and acts as a marker to show where the cursor is (see figure 23).

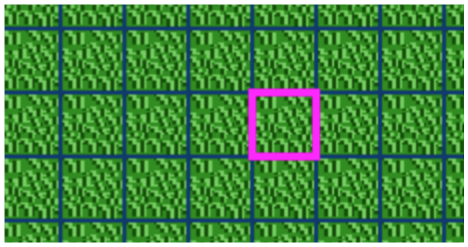


Figure 23: Screen shot showing the marker coloured magenta

As mentioned in the design section there are 4 types of tiles. For each of the types of tiles the marker will be a different colour depending on the type of tile that is selected . This is in order for the teacher to be aware of the type of tile they are drawing the map with. The character tile has the colour blue, goals are red, obstacles are green and terrain tiles are magenta. To select a tile the teacher must hold down the shift key, then click on an item in the *item pallet*. To then draw with the selected tile the mouse needs to be clicked down on the map area and a tile will appear where the maker is. The reason the shift key was needed, was to stop teachers accidentally selecting another tile from the *tile pallet* (see figure 24). The shift key is also a natural key for users as its often used for selections in systems.

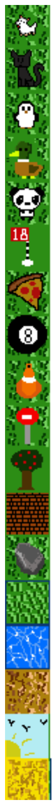


Figure 24: *tile pallet* of admin centre

The create function also adds all the items that appear in the *input pallet* (see figure 25). All these items are attached to a *group*, since it is simple to access all the items in a group and this will be used when parsing the items. A for loop is used to generate multiple objects for each type of input object. For each object the image representing the object is added to the group. It is then scaled to the game, then its attributes are set to allow the user to interact with the object, move it and be able to snap it to the grid space that was demonstrated in the design section. The object has its drag capabilities enabled and then has its enabledSnap() function set to true. This causes the object to snap to a area that is defined in the function. The size of the area chosen is the size of the squares in the background, which in this case is 32x32.

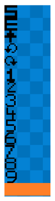


Figure 25: *input pallet* of admin centre

Finally there is an event added to each of the objects that is run when they are released from the dragging motion the user was performing on them. These events are slightly different for each object. There are a set of rules for each type of object about how they can be placed onto the solution area. If a rule is violated relating to an object, that object is returned to the pallet of input objects. The rules on placing the objects was discussed in section 9.12.

### 10.5.3 Update

The update function has four different purposes. It must keep track when an item is selected from the *tile pallet*, when the *map area* is drawn on, when an item is selected from the *input pallet* and when an input is dropped in the *solution area.*

To decide which of the screens the user is in, there is a conditional statement checking the position of the cursor, If the cursor's position is left of the input pallet then they are creating the level, else they are handling the solution input. As the update function is continually run after the page is loaded, its used to keep track of the cursor position and updates it in real time. The marker that shows the tile the teacher is currently over can only appear in the map area section of the page. If the cursor exits the Tileset section of the page, the marker will be destroyed, due to the fact the marker can only select a tile. The destruction of the marker is performed using the Phaser method destroy(). The destroy function removes an object from the Phaser Game. There is another nested conditional that checks whether or not the marker had been destroyed. If the marker had been destroyed due to the cursor leaving the level creation area a new one is drawn.

To select a tile from the *tile pallet* a function is used to firstly obtain the tile. The getTile() function takes two parameters, which are the x and y positions to get the tile from. To obtain these x and y positions a *layer* is used alongside the Tilemap. The *layer* object has functions getTileX() and getTileY(), which returns the tile that the mouse has interacted with. The tile selected is stored in a variable, which is then used when drawing on the map area.

There are several rules that apply when creating a level. One is that a tile cannot be drawn over the *tile pallet*, this is to stop the destruction of the pallet, that would then prevent a teacher from drawing a level in the *map area*. In order for teacher to only be able to select from the pallet and not from any tile that has been drawn on the screen so far, part of the check for selecting a tile is that is must come from the *tile pallet*.

As discussed in the design, there are two representations, the visual representation (skin) and the logical representation (layout) to the Tilemap in the project, which is why the type of tile that is selected is also stored. The type of tile is placed inside a 2d array that represents the Tilemap in the same location it is being placed on the Tilemap. The check for only one character and goal square is done using the function putTile() that is also from the map object. putTile() takes the same parameters and uses the same functions that getTile() uses. Once the putTile() function has run, the Tilemap now contains a tile in a new position and so does the 2D array that represents what kind of object is being stored there. After building the map (see figure 26) the teacher can then build a solution to the level they created.

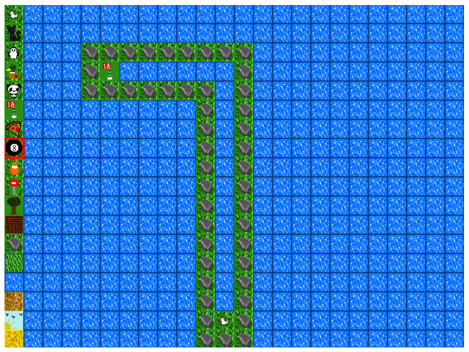


Figure 26: Example Level from project

Once the teacher has finished placing items from the *input pallet* into the *solution area* (see figure 27), they can save this solution by clicking on the run button. All the objects that have been placed by the student are inserted into an array in the order they appear in the area from top left of the grid to the bottom right of the grid.

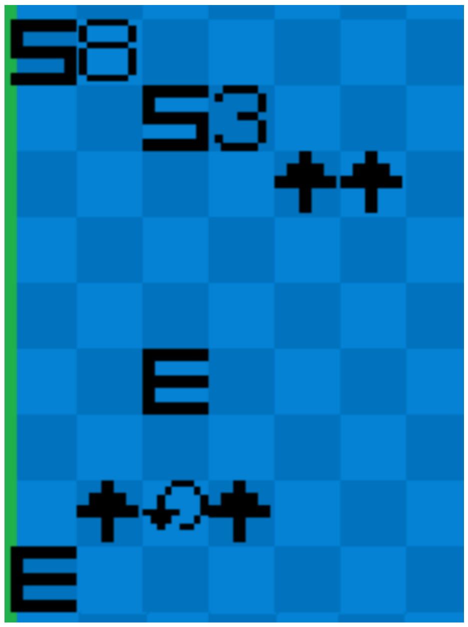


Figure 27: Example input from game.

The array with all the commands placed inside it, is then processed by the filterArray() function. The loop objects duplicate these objects equal to the number of iterations of the loop. Commands in the array are expanded where appropriate, so a loop with n iterations is replaced with n times the commands from the loop. At the beginning of the method there are some variables defined to be used for the parsing. These will be used to track the beginning and ending of loops, the size of the loops and whether any loops are still to be parsed from the array. After finding a start loop symbol, the symbol straight after it must be a number, according to the defined rules. The loop number is set to the value the number object represents. The array containing the commands is then split into three arrays. All the symbols that appear before a loop, all the symbols inside the loop and all the symbols that appear outside the loop. The loop symbols for that loop are then removed, and the new array that contains all the symbols inside the loop is then duplicated equal to the iteration value. All the duplications are concatenated onto the array containing the symbols before. The array containing the symbols after is concatenated onto the result of the previous concatenation. This process is continually performed until all loop symbols have been removed. After this has been completed the best solution for the level is ready to be stored in the database.

The level for game has now been completed and can be saved to the database. After clicking on the submit button a form will send all the information required for the level table in the database to the file AssignLevelOnly.php, ready for being inserted in the level table.

## 10.6 Assigning Levels

There are three routes when assigning levels: A new level with no students, existing level with no students and existing level with students. For the first route the level needs to be first written to the database. The variables that contain the information that makes up a level is collected from POST data. All the students in the database are then selected because by definition, no students are yet assigned. An empty array is created for the assigned students that will appear in the table showing all the assigned students.

For the second route, existing level with no students, all the students currently assigned to the level in question are selected to display in the table. Then all the students not assigned to the level are selected, to display in the drop down menu, which can be seen in query below.

```
SELECT name FROM student WHERE NOT EXISTS
    (SELECT 1 FROM studentLevel
    WHERE student_name = name AND Levelname = '".\$level_name."')
```

The next route, is the same as the route above but also requires the information to be written to the studentLevel table. The variables to be written are obtained from a query based on the level name.

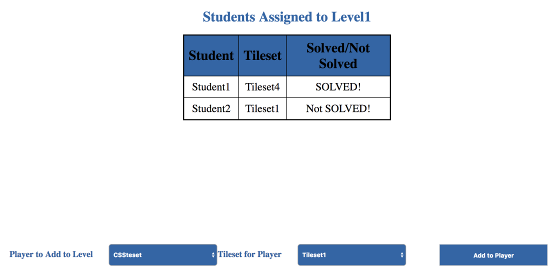
All three cases use the same variable names and use the variables to build the table and the form.

On loading the AssignLevelOnly.php the only value that is immediately loaded is the level name, as this value is not dependent on the route taken to navigate to the page. Regardless of the route taken, the page displays a form containing a list of all students not currently assigned to the level, and a list of all the tilesets. These are shown in separate drop down menus.

Each form that launches this page has a hidden field to indicate what action should be taken. A hidden field is used, as the user does not need to know how a query was reached. To keep the system in a professional styling the method to add levels needs to be consistent and appear the same to the user on the front end. What happens on the server side is not important to the client, just the result. Another reason to use this extra field to show where the form was launched from, as in terms of business logic it makes sense. With this indicator the other values being passed will not have logic performed on them, meaning they can be changed without affecting this section of the system.

A conditional is used to determine what action to take in the PHP file. A built in PHP function called isset() is used. This function checks whether a value has been set or not when a new level is to be created. The value is “AddLevel”. This value appears when the new level with no students route is taken. The query for the database is adding a level to the database.

In figure 28 it shows the AssignLevelOnly.php page where there is a button for adding players. When this is selected a form is submitted to the AssignLevelOnly.php page, but this time it contains a hidden field with “AddStudent”.


Students Assigned to Level1

| Student | Tileset | Solved/Not Solved |
|---|---|---|
| Student1 | Tileset4 | SOLVED! |
| Student2 | Tileset1 | Not SOLVED! |

Player to Add to Level: CSSteset
Tileset for Player: Tileset1

Figure 28: A Screen shot of assign levels page.

By the form submitting back to the same page, students can be added in bulk more easily. The else of the conditional checks for the “AddStudent” value and if this exists a query is launched to link the student selected with the level.

The initial set up for the project involved having multiple files for the different ways this part of the process could be reached from. This was a less efficient and more cluttered solution, where the amounts of repeated code would be high. This is why the isset() function is used to ease these issues.

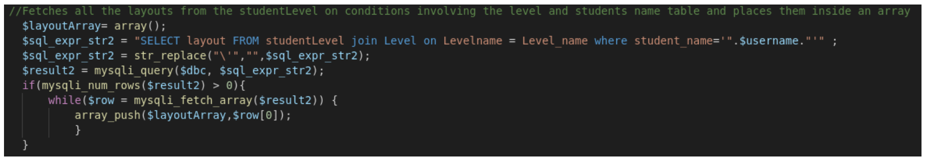
Outside the conditional statement the information required for the table is accessed from forms on previous pages. A loop is done over the results of a query to obtain all the entries of one of the variables. While there is still information to be put through the loop, it is pushed onto a PHP array, which will be echoed into a JavaScript array, which subsequently is placed into fields in the form on the page. The information for the table is required whenever the teacher accesses the page because of the way students are added, hence is placed outside the main conditional.

The Tileset chosen on this page will decide how the game looks to the student playing it and appears in the table along with the student name. Finally the other value in the table is whether a student assigned to a level has solved it or not. This is decided, by checking whether a student's best solution is different from its initial value. The only way for their best solution value to change is by solving the level, hence can be used to calculate the table value.

## 10.7 Student Game Play

Similar to when a teacher first logs into their account, when a student first logs into their account, their details are verified using the database. If no account exists the code will terminate with an error message. If the account does exist, all the information that will be required for all the levels the student has been assigned will be loaded in from the database.

In figure 29 you can see the same loop structure that appears in the AssignLevelOnly.php file when the students and tilesets are retrieved. All the information is retrieved using the SQL join function on the studentLevel and Level tables.



```
//Fetches all the layouts from the studentLevel on conditions involving the level and students name table and places them inside an array
  $layoutArray= array();
  $sql_expr_str2 = "SELECT layout FROM studentLevel join Level on Levelname = Level_name where student_name='".$username."'" ;
  $sql_expr_str2 = str_replace("\'","",$sql_expr_str2);
  $result2 = mysqli_query($dbc, $sql_expr_str2);
  if(mysqli_num_rows($result2) > 0){
      while($row = mysqli_fetch_array($result2)) {
          array_push($layoutArray,$row[0]);
          }
  }
```

Figure 29: Screen shot of code retrieving the layouts for all lessons from the student's lesson plan

The HTML that is used inside the student login page has a mix of elements. There are text elements that display the information of the level. All of these elements have the readonly value inside them. This is so no one would be able to alter the value in these boxes which could affect the student playing the level. There are also text fields that are hidden from the student. These will contain values that will be written into the database when the student finishes their lesson plan.

The "NextLevel!" button is disabled, meaning a user cannot click on it to activate an event. This only becomes available when certain conditions are met. The reset button resets all the variables in the program that are needed for the student to play the game. It returns the game state of the current level appearing on the left hand side back to its original position. The submit button is also disabled until certain conditions are reached. Similar to the when a level is created there is the Phaser program structure of preload, create and update.

### 10.7.1 Preload

The preload function is similar to before, once the JSON file has been loaded in that represents the Tileset that the student has been assigned. The images that make up the input graphics and the background are then loaded in. The images that make up the Tileset are also loaded into the Phaser Game. All the information from the first level is loaded into the HTML text elements.

### 10.7.2 Create

As in section 10.5.2 the create function starts by adding the graphic files of the map after it has been defined. The background is created by adding a sprite to the game, using the image that was designed for the background. The reason a sprite was used is that a sprite in Phaser can have camera movements and animations attached to it. These features would be very helpful when adding additional features in future.

The images associated with the Tileset are then added to the map. In order to carry the movement of the game easily and be able to continually draw the map throughout the playing of the game, there is a variable defined and used to store each type of Tile that appears on the map. The Tile is retrieved from the pallet, which is there when the map is first loaded as it is built into the JSON file. The Pallet cannot be visible to the student when they are playing the game since this would detract from the user experience. Unlike when a level is created the Pallet is covered by an obstacle tile during the update function. This is currently done naively and could be improved.

A Phaser button is added to the game for the user to click on to Run the input they have created. When the button is clicked on an event is triggered that starts the parsing of the user input. The same parsing occurs but this differs from createLevel as it moves the character after parsing.

To give the student information about how each of the input objects function, it was decided that tooltips have been added to the system explaining what each object does. There is no predefined tooltip features in Phaser, but after some research, a library was found from a 3rd party developer who had created tooltip objects for Phaser[5]. Once the tooltips were implemented they were assigned an object to attach to for every object that was created in the creation loop. The text that will be displayed inside the tooltip and the colour of the tooltip is then set. Then the way the tooltip will be enabled is also set, which in the case of this project it is when the object is hovered over. Finally the position of the tooltip is chosen, which is set to the right (see figure 30).

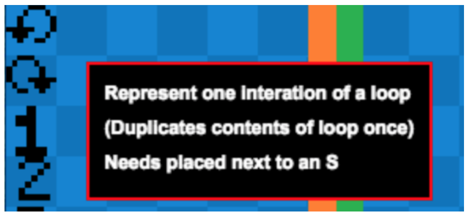

Represent one interation of a loop
(Duplicates contents of loop once)
Needs placed next to an S

Figure 30: Screen shot of tooltips

This was so the tooltips could not hover over any of the game map, hiding how the game is progressing from the student. After a student has familiarised themselves with what each of the objects do, then they do not want to constantly see what the tooltips say. Two events were set up in the file to change the student's view of the tooltips: if the up arrow is pressed the tooltips are disabled and the if the down arrow is pressed then the tooltips are enabled again.

There is no marker created in the student section of the system. This is because the student should not be able to select a tile as there is no *tile pallet* and the *map area* is not editable.

### 10.7.3 Update

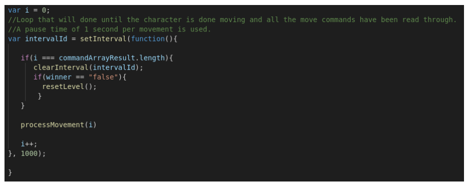
Similar to section 10.5.3 the update function of the file starts by generating the user's current level. It loops through the 2D array that contains the general representation of what the level looks like. For example if it reads the string "character5", then it places the character 5 tile that was set earlier in the code in the position in the Tilemap that is linked to. It also updates a 2D array called levelArray that is the current representation of the level. This will be used to keep the map up to date and allow the user to move the character through the level. To place a tile on the Tilemap that is linked to the 2D array the following code is used:

```
map.putTile(currentTile,
layer.getTileX(j*32), layer.getTileY(k*32));
```

As the tiles are all 32x32, the position in the 2D array combined with the value 32 can be used to assign the tile to the correct location.

If the run button has been pressed, the update function calls a function called moveCharacter() (see figure 31).

The following code below is the loop that is controlling the movement of the character in the game.



```
var i = 0;
//Loop that will done until the character is done moving and all the move commands have been read through.
//A pause time of 1 second per movement is used.
var intervalId = setInterval(function(){

   if(i === commandArrayResult.length){
      clearInterval(intervalId);
      if(winner == "false"){
        resetLevel();
       }
   }

   processMovement(i)

   i++;
}, 1000);

}
```

Figure 31: Screen shot of the loop controlling the movement of the character

To increase the user's experience and to help show them where their input becomes incorrect, it was decided that a delay between each movement was needed, instead of the character, appearing at the end location of their movement.

To insert a delay between each movement of the character, the JavaScript function setInterval() is used. This function works by calling an external function at specified intervals. To have this function, a dummy function is needed that then calls the function to be delayed. In this example it calls the function processMovement() with a delay of one second between each call. setInterval will constantly run until it is stopped. There is a check to see if the setInterval had been called enough times, and if it has then a function called clearInterval() is called that ends the timer. If the student has not successfully completed the game, then level is then reset.

When the program initially enters the processMovement() function, the update() function is called. Due to the update function being responsible for updating the state of the Tilemap, it will also be responsible for moving the character.

The *input pallet* and *solution area*, work the same as in the teacher area. The inputs in the solution area are parsed down into a 1D array once the run button is clicked. This array is then looped over inside the time loop.

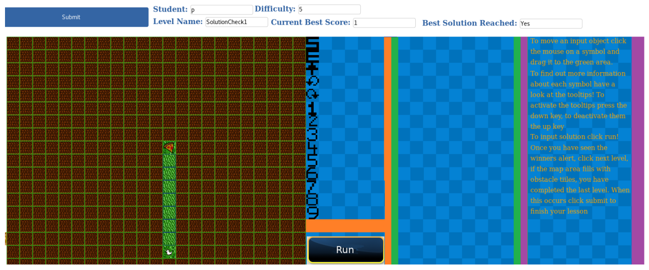
There are three different commands to deal with: Clockwise rotation; Anti-clockwise rotation; and forward. To keep track of the rotations the direction the character is facing is stored. The possible options for the direction the character can face are: forwards; right; backwards; and left. When clockwise is found in the loop then the direction is shifted one to the right in the list. The anti-clockwise symbol shifts the direction one to the left in the list.

For the forward command there are two possible states the game can be in, either it is the first move or not. The reason there needs to be two states is that when it is not the first turn, the tile the character moves onto needs to be stored in order to be put back once the character moves off the tile again. The tile the program stores depends on the direction the character is facing. This affects whether it is +/- 1 on the x or y coordinates in the array. The position of the character is stored and used to get the position of the next tile.

The 2D array that is used to generate the level is altered as the character moves through the level. The 2D array is also used to check the validity of the character movement. If the tile the character is going to move onto is a obstacle tile then the student is told using a HTML alert and the level is reset. If the character is going to move onto a goal tile, then the state of the game is completed and the user is given an HTML alert saying they have won the game. The button that allows the user to run and have the inputs parsed is then disabled and the winning condition is set to true, which means the setInterval() function does not need to reset the level anymore.

Finally, when the user has completed the level, the "NextLevel!" button is enabled and once the student clicks on it the next level is loaded. This continues until there are no move levels for the student to complete, upon which the game screen is covered in purely obstacles and the submit button becomes active.

The best solution for each level is calculated by counting the number of symbols that could affect the character after the initial list of symbols had been parsed. At the beginning of the file the current best score the student has for each level is loaded in. They are placed into a JavaScript array, with the first level being the first entry in the array and so forth. If the student completes the level in a method that betters their best score, then their new best score is stored in the array replacing the old value in the array. The array is stored in a hidden elements that will be written to the database when the submit button is clicked on. If the new best solution that was reached is equal or less than to the best solution that was set by the teacher then the "Best Solution Reached" box will display "yes", every time the level is loaded (see figure 32).


Student: p
Difficulty: 5
Level Name: SolutionCheck1
Current Best Score: 1
Best Solution Reached: Yes

To move an input object click the mouse on a symbol and drag it to the green area.
To find out more information about each symbol have a look at the tooltips! To activate the tooltips press the down key, to deactivate them the up key
To input solution click run!
Once you have seen the winners alert, click next level, if the map area fills with obstacle tiles, you have completed the last level. When this occurs click submit to finish your lesson

Figure 32: Screen shot showing student has the best solution

Once the submit button is clicked the "best_solution.php" file is launched from the form. The arrays that were passed through from the previous page are in the form of a string. These need to be in the form of an array again, therefore the PHP function explode() is used. This splits a string on a designated delimitter, which in this case is a comma. The new PHP arrays are looped over with the following command being used to write into the database:

```
UPDATE studentLevel SET best_solution = '$best_solution_array[$i]'
WHERE student_name='".$username."'
AND Levelname='".$best_solution_level_name_array[\$i]."'"
```

The query updates the value of best solution in the studentLevel table for the student that is currently logged in. After the query has been performed the student is greeted with a message saying that they have successfully completed the game. The student can now log back in to try and better their solutions, using the home button to get there immediately.

# 11 Evaluation

This section considers the success of the project in different contexts. First, the system will be evaluated against the primary and secondary project objectives. Then the success of the project as a whole will be looked at, with finally the comparison of the project against real world examples of similar work.

## 11.1 Progress of Objectives

### 11.1.1 Primary Objectives

P1 Create an admin centre for users to log in to where they can build their own GBL activities.

P2 Create a base game for GBL that will be used in the admin centre.

P3 Create levels of different difficulties to be used in the game.

P4 Have different skins for the game based on target audience of users.

All the primary objectives that are stated above have been successfully completed.

For objective 1, creating the database was the first task that needed completed. Once the database was running, the log in features could be implemented. Using the activity and web flow diagrams, the admin centre was created with a professional and slick appearance. After the admin centre was constructed the ability to create levels was added. The ability to complete this objective was dependant on Objective 2.

The structure for game was researched and developed using the Phaser framework, after there was the logic for the game, the input could be added. A custom format and process was produced for storing the maps and game components. The way the input is parsed would also allow other input options to be used in future. The structure of the game created for Objective 2 was used for creating levels in Objective 1.

Objective 3 was completed after objective 1 was completed. The tools that were used in the admin centre for creating levels was used to create the levels of different difficulties. Various levels were created to show what the system was capable of.

Finally objective 4 was completed after there was a game to apply skins too. Five skins were produced using the Tiled software. More skins could have been produced for the project, but it was decided that it would be more important to focus on other sections of the project. It was more challenging than first thought to create different skins, so the amount of skins that could be produced was unfortunately lower than first expected.

## 11.2 Secondary Objectives

S1: Design a ratings system to give an accurate representation of the difficulty of the game.

S2: Allow users to create their own levels of the game for other users in the admin centre.

S3: Have a automatic moderation system to keep the admin centre clean and fit for use in classes.

The first of the secondary objectives was not met. Calculating the best score of a student took longer than expected on the project, which did not leave much time to work on a algorithm for calculating an accurate representation based on how the user performed. After knowing how close the user was to the best solution possible, all the information necessarily to write an algorithm is available, but as it needed to be an accurate representation, the algorithm would need to go through planning and there was not enough time to complete this to a high enough standard.

Objective 2 was partially completed. As discussed with the primary objectives, there was a feature in the admin centre for the teacher to create levels to be used. This wasn't extended to be a feature on the student log in section of the system. In order to do this there would have needed to be a intermediate menu such as when the teacher first logs in. There would need to be a approval process so that teacher could prevent inappropriate levels being created and restrict which students could create levels. One method of doing this would be only giving students access to this feature after they have completed their lesson.

Objective 3 was also not completed in this timescale. As discussed, the scope of the project had changed and the primary objectives were much larger than first expected. There was little time to precede with this objective. To complete the objective a chat system that stores all comments would need to be produced first, then a moderation system would need to be applied on the top of this. Research into how the moderation system would function would need to be done and the kind of information would need be filtered out.

### 11.2.1 Reflection on Objectives

As the project continued the set of primary objectives that were initially decided on evolved and were broken down. The first primary objective could be broken down into two separate objectives, one objective to create the admin centre and one to create levels for the game. Similarly the second objective creating a game based learning game on reflection should of been two objectives as well. Firstly creating a game then another objective being creating the new user input for the game.

Retrospectively looking back, the primary objectives should have been broken down into more objectives, with some objectives maybe becoming secondary objectives. The MoSCoW method could of been used to iteratively produce the objectives.

## 11.3 Evaluation against current systems

Considering how the project compares against current systems is necessary for a good and complete evaluation of the system. As seen in the context survey there were two examples of currently used systems that are directly comparable. Robozzle was a game that involved moving around a grid like system similar to the game created for this project. There were lots of features in Robozzle that did not create the best environment for an average student to use. Robozzle's entire system was designed in a plain, dark coloured theme with sharp edges suggesting it was for an older user. The menu was not intuitive and complicated to use, failing Shneiderman's golden rules of striving for consistency and reducing the short term memory load of the user. The admin centre's central idea was around creating a game that could be customised to a user's preferences. This has been shown with the skins that have been produced. Also the professional look for the system that was designed using Shneiderman's golden rules is more usable for both the teacher and the student.

The other example of a current system was Hour of Code. The problems that were identified for it were it was targeted at individual demographics which alienate some students from possibly might be a useful lesson. In this project, this was addressed with the use of different skins. There is still room for the graphics to become as sharp as Hour of Code's, but in the limited time of the project the proof of concept has been shown to solve the issue of styling for a specific person. The level of ability in Hour of Code is often directed towards a beginner level. This could become problematic if not everyone in a class is at that level. The admin centre allows the teacher to easily set different levels to different students of different difficulty, so that everyone can have a lesson that matches their ability.

The admin centre has shown that it has features that can address the issues that the current systems have, which could increase the motivation and engagement levels of the student. The skins and game features can be improved with more time spent on the project, but the concept of it being able to work have been shown to be true.

## 11.4 Limitations and Improvements

There are a few limitations with this system, for example, the procedure in creating a tileset using Tiled can be a long process, which at first would be problematic when there are not many tilesets to choose from. There would need to be development cycle to create these skins. The students who would be completing the level would need to state their preferences to their teachers if there was no level they currently liked. The teacher would then discuss this with the developer who would then produce the skin. This would be a continuous process throughout the lifespan of the software, but would become lower over time as more skins are produced.

As part of the development cycle assets would also need to be drawn for the image overlaid on the tileset. All the graphics were produced from scratch in the project, which could also be a limitation. Graphics could have a higher quality if a graphic designer was used, this again could add to the time taken to produce new skins, however it would give a higher immersion to the game.

The input section of the system has limitations, but can also be improved. The grid based system has restricted the different inputs the game could process. Currently the system teaches for loops to the user. This could be changed to teach different types loops due to the method for parsing inputs. The grid system and input could also be used to teach conditionals as shown in Robozzle.

There is currently no restrictions on who can become a teacher in the admin centre. Since a teacher account has the power to assign and create levels, access should be restricted to genuine teachers. This would require some additional approval process, possibly managed by a system administrator who would need to provide a code for the teacher to then use when logging in, or should set up the teacher's account for them on request. The code/passwords that would be produced would need to be randomly generated strings of characters, numbers and symbols to guarantee the security of the codes/passwords.

More refactoring could be performed on the system. Since I was learning several new languages, getting used to them took time. The first priority was getting the system to have the correct functionality in order to meet the objectives and this was easiest with the code centralised in one file. As I was learning new languages and developing a system using them, there was limited time to complete refactoring and the process was not fully completed. If more time was available on the project this could be completed.

Limited testing was performed on the system and a couple of minor issues were detected. When using the input symbols and placing multiple loops, a loop can be indented inside another loop, but a loop cannot be placed in line with another loop, this causes the input to be incorrectly parsed. As time was limited a fix was produced to keep the intended functionality, but not as initially wanted. A loop can be placed outside a loop but not directly in line. Another issue is when drawing using the fifth terrain. This can cause the *map area* to incorrectly display. This was due to the fifth set of terrains not being drawn till later and there was not enough time to debug. Finally only terrain tiles can be placed around the outside of the *map area* currently. The reason for not being able to remove all the bugs in the system, was due to it taking time to become familiar with all the new languages and how they function. If more time was available the bugs could be removed through more thorough testing.

# 12 Conclusion

## 12.1 Overview of objectives

The main objective of the project was to create an admin centre for creating customised levels for a game-based learning game that are suited to the student participating in the lesson, which can boost their engagement and motivation. This is represented by the primary objectives where creating the admin centre, the game and the different skins were split into different objectives. These objectives were met, as there was an admin centre where levels could be created and assigned to the student, customising them with five skins. There was also a game that would teach users how loop structures work. There were additions and directions the system could take in the future.

## 12.2 Future works

The system can be expanded to add the additional features discussed in the previous sections. More skins can be added to the system so there are more choices for the student when playing the game. By having more input types the user base could be expanded, which in turn could expand the number of skins in the system.

The main new piece of work that could be incorporated into the project would be an empirical studies into the advantages of the system. Participants in the study could be tested with a game with generic skin that was not personalised to them, they could then perform the experiment using a customised skin. This is a within-subjects design, which has the advantage of needing less participants than a between subject experiment. The variance can be controlled within the participant as well unlike between experiments. To stop a skill transfer between the experiments training them on how the system works can be performed before hand to try and limit them improving as the experiment continues. Also participants initial environment should be equally distributed between the two systems, half should use the customised version first and half the basic version first, which would aid in balancing the experiment.

The empirical study could be carried out over a range of users across many demographics to see its effectiveness with different users. Extensive ethical consideration would need to be completed first due to the sensitive are of experimental research with children as participants. This would need to be carried out before any empirical studies would be preformed.

## 12.3 Summary

I feel this project has been a success. I have produced an admin centre that allows a teacher to register then create levels, customising them to students.

Students can then log in to the system and complete a lesson plan that is tailored to them through how complicated the level is and how the styling of the level appears. These features combined may increase the motivation and engagement of the students. I feel the system has lots of potential if the skins, inputs and lessons are expanded, meaning the project could benefit a wide range of users.

On a personal level I have learned through the entire experience. This was the first time I proposed a project from scratch, which allowed me to then gain experience about how to create objectives and estimate the scope of a project, while working with a supervisor/customer. The depth that the design process went into was much greater than I had previously experienced. I learned how to create, then interpret activity diagrams and System maps. The information I gained from here made creating the structure for the database easier, which will be something I will use in future.

I decided to propose a topic in this research area with web based technologies as I had no prior experience with them. I also wanted to learn how to gain expertise in a new framework, something I have struggled in the past with. As the project concluded, my skills in web based programming and using new frameworks have vastly increased and I now feel confident in solving problems using them.

Finally this was the first large software engineering project I have completed individually and the experiences gained from the process will be invaluable, when developing future products.

## 13 Acknowledgements

Personally I would like to thank my supervisor Ruth Letham for all the support throughout the year even after the project was originally due, I would not of been able to finish the project without it.

# 14 Appendix

## 14.1 A - User Manual

There are two types of users who can use the system. A teacher, who can create different levels and assign them to students, changing the appearance of the level for each student they assign it to. There is also the student, who can log in and complete a lesson plan that has been created by a teacher.

The system's homepage is found on https://caf8.host.cs.st-andrews.ac.uk/CS4099/homePage.html. After arriving at the homepage there are four different input areas. To create either a student account or a teacher account, a username and password must be entered. The username must be unique and if the entry is not then you must re enter the details and try again.

### 14.1.1 Teacher

When logged in as a teacher, a user is given two options. Creating a new level, or assigning a preexisting level. To assign a pre-existing level, select a level from the drop down menu and click the next button, which takes the teacher to the assign level area.

To build a new level, click on the build level button, which takes the user to a page where you will select the difficulty of the level being built, the tileset you want to use while building and the name of the level. Once these values have been entered, click on the 'Build Level!' button.

The level can now be created. There are instructions down the right hand side of the solution area describing how to build your level. After the teacher is happy with the level, click the submit button. The teacher then arrives where levels are assigned.

In the assign level area, students can be assigned. Students and the tileset the teacher wishes to assign to them are selected from drop down menus. To then add a student to a level, the teacher clicks on the 'Add to Player' button.

### 14.1.2 Student

When a student logs in they are greeted with the game play section of the system. There are instructions down the right hand side of the solution area describing how to play a level. Once an input is run and the character moves, the student receives an alert saying if they were successful or not. Once a student completes a level they can either click on the reset button to solve the level again with a better solution of click on next level.

Once all levels have been complete, the game screen will be full of obstacles and the student can now click on the submit button. Their best solution for each level is stored (if it was better than their previous best solution) and will be displayed at the top of the level next time they log in.

## 14.2 B - File Listings

- homePage.html
- createAccountTeacher.php
- AllLevels.php
- createLevel.php
- buildLevels.php
- AssignLevelOnly.php
- studentLogin2.php
- createAccountStudent.php
- best_solution.php

## 14.3 C - Administration Centre Accounts

### 14.3.1 Student Accounts

| Username | Password |
|---|---|
| Student1 | Password1 |
| Student2 | Password2 |
| Student3 | Password3 |
| Student4 | Password4 |
| Student5 | Password5 |

### 14.3.2 Teacher Accounts

| Username | Password |
|---|---|
| Teacher | TeacherPassword |

## 14.4 D - Current Game Levels

- Level1
- Level2
- Level3
- Level4

## 14.5 Ethics Form

UNIVERSITY OF ST ANDREWS
TEACHING AND RESEARCH ETHICS COMMITTEE (UTREC)
SCHOOL OF COMPUTER SCIENCE
ARTIFACT EVALUATION FORM

Title of project

Admin Centre for Customizability of GBL Material

Name of researcher(s)

Christopher Fleming

Name of supervisor

Dr Ruth Letham

Self audit has been conducted **YES** [✓] **NO** ☐

This project is covered by the ethical application CS12476

Signature Student or Researcher

[illegible]

Print Name

CHRISTOPHER FLEMING

Date

23/11/17

Signature Lead Researcher or Supervisor

[illegible]

Print Name

RUTH LETHAM

Date

30/11/2017

## References

[1] `https://www.uml-diagrams.org/`. Accessed on 3/6/18.

[2] *Babkin, J. VISUAL PROGRAMMING LANGUAGES.*

[3] BBC. *BBC micro:bit.* `https://www.microbit.co.uk/home`. Accessed on 3/6/18.

[4] R Davey. *Phaser.* `https://phaser.io/`. Accessed on 9/6/18.

[5] M Dobekidis. *PhaseTips.* `https://github.com/netgfx/Phasetips`. Accessed on 3/6/18.

[6] *Dr Adam Barker CS3051 Software Engineering Lecture 2: Software Development Processes.*

[7] *Erhel, S., Jamet, E. (2013). Digital game-based learning: Impact of instructions and feedback on motivation and learning effectiveness. Computers Education, 67, 156-167.*

[8] Google. *Blockly.* `https://developers.google.com/blockly/`. Accessed on 3/6/18.

[9] *Huizenga, J., Admiraal, W., Akkerman, S., Dam, G. T. (2009). Mobile game-based learning in secondary education: engagement, motivation and learning in a mobile city game. Journal of Computer Assisted Learning, 25(4), 332-344.*

[10] *Jiau, H. C., Chen, J. C., Ssu, K. F. (2009). Enhancing self-motivation in learning programming using game-based simulation and metrics. IEEE Transactions on Education, 52(4), 555-562.*

[11] *O'Kelly, J., Gibson, J. P. (2006, June). RoboCode problem-based learning: a non-prescriptive approach to teaching programming. In ACM SIGCSE Bulletin (Vol. 38, No. 3, pp. 217-221). ACM.*

[12] I Ostroovsky. *Robozzle.* `http://www.robozzle.com/`. Accessed on 7/6/18.

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

[14] H Partovi. *Hour of Code.* `https://hourofcode.com/uk`. Accessed on 7/6/18.

[15] *Pivec, M. (2007). Play and learn: potentials of game-based learning. British Journal of Educational Technology, 38(3), 387-393.*

[16] *Prensky, M. (2001). The Digital Game-Based Learning Revolution . Computers in Entertainment (CIE), 1(1), 21-21.*

[17] *Ramamoorthy, C. V., Prakash, A., Tsai, W. T., Usuda, Y. (1984). Software engineering: Problems and perspectives. Computer, 17(10), 191-209.*

[18] *Schwaber, K., Beedle, M. (2002). Agile software development with Scrum (Vol. 1). Upper Saddle River: Prentice Hall.*

[19] *Shneiderman, B. (2010). Designing the user interface: strategies for effective human-computer interaction. Pearson Education India.*

[20] B Shneiderman. *Shneidermans Eight Golden Rules of Interface Design.* `https://faculty.washington.edu/jtenenbg/courses/360/f04/sessions/schneidermanGoldenRules.html`. Accessed on 8/6/18.

[21] B Silverman. *Scratch.* `https://scratch.mit.edu/`. Accessed on 7/6/18.

[22] PHP Team. *PHP.* `http://www.php.net/manual/en/mysqli.overview.php`. Accessed on 9/6/18.

[23] Montana State University. *PHP vs Java.* `https://www.cs.montana.edu/~tosun/phpvsjava.pdf`. Accessed on 9/6/18.