

Developing an educational web game based on water management

H2O Flow

CS39440 Major Project Report

Author: Matthew Lee (mal80@aber.ac.uk)

Supervisor: Dr. Wayne Aubrey (waa2@aber.ac.uk)

30th April 2019

Version 1.4 (Release)

This report is submitted as partial fulfilment of a BSc degree in
Computer Science (G400)

Department of Computer Science
Aberystwyth University
Aberystwyth
Ceredigion
SY23 3DB
Wales, UK

Declaration of originality

I confirm that:

- This submission is my own work, except where clearly indicated.
- I understand that there are severe penalties for Unacceptable Academic Practice, which can lead to loss of marks or even the withholding of a degree.
- I have read the regulations on Unacceptable Academic Practice from the University's Academic Registry (AR) and the relevant sections of the current Student Handbook of the Department of Computer Science.
- In submitting this work, I understand and agree to abide by the University's regulations governing these issues.

Name: Matthew Robert David Lee

Date: 30th April 2019

Consent to share this work

By including my name below, I hereby agree to this project's report and technical work being made available to other students and academic staff of the Aberystwyth Computer Science Department.

Name: Matthew Robert David Lee

Date: 30th April 2019

Acknowledgements

I am grateful to the project supervisor, Wayne Aubrey (waa2@aber.ac.uk) for helping with the concept for the project and giving feedback on the creative decisions made within the game.

I'd like to thank the numerous play testers that played the game that I have produced. I am grateful for the positive and negative feedback that I was given.

Abstract

With the problem of unnecessarily wasting water becoming a big issue, this topic could benefit from being presented to audiences through a new type of media.

The aim of this project was to produce an educational web game, which uses the media of gameplay to teach people why wasting water is bad. A successful game would address the costs of cleaning water, using interactive gameplay.

The educational topic within this project should be highlighted through gameplay mechanics but can also be shown to the user through information. It is important that the educational message is addressed during this project.

This report discusses the development of this game as well as the results and feedback from members of the games target audience. The game will be created using HTML and JavaScript, so the game can be accessible to a wide range of people without relying on additional software.

An approach focused on Feature Driven Development was taken to develop this project. The use of FDD allowed for the analysis, design, and implementation of all the key features that will make the game function as intended.

The approach taken and the game produced was ultimately successful based on user feedback and the aims of the project is met. However, there were some issues with some features not being implemented and with some features requiring additional design steps.

Contents

1. Background, Analysis & Process	7
1.1. Background	7
Similar Systems	7
Technology Research	9
Functional Research	10
1.2. Analysis of research	11
Functional Requirements	11
Non-Functional Requirements	12
1.3. Method	13
2. Design	14
2.1. Overall Architecture	14
2.2. Function Design	15
2.2.1.1. Water Properties	15
2.2.2. Filter Properties	18
2.2.3. Score system	19
2.2.4. Saving scores	20
2.3. User Interface Design	22
2.3.1. Home page	22
2.3.2. Game Screen	26
2.3.3. Tutorial Page	30
3. Implementation	33
3.1. Function implementation	33
3.1.1. Weather randomization	33
3.1.2. Filter Object	34
3.1.3. Game timers	35
3.2. Point Systems	36
3.2.1. Preventing re-initialization of the points variable	36
3.2.2. Calculating points	37
3.2.3. Point Storage	38
3.2.4. Displaying scores	39
3.3. Graphical User Interface	40
3.3.1. Animated water textures	42
3.4. Conclusion	42

4. Testing	43
4.1. Overall Approach to Testing	43
4.2. User Interface testing	43
4.3. Feature Functionality testing	44
4.4. Storage testing	45
4.5. User tests	46
4.5.1. Science Week Feedback	46
4.5.2. Game Tester 1	47
4.5.3. Game Tester 2	47
4.5.4. Game Tester 3	48
4.6. Known Bugs	48
4.6.1. User Interface bugs	48
4.6.2. Functional bugs	48
5. Critical Evaluation	49
5.1. Project aim	49
5.2. Project Management	49
5.3. Design	50
5.3.1. GUI design	51
5.4. Testing	51
5.5. Future Development	52
5.6. Conclusion	53
6. Annotated Bibliography	54
7. Appendices	56
A. Feature List	56
B. Test Tables	60
1. User interface testing table	60
2. Functionality Tests	62
3. Score storage tests	64
C. Third-party code and libraries	66
D. Ethics Submission	67
E. Version control	69

1. Background, Analysis & Process

1.1. Background

The aim of this project is to create an educational game for public engagement that will be used to teach a topic of interest to a target audience. This game aims to tackle the topic of “how water is cleaned and costs of doing so.”

The project problem has a large amount of appeal due to the ambiguity of an educational game. There are lots of different topics available to develop into a game for this project, which allows for a sense of creative freedom. The topic of showing how much work goes into cleaning water was a topic suggested by the projects supervisor and was then chosen to be the topic for the project.

The idea of making an educational game with the underlying message of “don’t waste water” is something that is wanted by Welsh Water (1) and is of interest as a stand-alone project. This topic is not typically explored within games. Having the opportunity to create a unique product which addresses these issues through the media of gameplay is an appealing topic to cover.

The motivation in this project comes from the appeal of creating a unique product within the gaming market, while also touching on an important educational topic. The idea of creating a fun product for people to play, while also teaching the users an important educational topic, is a driving factor for wanting to complete this project to a high standard.

Similar Systems

To start the research process for the project, some background research into similar games was undertaken.

The research started by looking at some mobile applications which were related to water management. The research consisted of searching “Google Play” and “Apple App Store” in order to find a product that was similar in concept but offered a unique design that could be used as inspiration for the design of the project.

The research showed that there were applications available that include functionality for the management of water properties, however, these applications are for business purposes and are not considered as games. This meant that further research into finding “water management games” would be needed.

After a long search on the app store, the following games were found:

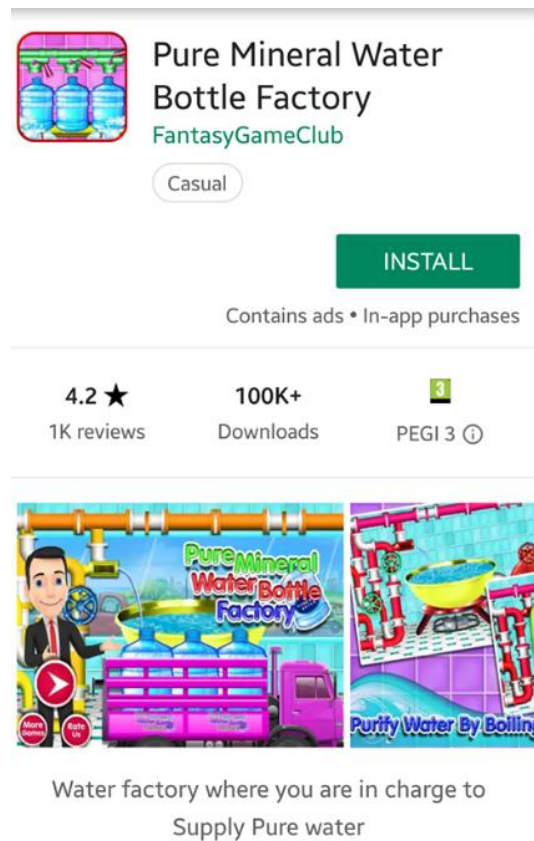


Figure 1: The above picture shows a game which focuses on the user cleaning and marketing mineral water for customers.

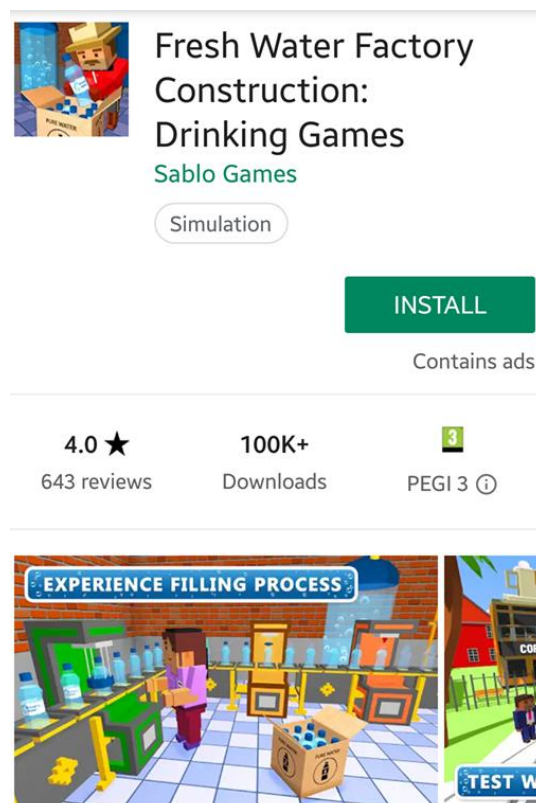


Figure 2: The above picture shows a game which focuses on the user picking a location and building a water factory, in order to make the most money possible.

These games have a similar idea to what the project aims to achieve, by having gameplay centred around cleaning water and shipping it to customers. However, these games were not meant to be educational and focus on different versions of gameplay.

For example, Pure Mineral Water Bottle Factory (2) focuses on shipping as much clean water as possible to customers by boiling water to remove any harmful chemicals. This process could be adapted to have an educational message.

Whereas, Fresh Water Factory Construction (3) focuses on the aspect of building a perfect water cleaning factory by having the user construct a water factory. The user can choose what goes into their factory to try and make the purest water possible. This game does include a mobile water lab in which, the player can analyse the water quality to find out what is needed to clean it. This gameplay aspect is very useful to research, due to the concept being similar to what the project is trying to achieve.

From reviewing these games, the structure and gameplay of these apps can be useful for adaption into a new piece of software with an added focus on an educational aspect. The games researched do not have an educational aspect, so the gameplay present within these games is important to the consider.

Technology Research

As the overall description of the project did not specify what technologies could be used in creating the project some research into which available programming language would suit this project was undertaken.

During this research different programming languages and platforms were considered. The languages considered were:

- C# using Unity
- HTML and JavaScript

The first platform that was considered was Unity (4). This platform was considered because Unity offers a wide range of libraries that can be used to help create software which is more gameplay focused using a 3D art style. Unity was not chosen because the reliance on having additional software installed on systems limited the flexibility that is needed for a public engagement project.

The second platform that was considered concerned the use of HTML and JavaScript. These languages were chosen as the languages of choice for the project from early on into the development.

The reason HTML and JavaScript were considered was dependent on a few factors. The first being that JavaScript is a flexible programming language which offers a range of useful functionality and assets which can be used within this project.

HTML and JavaScript offer good accessibility as languages. HTML does not require any additional software to help run and maintain, all that is required is a web browser. A game made using this

language is not limited by software availability, which lends itself to the “public engagement” aspect of the project, as the game can be transported via pen drive.

Taking the accessibility and flexibility of HTML and JavaScript into consideration through this research allowed for favourable consideration for using these languages over other available languages such as C# and Java. This is due to C# and Java being limited by the requirement of additional software.

Functional Research

The project topic could be taken in many different directions regarding gameplay. Due to this, research into how a game with this topic could work was undertaken.

Potential ideas and functionality that could be used within the game are listed below:

- Adding chemicals to a water supply
- Presenting the water quality to the user
- Sending clean water
- Discarding dirty water

While the project topic is vague and gives a lot of “breathing space” when deciding what direction to take the game. It meant that deciding and prioritizing which functions would best fit the topic of the project while being fun to play was something that needed a lot of thought.

Due to this, it needed to be decided what the aim of the game is, and how this could be achieved through gameplay. It was crucial at this stage to have an idea of what direction the game could take, when conveying the educational message. The game could be a simulation of cleaning water, a level based competitive game or an interactive story game.

From this research, it was decided that the best option for this project was to make a simulation game. Having a simulation game that teaches the user the steps required to clean water, would be a good theme to explore for this project.

The research made from looking into potential functionality and the direction the game could take, allowed for the concept and direction of the project to be solidified early on.

1.2. Analysis of research

Taking into considering the information gathered from the research of the project, some considerations and conclusions were made.

The first conclusion made was the use of HTML and JavaScript as the languages to be used for the project's development. These languages were found to be the best option for this project, due to the accessibility and the range of functionality that the languages offer.

The research helped rule out any consideration of other software such as Unity. This is due to the dependencies that Unity requires, which may not be available at the events that this project will be shown at. This was the main reason that after the research was undertaken, Unity was not chosen over HTML and JavaScript.

Alongside the conclusion of using these languages, a consideration for the JavaScript plugin, THREE.js was made and was later chosen to be used alongside JavaScript.

THREE.js was chosen as the preferred plugin because of the option to use 3D objects and animation functions to give a new level of depth and possibility for the direction of the project. user offers features that could be useful in the development of the project, such as, water simulation and object-specific data.

Functional Requirements

When undertaking the research into what the game could be like and how it would function, some features that should be in the game were suggested by the project supervisor.

The features that were chosen to make the core functionality of the project were listed and discussed within **Appendix A**.

The features that were chosen for the gameplay were reasonable choices based on what would happen in the real world to clean water.

Some of the features decided upon, could be taken in a different direction to what is specified within the feature list. However, the features that have been planned at this stage of the project will be designed with the intention of being within the final system.

The majority of the functionality can be translated into button interaction, which lends itself well to the accessibility of the game. However, the game can be seen as simple due to the lack of technically challenging features and requirements.

The storage of points and player names needed additional research, as there are several ways that this could be implemented. Cookies and database storage were considered for the storage of the specified values, but it was decided that cookies would work better for this project.

It was discussed that data base storage could also be used to store the scores. However, this method was scrapped, as it did not seem necessary to create a database to store scores which would be ultimately deleted after a certain time.

The idea behind this system is to allow the user to save the score that they got, which could then be added to a high score table. Therefore, a data base to store every score that had ever been set, did not seem necessary to create. However, the idea of having some scores that would be present for new players to aim for when the game is first loaded, was an idea that needed further exploration.

If the chosen features were implemented within the given project time, additional functionality would be considered based on user feedback and tests.

Non-Functional Requirements

Additional browser support

As the game would run in a web browser, some consideration into different browser types and screen resolution would need to be considered. Research done on different web browser showed that the properties of individual web browsers such as Google Chrome and Firefox are mostly similar but not identical.

The sizes of assets on the screen and the functionality of the features in different web browsers would need to be considered and researched, to make sure that the game would be able to function, no matter what web browser is being used.

Security

Storing harmless information in cookies such as a random user name might not seem like a security concern, however, its security needs to be considered. Measures need to be taken so that code injection or other malicious code cannot be entered into the cookie text fields when designing the storage system for the user points.

Mobile compatibility

As the game will be available through a web browser, it is to be assumed that the game would be tried on a mobile device. Mobile devices have different screen resolutions and functionality. The inclusion of mobile compatibility can be explored in the form of making the game functional and visible on devices of lower technical power.

1.3. Method

Initially, it was considered that the game would be developed following the agile approach of Feature Driven Development (FDD). This process would involve planning out what features would need to be included within the game from the start of the project, after which they would be incrementally integrated into the overall design of the game.

The idea was to work on a feature for each of the two concepts of the game, being the educational and gameplay aspects of the project, each iteration and then implement the features into the overall system at the end of each week-long iteration.

This process of development was modified so that the features that help create the educational and the gameplay aspects of the project could be developed again after the original development. This means that it was adapted to allow the development of the features together to create one whole feature, after the initial split of feature development.

This was chosen as the method of choice because, FDD can lead to more flexibility, by allowing the features to be updated and changed at any step of an iteration. This is useful for this project as it allows for alterations to be made on features that do not follow the initial plan of the program.

However, one of the risks of this methodology is that FDD can lead to the lack of an overall architecture of the software. As the features are created separately to the overall system, there can be a problem with the architecture, in the form of the program becoming disjointed.

The adaptation of the FDD methodology that was chosen, to allow for extra iterations in which the features could be integrated and tested together before the features were implemented into the final system. This was chosen with the intent to keep a solid architecture for the project.

A form of version control was used for this project. GitLab (10) and WordPress (11) were used to keep track of the projects progress.

2. Design

2.1. Overall Architecture

The architecture for this system would be comprised of separately developed features, which are then brought together to form the final system. As the project followed the feature driven approach, the features would have needed to be designed upfront and implemented separately from an overall system.

This would mean that the required features discussed in **Appendix A** could be designed initially, with a design plan for the system being made later.

The features of interest discussed within this section are:

1. Analyzing the water quality
2. Altering water properties
3. Score System
4. User Interface

The list above represents the features of the game that will be discussed within this section. Each of these features were briefly discussed within Appendix A, but needed to be properly designed before being implemented into the system.

2.2. Function Design

2.2.1.1. Water Properties

This section highlights the features that were designed to have an impact on the water properties within the game. The features concerning Flocculant, Turbidity and the overall Water Quality is discussed within this section.

2.2.1.2. Flocculant

Flocculant is the name given to the chemical that helps bind dirt in the water so it can be caught by the filter, so the water can be cleaned.

The Flocculant feature that will be used within the game follows the same idea. The aim of this feature is to have the user, manipulate the Flocculant levels in the water, to see what effect this has on the water quality discussed in section 2.2.1.2.

There are a few ways that this feature could be designed. The first considered design was to have the user input a value for the Flocculant manually, which then returns a string which indicated the water quality. This design was scrapped early on because it did not offer enough gameplay aspects to be considered worthwhile. Due to this, another design was considered.

This design focused on the concept of having the player decide how much Flocculant should be added to the water, by using buttons. To help illustrate this idea, the following diagram was made:

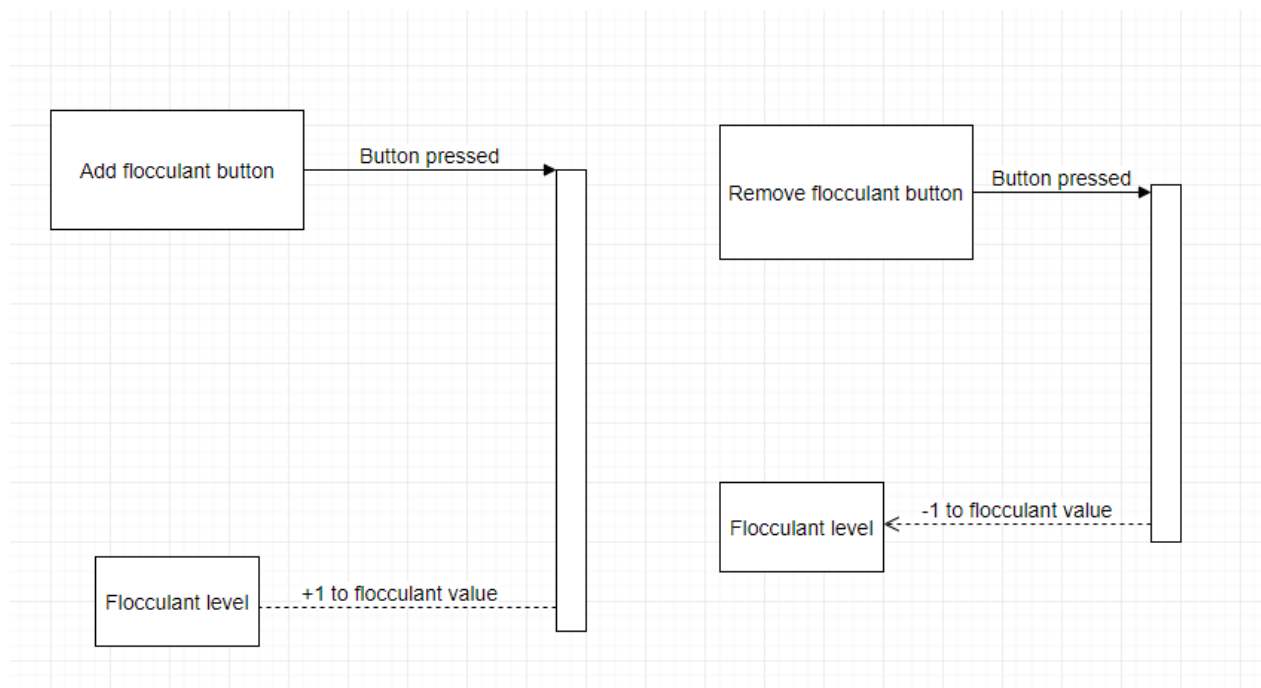


Figure 3: This use case diagram shows the desired functionality behind the buttons that the user can interact with.

The above diagram was created to help visualize the interaction that would occur when the button is pressed. The diagrams show that when the button is pressed, a value of 1 is added to the Flocculant value when the respective button is pressed, and the opposite occurs with the remove Flocculant button. For the buttons to help illustrate the functionality, it was decided that the buttons should be designed to look like, up and down arrows. A concept design of these arrows is shown in figure 4.

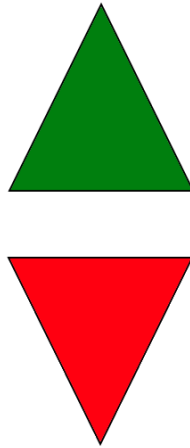


Figure 4: Initial design of the buttons that represent adding and removing Flocculant.

2.2.1.3. Water Quality

The water quality is a Boolean that is designed to return true if the water quality is good, or false when the quality is bad. The function that helps calculate if the Boolean should return true or false, can be complex on the surface.

This function was designed with the idea of having an external variable decided on what the Boolean should return. This variable is called the Turbidity. The Turbidity within the game refers to the amount of dirty within the water.

The Turbidity value was designed with the idea of being randomized between two values after set intervals of time. This was decided, to add more complexity to the game, with the water Turbidity changing constantly throughout the game.

This would prevent the water quality from remaining on the same status throughout the game. In addition, this would make the game more interactive, with the player constantly having to keep the water quality good.

The effect that the Turbidity value has on the water quality Boolean is simple. If the Turbidity is below 5, the Boolean returns true, if it above 5, it returns false. This illustrated in figure 5.

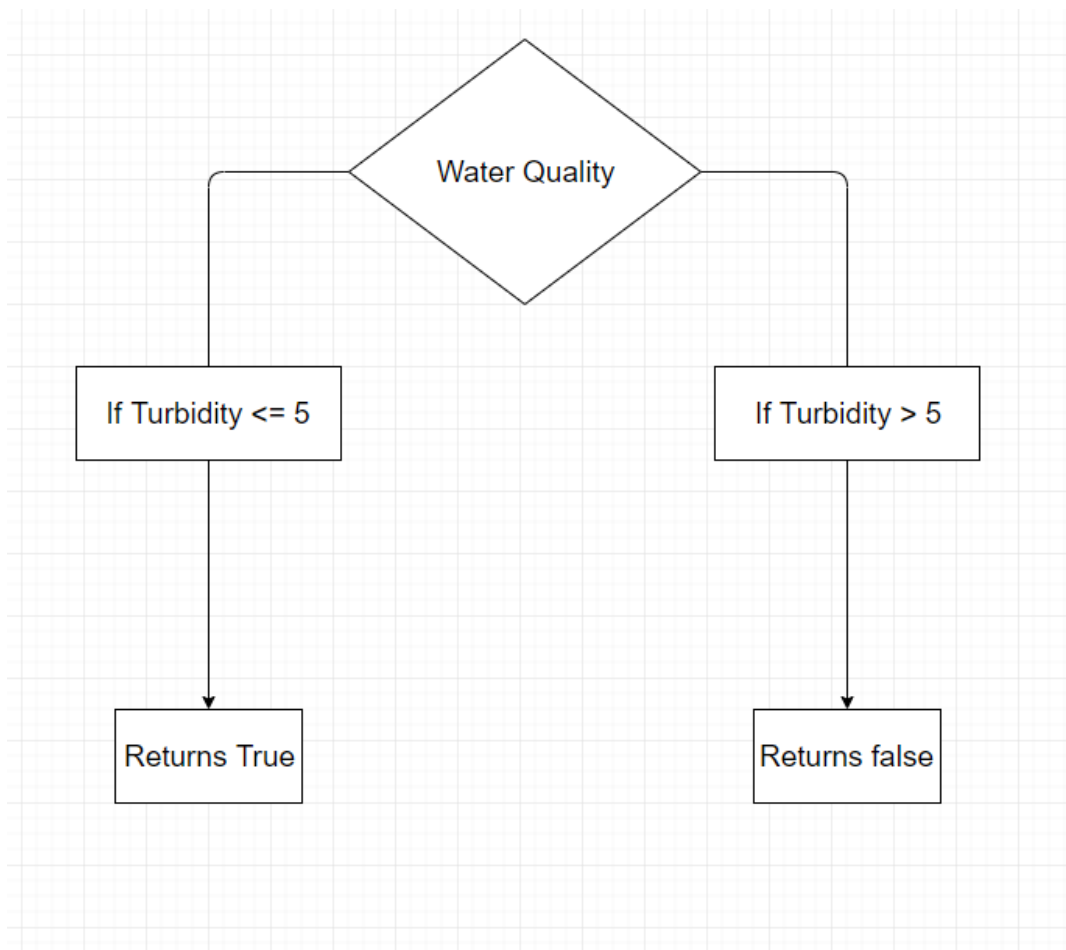


Figure 5: Flow diagram showing the interaction between Water Quality and Turbidity value

The above diagram illustrates the conditions that calculate the outcome of the water quality Boolean. To help visualise the quality of water to the user, a string was created that will print “good” when the Boolean is true, and “bad” when the Boolean is false.

2.2.2. Filter Properties

Analysis

One of the main features for the game is the filter present within the game. The filter would act as the object that would change the water quality to allow points to be gained.

When analyzing how to implement the filters, some decisions were made into what effect the filters would have on the gameplay. It was decided that the variables for the Flocculant discussed in section 2.2.1.2 and the Filter Load variable, were to be linked to the filter object.

This process could be carried out by using object user data. User data is a THREE.js method which allows variables to be assigned to specific objects within the scene. It was decided that this method would be used to link the Flocculant variable and the Filter Load variable to the filter object.

This was chosen because in the initial concept for the game, there were to be more than one filter. The use of user data allows for object-specific data to be mapped to an object, which could be manipulated separately. This method is useful for allowing different values to be used on different object, but this method can be used when one object is used to secure important information.

Design

The Filter Load variable mentioned above, is a key component to the gameplay, as it is used to manage the amount of flow that is being produced.

The filter load variable was designed to have a value between 0 and 100. This variable would start at a random number between 0 and 80 each time the game was played and would have the value of the Flocculant added every 2 seconds. The concept for the idea of having the Flocculant affect the value that is added to the filter load is described in greater detail in section 2.2.3.

This design was chosen and was used in the final design as it offers more complexity to the gameplay. Additionally, having the amount of Flocculant within the water, dictate how fast the filter would get dirty, mimics real-life scenarios and therefore, helps strengthen the educational message.

Final Design

This design worked well for the gameplay when implemented early on, however there was a problem with the system, when the filter reached 100%. As the design currently suggests, when the filter got too dirty, no flow was being produced and as a result no water was being sent to the tank. This caused the problem with the user running out of water and the game ending faster than expected.

A redesign of this feature added the option of a button which would reset the filter load to 0. This was a good addition to the design as it allowed the user to clean the filter manually. This however, needed an added constraint of costing points to perform, so that the button could not be over used.

This was implemented into the design, because it helped enforce the message of, "cleaning water costs money". By having the user spend points to clean the filter, the educational message can be translated better into the gameplay.

2.2.3. Score system

Analysis

It was decided from the early stage of development, that a scoring system would work for the game. This was due to the competitive nature of games within the industry, so it would be a good idea to incorporate a scoring system into the project.

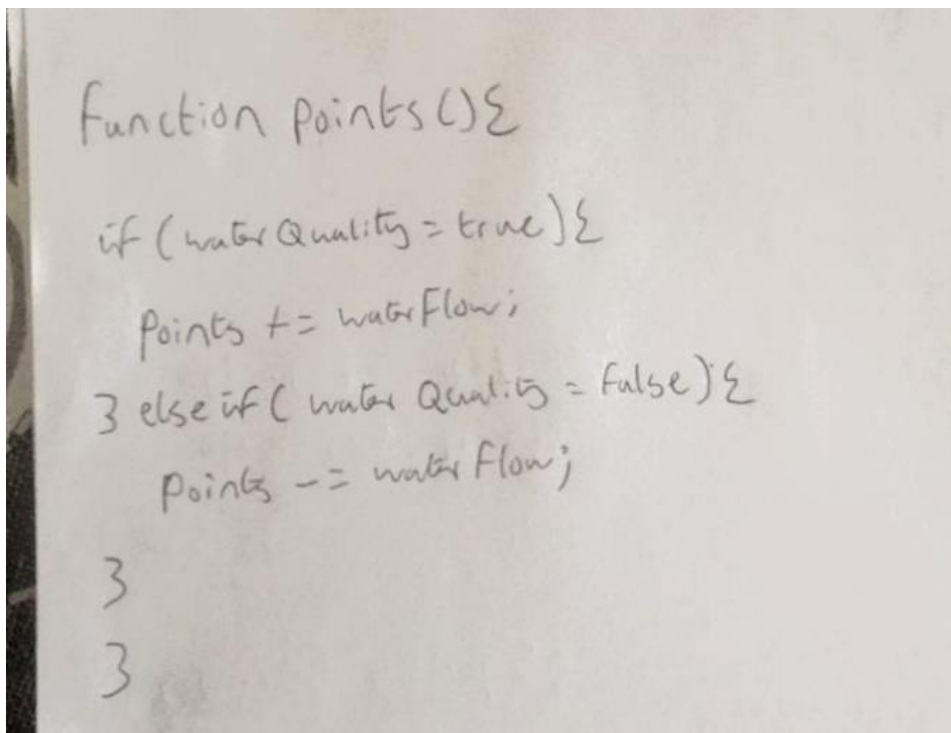
While looking into this feature initially, there were some questions on how a scoring system would work within the game, and how it would be ultimately shown off to the users of the game. If a scoring system was implemented into the game, it would need to add something to the player's experience.

The initial idea for the points system was to have the user gain points from sending clean water to customers. This idea could work well within the game, as it solidifies the aim of making as much water as possible.

Having the user's score linked to the amount of clean water made throughout the game was chosen to be the formula that would be designed and implemented later into the project. The same system could be used for having the player lose points when the water quality is bad.

Design

With the concept of having points gained when clean water is sent to the tank, a list of what the function could include was created.



```
function points() {  
  if (waterQuality = true) {  
    points += waterFlow;  
  } else if (waterQuality = false) {  
    points -= waterFlow;  
  }  
}
```

Figure 6: Initial design of the function that would calculate if points were gained or removed from the players score.

The above picture shows a rough design for the function. This design focuses on performing different actions depending on what the water quality is. The if statement checks to see what the water quality is and then adds the value of the water Flow to the points variable if the quality is true (good) and removes the value if it is false (bad).

The design for this function is based on simple logic. If the water quality is bad, then the player would lose points for not acting on the change in water quality and if the water quality is good, the player would be rewarded.

The idea of using the water flow variable to calculate how many points the user would get was implemented into the final design of this feature. This function could be called through a “setInterval” function, so that points could either be added or removed every 2 seconds, to keep the game from feeling stagnant.

This method was chosen because it gets the user to think more about the amount of water that is being produced. The added challenge of keeping the filter clean on top of having clean water, prevents the game from being too easy and boring to play.

2.2.4. Saving scores

Analysis

With regards to the scoring system, it was decided how the points will be generated and how that would affect the gameplay. However, the problem of how to store the scores for the users needed to be analyzed.

The scores generated through the game would need to be presented to the user in a meaningful way, for it to be significant. The suggestion of a high score table was considered when analyzing this problem. A high score table was found to be an optimal solution to this problem, due to a score table being the industry standard way of displaying scores to the game players.

Another way of displaying the scores would be to show the user what they scored when the game ends. This was found to be decent for a solution, however, this method lacked the competitive aspect that having a high score table would generate. A score table would benefit the gameplay by adding replayability to the game alongside a competitive element.

When analyzing this problem, in section 1.2. it was decided that cookie storage would be a good option for score storage. However, some research into how cookie storage could be designed was needed.

Storage Research

A cookie is a small text file which is stored within the user’s computer for a browser session or permanently on the user’s hard drive. A cookie can store information based on user preference, such as shopping cart details. For this project, a created cookie would need to store the score of the player and a name for the score to be linked to.

During the research about cookie storage, it was found that using cookies to store data is good for temporary data, that is not overly significant. This would suit itself well to the project, as the game would only need to store the scores temporarily while being used for its initial purpose.

A cookie can be created by using the “document.cookie” method that was found during the research process on W3Schools (5).

The research ultimately found that cookie storage could be implemented by creating a simple cookie with the method shown above, and then passing in the game score value. This process could be a useful way of storing the score for the player. An addition of allowing the user to input a name could also be considered.

Design

After the research ended, it was decided that cookie storage would be the best option for storing the score data. Due to the nature of the project, the scores gained would not need to be permanently stored and could just be deleted after the game had been closed.

The design was chosen for storing the points concerns, giving the player the option to save the score that they got once the game has ended. The initial design consisted of a button which when pressed, would prompt the user to input a name. Once the name was inputted a cookie would be created with the game score and then its content would be printed to the screen.

For this design to work, a function that creates a cookie, while prompting for a name would be needed. This function would use “document.cookie” to create the initial blank cookie, while the JavaScript method “prompt” can be used to allow user input.

Prompt is a function that allows the creation of an input field for the user, which can then be mapped to a blank variable. A function that allows for this behavior could be designed like this:

```
var user = prompt ("Enter a player name:", "");
```

This piece of code would use the prompt method to allow the user to input a string. This string would then become the value of the variable “user”. Once this code has been executed, the “user” variable can be pushed into the created cookie. This design could be improved, by pushing the score variable into the cookie at the same time.

This design could have some potential security concerns if the user can input any characters that they want. This is a problem which would need to be followed as the design is implemented, but the principle of pushing two variables into a cookie could work for the final design of the feature.

2.3. User Interface Design

2.3.1. Home page

Analysis

The first visual feature that would need to be designed considers the games home screen. This page would be the first thing that the user of the game would see and therefore would need to be visually appealing while having relevant information about the game displayed to the user. The look and placement of the information would be changed throughout the project, therefore an overall design of the home page and what functionality it would deliver would need to be designed upfront.

Since the game could be complex initially, it was decided that some information about the game would need to be displayed to the user. This would require some research into how typical “start screen” help to convey information to users while being visually appealing.

It was decided that the design of this feature could not be done initially and would require, constant updating and changes as the project evolved. This is something that can be done due to the nature of Feature Driven Development. The initial design of the “home screen” could be made and implemented, but as the project continued, further updating can be made to help represent the overall aesthetic of the game.

Research

The first task that would need to be researched was, how other games represent information to their users. This research would initially focus on what the best approach of showing game data to the user would be, this research would not concern what information would be shown. To start, research into “AAA” game titles would be taken. This research would aim to give insight into how “high end” game developers represent information to their users, so inspiration could be gained on the subject matter. The game that was investigated was, “Assassin’s Creed: Odyssey” (6).

While looking into how this game managed the display of information to the user it was noted that the game and others of the same genre, tend to use buttons that link to other pages or open pop-ups as seen in figure 7.



Figure 7. The menu system that is used within Assassin’s Creed Odyssey, shows an expanded menu system that provides an easy to follow UI.

The game researched, also has some “tips” displayed in the loading screen of the game and throughout the game itself to help remind the users of the information that they previously saw. This would be an acceptable way of showing the user information.

In conclusion, this research helped aid the decision of having the game information displayed upfront to the user, to help remove any unnecessary confusion.

Design

Initial designs

The overall design for the home page is simple, with a limited number of buttons for the user to interact with, and some on-screen game tips. The page would consist of a button which starts the game, the tips for the user and a top 15 high score table, which is passed the scores from the game. The score table design was discussed in section 2.2.4.

A simple diagram of the user’s interactions is shown below:

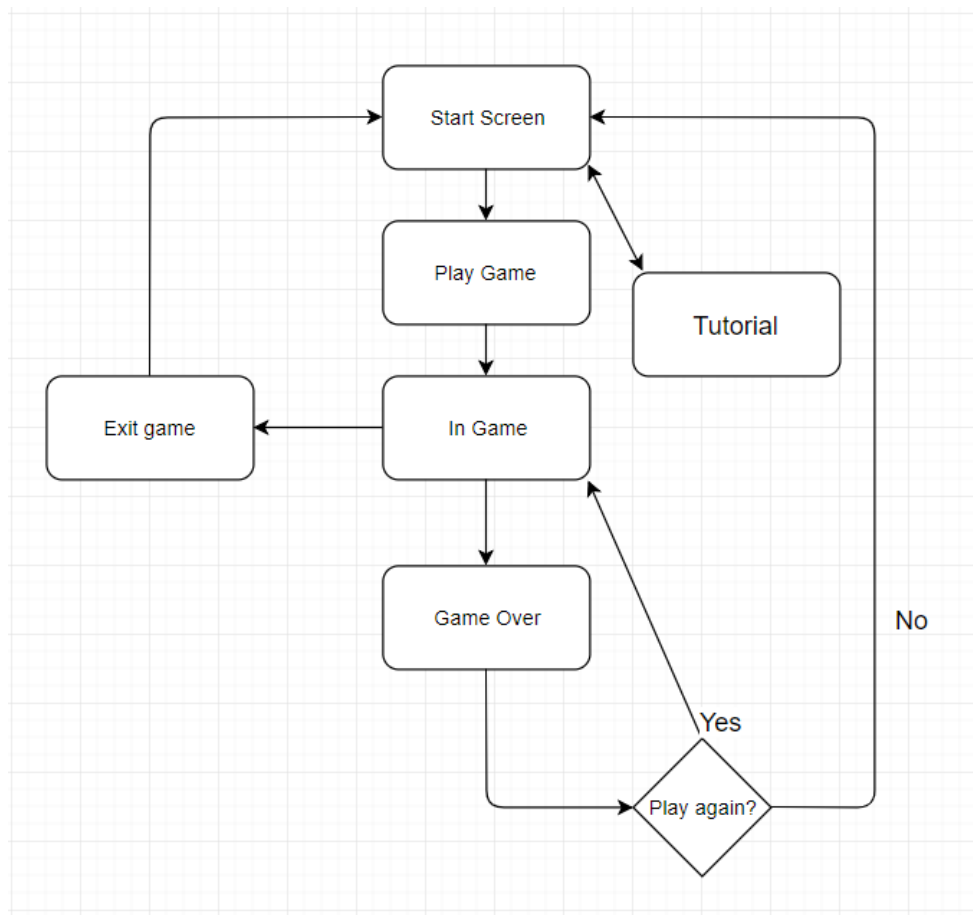


Figure 8: Flow diagram to show the interactions and flow of the games menu system and behaviours between buttons.

The above diagram shows the interaction between the buttons on the home screen, and how navigation back to the screen can be achieved. Within the game itself, there will be a button which

allows the user to “exit” the game back to the home screen. This can also be done with the planned “Tutorial page”.

After the consideration of how the home page would function and add to the player’s experience, an upfront design of the home page was created.

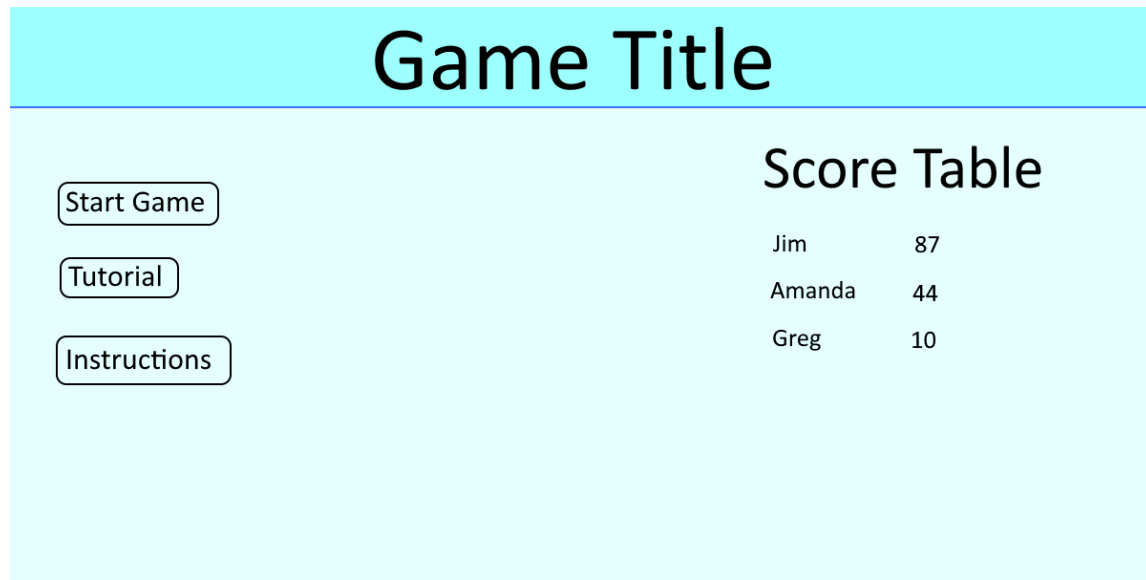


Figure 9: The initial design for the game home screen is shown here. This picture was created to help visualise how a final design for the home screen could look.

The above picture depicts the first initial design for the home screen. The initial design was created during the research process in order, to get an idea of how the page could be designed based on the research taken. This design relies more on the use of menu buttons to try and keep the game in standard with traditional games.

Additionally, the score table was designed to be visible at all times, so that it could be easily referred to. Some mock values were added to the high score table to see how the table could be formatted.

Final designs

After the research had concluded on how traditional home pages are created, a final design for the home page was created. This design would scrap the idea of having an instruction button, in favour of displaying the game information upfront to the users. This was decided to limit the amount of confusion which could occur by having too many buttons.

The design created is shown below:



Figure 9: The final design of the home screen was designed using an art program and features an updated design based on the decisions made later.

The design created for the final product features the same overall ideas of the initial design, with buttons to start the game and to start a tutorial and an updated colour scheme.

However, the main difference is the inclusion of a “Game Information” section on the home screen. This section will list the overall aims of the game and how the game functions in an easy to follow list.

This was chosen, as it would give the user a secondary option to work out how the game works. The aim of this decision was to remove the reliance on viewing the game tutorial for users who just want to play the game upfront.

2.3.2. Game Screen

Analysis

The UI for the game screen is the most important factor for the game. This is feature was very important to design correctly because it would be the most important scene for the game, therefore, it needed to be appealing for the user.

When analyzing the potential designs for the UI, some inspiration from the systems discussed in section 1.1 were taken. It was decided that the overall design for the game would focus on a 2D perspective opposed to a more 3D design.

This was decided as 3D could be confusing for younger users with a focus on screen depth. 3D could also have some accessibility issues with some users not being able to see the difference in depth, which could cause issues with the game functionality. In contrast, the 2D design would allow for a simple and streamline design which would not be overly complex to navigate through.

2D was chosen over 3D for the previously stated reasons as well as given the game a “cartoony” theme. If the scene perspective was to be 3D and the buttons on the screen were to be 2D, then the overall aesthetic for the game would not be consistent and could result in an unusual game experience.

Once the game perspective was chosen, a decision on how the game would control and function would need to be discussed. There were many options available for control systems for the game, such as keyboard controls, mouse controls or a mixture of the two.

It was decided that mouse controls would suit the theme of the game well. This is due to the user having to manipulate values discussed in section 2.2, to gain points throughout the game. Mouse controls allow for the use of buttons, which would be implemented to visualize the user changing the game values manually.

The overall decision for the game screen GUI was to have a 2D point and click game, which features visual prompts to show the user what effects the buttons had on the game.

Initial Design

The initial design for the game screen consisted of the usage of pop-ups that would appear when a filter within the scene was clicked on. This design used the method of “Raycasting” (7).

Raycasting is a method in THREE.js which uses the mouse and camera vectors to allow for individual object selection within the scene. The purpose of Raycasting is to allow the mouse to work out its location within a 3D space, which can be used for object highlighting or moving an object.

The use of Raycasting in the initial design was to allow for interaction with objects that had object specific pop-ups and edge highlighting. Raycasting was essential for the initial design because it allowed for the desired functionality of clicking on the filter to function correctly.

A mock design of how the menu for the pop-up system would look was created and is shown below:

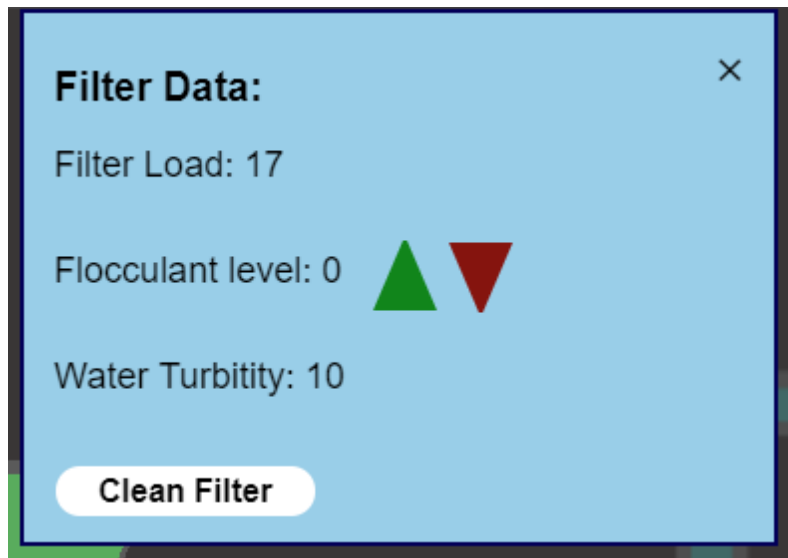


Figure 10: Early design concept of UI featured within a pop-up box, that appears when a filter is clicked on.

The above picture shows the design used for the game UI. In this picture the box that shows the information related to the filter would only be visible when the filter had been clicked on. The purpose of this design was to streamline the UI for the users when more filters were added to the game.

The idea of having the information visible when the filter is clicked on offered more interactive elements to the game. It allowed more exploration for the user in the form of clicking on the individual objects in the scene to find out what they can do.

This design did have some issues. The first of which, made it hard to understand which objects could be interacted with and what was background art. This was altered by having more noticeable object highlighting, for the objects that were interactable. Edge highlighting was mostly successful, but it did not remove this problem completely, with the object still being difficult to distinguish.

The main issue with this design, however, was the lack of direction. Based on user tests, this design did not help new players understand what needed to be done to progress.

Hiding the information from the user was not optimal in most cases and led to confusion. The information received from these tests, led the design of the home screen to be altered in later iterations.

Second Design

The second design for the game GUI kept the same information and features for the user to explore but removed the Ray Casting and the pop-up functionality. The new design favored having the information on the screen at all times for the user. Having the game information constant, allowed for added visual aid for this design. This included updating the text so that the user could see the effect of the button presses and some visual alerts.

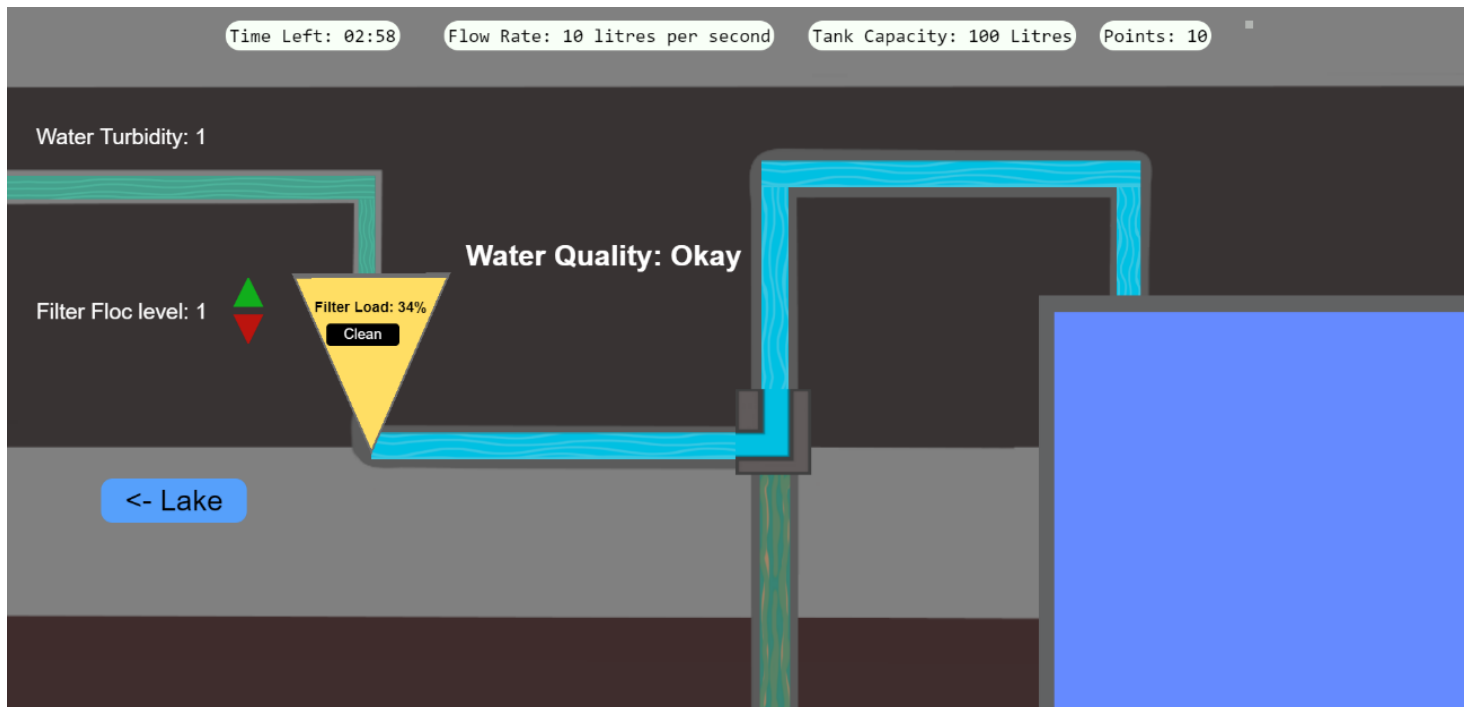


Figure 11: Second design concept for game UI shows the game data being visible on the screen without the use of a pop-up box.

The above picture shows the second version of the GUI. The picture shows the information that was once located on within a pop-up, is now available at all times during the game. This design allowed for a better experience for new users to get familiar with. Alongside the changes to the information location, a timer and points were added to the design.

These two values were added to the scene, to bring the project more in line with traditional games by having the timer and the points visible to the player.

This design also includes the added feature of the holding tank. The holding tank is present on the screen to help visualize how much water has been collected during the game, alongside the water pipes, which help show the user which way the water is flowing. The water pipes were designed to change colour based on if the water quality was good or bad.

This design worked well when implementing new features and offering an easy to follow user experience. However, there were some elements that could be added or improved upon. Due to this, another design was created to fix these issues.

Final Design

Not much was changed between the second design and the final design that was produced. A picture of the final design is shown below.



Figure 12: Final design for the game UI which shows added text labels and the inclusion of in game tips.

The final design that was created shows the inclusion of in game tips and additional text labels. The former designs were not descriptive enough to what everything on the screen was. This design fixes those issues, by labeling the important features within the game scene.

The addition of game tips was a decision that was made late into the design process of the UI. It was decided that having tips for how the game was played would help new users that had forgotten what to do.

This UI design included animated water textures for water pipes and the addition of a weather icon. These changes are discussed within section 3.4.1 and 3.2.1 respectively.

2.3.3. Tutorial Page

When testing how the game functions throughout the early iterations, it was decided that a tutorial page would be needed to help explain the controls and game logic to the new players.

Analysis

Traditional tutorials for games, have the user play a section of the game while information is shown to the user. This was the approach that was taken with the initial design of the tutorial page. The tutorial page needed to be useful to the player, but not necessary to look through in detail.

A balance between information and design would need to be considered. This was an important topic to analyze because there are many ways a tutorial page could be made. A tutorial could be plain text, that just tells the user what to do, but could also be an interactive experience.

Due to the nature of the project and the methodology chosen, having the freedom to change the design and aim of the tutorial page as the project evolves, is very important. The tutorial will need to explain to the user, the current controls and functionality of the game. It would feel unnecessary to have a tutorial page, that was designed upfront and did not evolve with the game itself.

Due to this, the initial analysis of the feature could only bring forward the functional goals of the page. These goals include conveying the aim of the game to the player and the controls of the game.

Design

Initial design

The initial design for the tutorial page was simple, in the fact that it was not given its own separate page within the first instance of its development. The first design for a tutorial consisted of an information button on the home screen that listed the game objectives to the player.

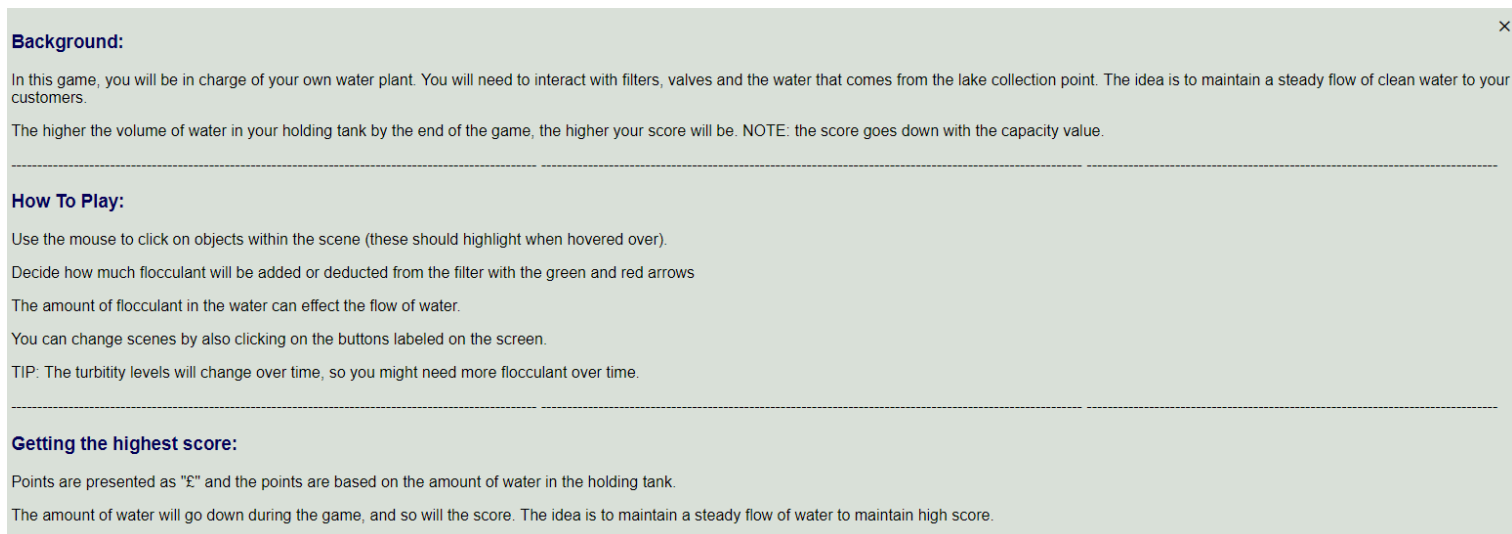


Figure 13: First design for the tutorial consisted of an information pop-up which shows information about the game in a text-based format.

The picture above depicts the initial design for the tutorial. The main issue with the design for the tutorial was the lack of interaction and worthwhile actions. This design felt too simple and did not offer a fun and interesting experience for the player.

The idea of having a button that presents the information to the user at its core, acts a tutorial for the game, but is not interesting to experience. The biggest issue with this approach was having the information displayed as just text, was boring for the user and led to the information being forgotten. A redesign and further look into what the tutorial page needed to convey was required.

Second Design

The second design for the tutorial was a big expansion of the initial design for the tutorial, with a separate game screen being created. This design was chosen to get the user familiar with the layout of the actual game while having “information buttons” available if further information is needed.

The initial design for the tutorial page was to be the same as the final design that was discussed in section 2.3.2 with some added buttons. A simple mock of how the information buttons could be linked to the game objects was created.

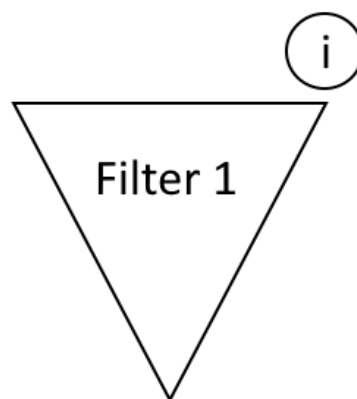


Figure 14: Mock design for the inclusion and location of information buttons. These buttons would open a pop-up displaying relevant information.

The above picture shows that the concept idea for the tutorial design was to have small information buttons positioned next to the objects of relevance. The information buttons would trigger a pop-up with information about the object closest to them when clicked.

It was decided that this design would follow the information button concept and have an individual button present for each object of interest. However, this design would only require the user to press on the info buttons, without having a “hands on” experience of the game functionality.

Final Design

The final design for the tutorial kept the same overall design of the second iteration, however, focused more on gameplay button interactions.

It was decided that some of the information pop-ups should be shown to the user when they click on one of the gameplay buttons, such as “buying water”.

A method was created for the tutorial page which would monitor the number of clicks used on the gameplay buttons. This was an important method to produce as it could allow the pop-up information to only appear after the first 2 clicks on the button.

This was the decided method because it would allow the user to have two opportunities to read the information given, and then an unlimited amount of opportunities to see the effects the buttons have on the game.

Every time a button is pressed, the relevant pop-up appears and adds 1 to an empty “clicks” variable. The pop-up function sets the display attribute for the pop-up to “hidden” if the clicks variable gets above 2.

This function is not optimal as there needs to be an individual “clicks” variable assigned to the “Flocculant” and “clean filter” buttons to allow this functionality to perform correctly.

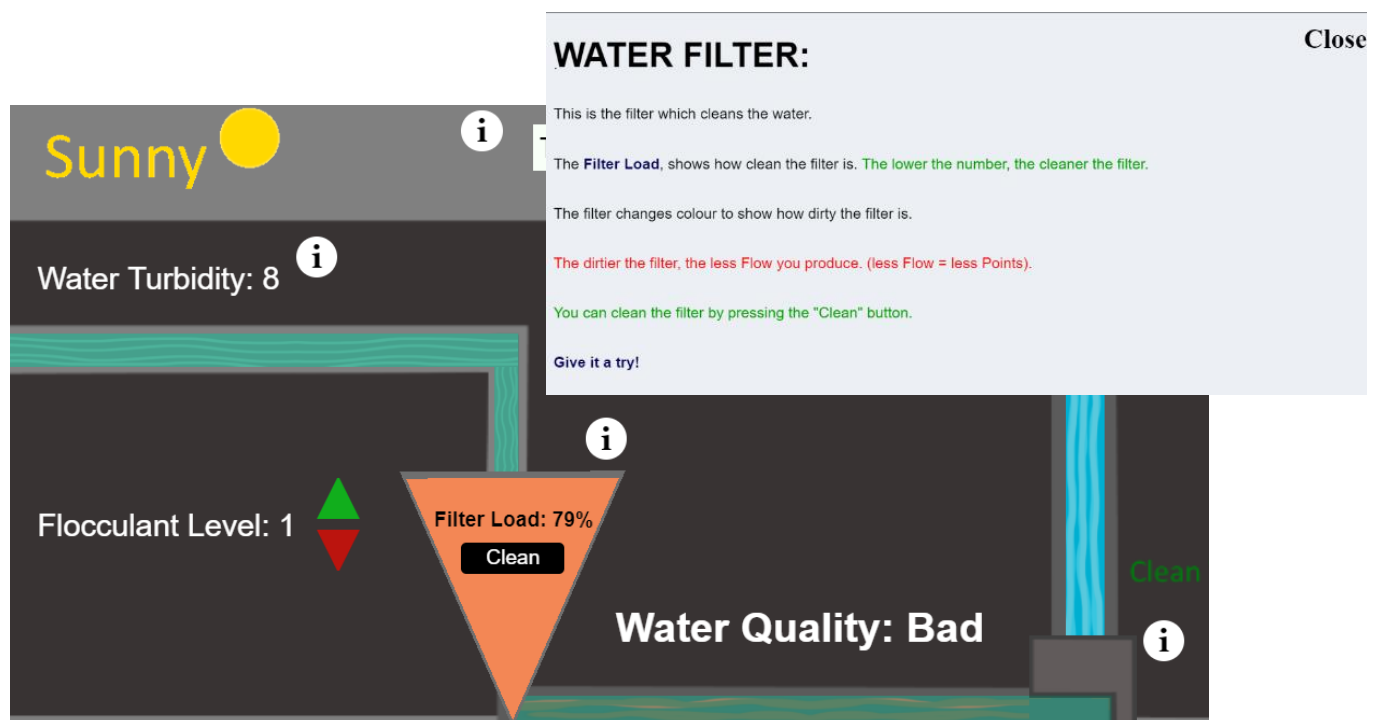


Figure 15: The final tutorial design shows the inclusion of the “i” button and the respective information pop-up that appears when the button is pressed.

This above picture shows the final design for the tutorial page and the pop-up that is presented to the user. The information layout in the pop-up is split into a list of the most important information related to the object functionality. This method was chosen as it does not over complicate the system for the user, while still documenting the important factors within the game.

3. Implementation

3.1. Function implementation

3.1.1. Weather randomization

The feature of having a weather system was not initially designed to be a part of the system but was later implemented to allow for diverse gameplay.

This feature consisted of having a variable which would be used to calculate the randomization of the Turbidity value discussed in section 2.2.1.2. The initial design for the Turbidity was to have the value randomize between 1 and 10 every five seconds. This design worked initially, but it could cause the game to be too erratic, with the Turbidity jumping from a value of 2 to a value of 10 instantly.

Because of this, an additional function was implemented to help counter this erratic behavior. The weather feature would be used to help limit the range the Turbidity could randomize between. The new initial value for Turbidity would randomize between 1 and 5. However, another randomized value would be added to the Turbidity value depending on the weather.

It was decided that when the weather was equal to 1, a number between 1 and 3 would be added, and if the weather was equal to 2, the numbers would be between 2 and 5.

The weather was designed to swap between which weather was currently active. The picture shows a pseudo-code design of this interaction.

```
1  If the current weather is sun,  
2  the next weather in effect will be rain.  
3  If rain is in effect, then the next will be sun.  
4  
5  if (season === 1){  
6    season = 2  
7  
8  } else if (season === 2){  
9    season = 1  
10
```

Figure 16: Example of pseudo-code for the weather functionality. The example shows the use of a function to swap between the two weather types.

This process was implemented to help simulate different types of weather, with “season = 1” being sunny and “season = 2” being rain.

In addition, a different graphical icon would display to the user, to indicate the current weather. These icons would swap alongside the current value of the season, to show the user the change in weather.



Figure 17: Image icons that are used within the game to show the user what the current weather of the game is.

3.1.2. Filter Object

The feature of having an interactable filter object was implemented using the methods discussed in section 2.2.2. The filter object was passed the data for the Flocculant and the filter load using object user data as initially designed.

The filter load variable would be altered over time, based on the value of the Flocculant. This function was implemented by having the Filter Load variable increase by the value of the Flocculant each second using the timer function discussed in section 3.1.3.

This process did not create any initial problems until it was decided that the filter object would change colour depending on the filter load percentage. Due to the filter object being created in a separate JavaScript file and being passed into the game file as an array, a texture change could not be assigned to the object. As a texture change could not be executed on an object in an array, a separate object was created.

This object does not have any special attributes and was created to allow for the colour change. Once this object was created, the texture colour could be changed using, “object. material. color”.

This method allows the texture material of an object to change when used. This method was important to this feature, as it allowed the implementation of a visual indicator to the dirt level of the filter. Creating a new object to perform the colour change on is not an optimal fix to this issue, but the function implemented does allow this function to work as intended.

3.1.3. Game timers

Within the game itself, there are multiple timers active which all serve different purposes, but all function within a similar fashion. This section will discuss the implementation of the game timers. These timers did not have any prior design and were implemented as the project progressed.

The main timer shown within the game is the overall game timer. This feature was designed and implemented to show the user how long the users have remaining until the game is over. The game clock was implemented by creating a function which removed a value from an established variable. The initial variable was created to store an integer that would be used as the total number of seconds for the game.

This variable is used within a simple JavaScript function, which is designed to remove 1 from the variable each time it is called. To allow the function to be called each second, the JavaScript method “setInterval” is used.

setInterval is a method that calls a desired function at specified intervals. These intervals are measured in milliseconds. With the use of this method, the timer function could be called each second, to allow the desired functionality.

The additional game timers use the game method. The game timers used include:

- Points timer
 - This timer is used to time the points system discussed in section 3.3.2.
- Tank Capacity timer
 - This timer is used to reduce the tank capacity by 1 throughout the game, to help illustrate customers using the water available. This function calls from a tank capacity value and lowers its value by the desired amount when called.
- Filter load timer
 - This timer is used to have the dirty level of the filter increase over time. This timer is slightly different to the other timers, as the amount that the variable increases by is dependent on the Flocculant level. This method was discussed in section 2.2.2.

The additional timers were implemented in the same way as the overall game timer, as they all use the setInterval function but with different allocated intervals.

An issue that occurred with the timer functions, concerned the timers not stopping when the game had ended, which led to some unexpected results. This problem was fixed by having the separate timer methods cleared, using “clearInterval”.

This method would clear the intervals that the function would be called. To make sure this method occurred at the correct time, it was decided that all the timers would be cleared when the main timer value went below 0.

3.2. Point Systems

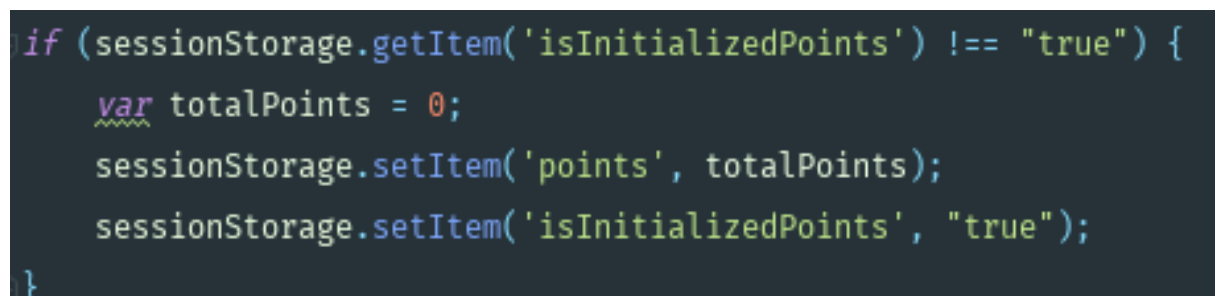
3.2.1. Preventing re-initialization of the points variable

The point system discussed within section 2.2.3, is a crucial feature for the gameplay elements of the project. Due to this, correct implementation of the designed feature was extremely important. The first element that would need to be created for the point system was a variable to store the points gained. This was done by creating a variable called “totalPoints” with a value of 0.

The totalPoints variable is passed into a JavaScript method called, session storage. Session storage is a JavaScript method which allows an allocated variable to be stored within the web browser, which is only deleted when the browser window is closed.

This method was chosen to prevent the score variable from being reset if the browser was refreshed during the gameplay.

However, this alone did not prevent the score variable from being lost. Therefore, an additional function was implemented to help counter this problem. The below image shows the function that was created.



```
if (sessionStorage.getItem('isInitializedPoints') !== "true") {  
    var totalPoints = 0;  
    sessionStorage.setItem('points', totalPoints);  
    sessionStorage.setItem('isInitializedPoints', "true");  
}
```

Figure 18: A sample of the code designed to prevent re-initialization of the points variable. This code shows the whole function that allows this functionality.

The above function uses a session stored variable called ‘isInitializedPoints’ to prevent the score variable from being reset on refresh.

This is done by initializing the points variable, session storage for the points and the session storage for the ‘isInitializedPoints’ variable within an “if” check.

The if statement checks to see that the ‘isInitializedPoints’ variable does not equal “true”. Due to the item being created within the statement, this check can only occur once.

The purpose of this function was to prevent the re-initialization of the points variable when the screen is refreshed, to prevent unusual game behavior.

3.2.2. Calculating points

The point system required the implementation of a function that would calculate how many points the user would gain, once the variable was established.

The design for gaining points discussed in section 2.2.3, discusses the use of the Flow variable to be used to calculate the number of points gained. This was implemented by having the value of the flow added to the current points value.

The Flow variable would represent the amount of water that is being sent to the tank each second. This variable was implemented to have an initial value of 10. The Flow value would start to reduce by 1 point when the Filter Load was above 60%.

The Flow variable was crucial to the overall point system of the game, as the value of this variable would be added to the points each time the function was called.

It was decided that if the Flow had a value of 8, 8 points would be added to the user's score. This was a simple concept to implement as the Flow variable could be added to the points variable dynamically, which allowed for more complex game mechanics.

As the points variable was stored using session Storage, the process of adding the point value to the variable was harder than initially thought. In order to implement this functionality, the session stored variable would need to be passed an Integer using the method, "parseInt", followed by the value that needed to be given.

This was a necessary step because it was discovered that when you add an integer value to a session stored string, the value that was added would be converted into a string. This would result in having a score of "010" instead of "10" if the value you wanted to pass was 10.

Due to this, the use of parseInt was necessary to convert the integer into a string value, which could be added to the value correctly.

3.2.3. Point Storage

The point storage refers to the process of storing the score from the game into cookies as designed within section 2.2.4. This process was far more complex than initially thought, as each cookie would need to store a user imputed string and the value from the points variable. This required some further function development.

Name input

The process to allow user input was simple as the JavaScript method “prompt” could be used. This method creates a window prompt box, which allows a user to input any character that they want. Once this method was implemented, the user imputed string is stored into a variable, which is then pushed into the cookie.

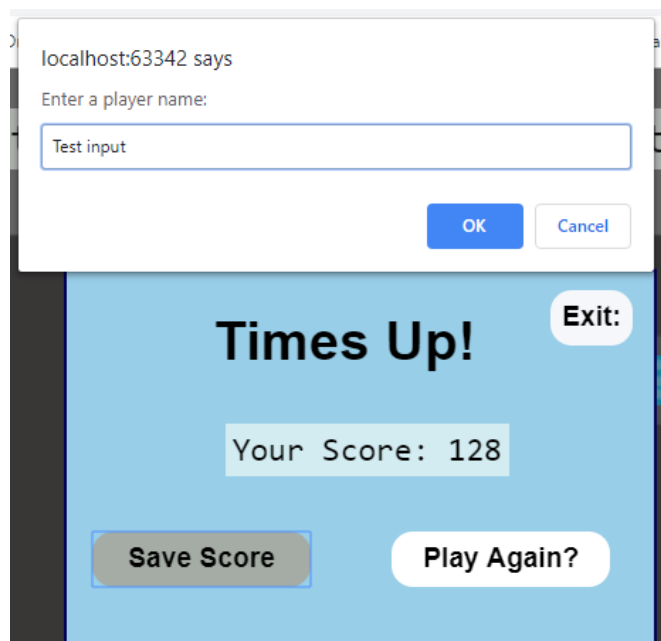


Figure 19: Example of the pop-up box which appears when the “save score” button is pressed.

The above picture shows an example of the user being able to input their own user name to be stored into the cookie.

An additional step was required to allow for safe data storage, this step was to sanitize user inputs. This method refers to the removal of special characters that could corrupt the cookie stored data.

This method was implemented by creating a function which would replace any characters that were deemed to be dangerous, with a blank string. An example of the code that was used is shown below.

```
user = user. replace(/[^\\w\\s]/gi, "");
```

```
user = user. substring (0, 10);
```

This above code shows the methods used to replace any unwanted characters. The characters present within the “user. replace” are not allowed to be used within the system. In addition, the “user” variable is only allowed to be passed 10 characters using the “user. substring” method. This was done to prevent any unwanted code inserts.

Storing the scores

In order to allow the points to be stored, a function that creates a cookie would need to be implemented. This was a simple function which created a cookie using the JavaScript method, “document. Cookie” and then passing in a variable for the name and value.

With a cookie created and a name string being added to the cookie, the next step was to get the game score and then push the value into the cookie. This process concerned passing the session storage for the points into a new variable which could then be pushed into the cookie.

This process was not successful initially due to the value type of the scores not being supported by the cookie storage method.

This occurred due to a cookie only being able to store string type variables. This was a problem for the scoring system, as the points would need to be recorded, but were not supported. Due to this issue, the score variable was converted to a string.

This was done by creating a reference name for the score variable called ‘scores. Once the score value could be found a method called “JSON. Stringify”, was used on the value. This method converts the integer value into a string, which could then be pushed into the created cookie.

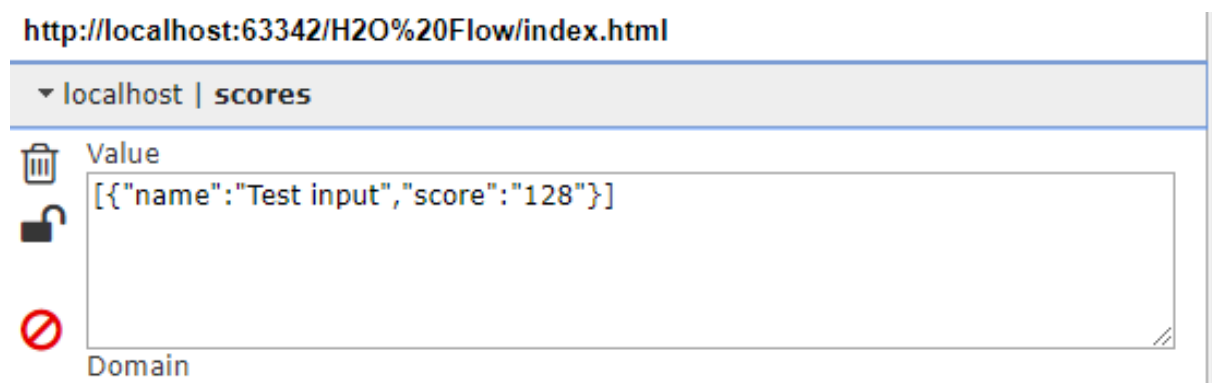


Figure 20: Example of a test cookie that was created and passed the desired game information. This was to test that the score was stored correctly.

Converting the integer value for the points into a string was proven successful for fixing the storage issue that occurred. The above picture shows the results of a successfully created cookie.

3.2.4. Displaying scores

The final step to having a fully operational score system was to have the data stored in the cookie, displayed on the home screen. This process involved using HTML and CSS to display the scores within a table format on the home screen. This was achieved by using a “div tag” to display the string values onto the screen. The addition of CSS elements allowed the data to be presented to the user within a table format.

This process was simple to implement, however, it took longer than initially thought due to the number of CSS elements that were required to get the username string and points string, to be displayed as intended.

Some additional steps were implemented at a later stage into this feature’s development. The main step taken was to sort the score table so that, the highest scores would be positioned at the top of the table. This was decided upon, to enforce the competitive element of the game, by having the user see the highest score from the previous session.

This feature was implemented using the “. Sort” method on the variable that stored the points. This method compares the value that has just been saved against the previous value. If the current value is higher than the previous, the current value is positioned higher within the cookie. This allows the scores to be stored in the cookie in the order of “highest first”. A sample of the function used is shown below.

```
scores.sort (function (a, b) {  
    return b.score - a.score  
});
```

The second step was to add a scroll bar to the score table. This was implemented with the possibility of the scoreboard not fitting on the screen. By having a scroll bar, the scores saved would still be visible to the user, even if the score table does not fully fit the screen. This feature was implemented by using the “overflow” CSS element.

3.3. Graphical User Interface

The implementation process for the game GUI was simple, with a focus on using HTML button tags and div tags to display information to the users. The more complex additions to the GUI implementation was the decision to prevent button interaction under certain conditions.

An example of this would be, the decision to disable the “red arrow” when the Flocculant had a value of 1. This was achieved by creating a function called “disableRemoveFloc”. This function would set the disabled method to true on the HTML button tag when the Flocculant value was equal to 1.

Another method would set the disabled method to false if the Flocculant value was above 1. This was done to allow for regular game functionality if the Flocculant was brought above 1 and needed to be reduced again.

An additional issue that was not necessarily a UI problem but did limit the accessibility of the game on other systems, was an issue with the THREE.js camera.

When the window was resized for smaller screens, the text and buttons would down-size as intended, but the camera would remain in the same position. This led to some of the game objects not being on screen for the user. An example of this problem is shown below:

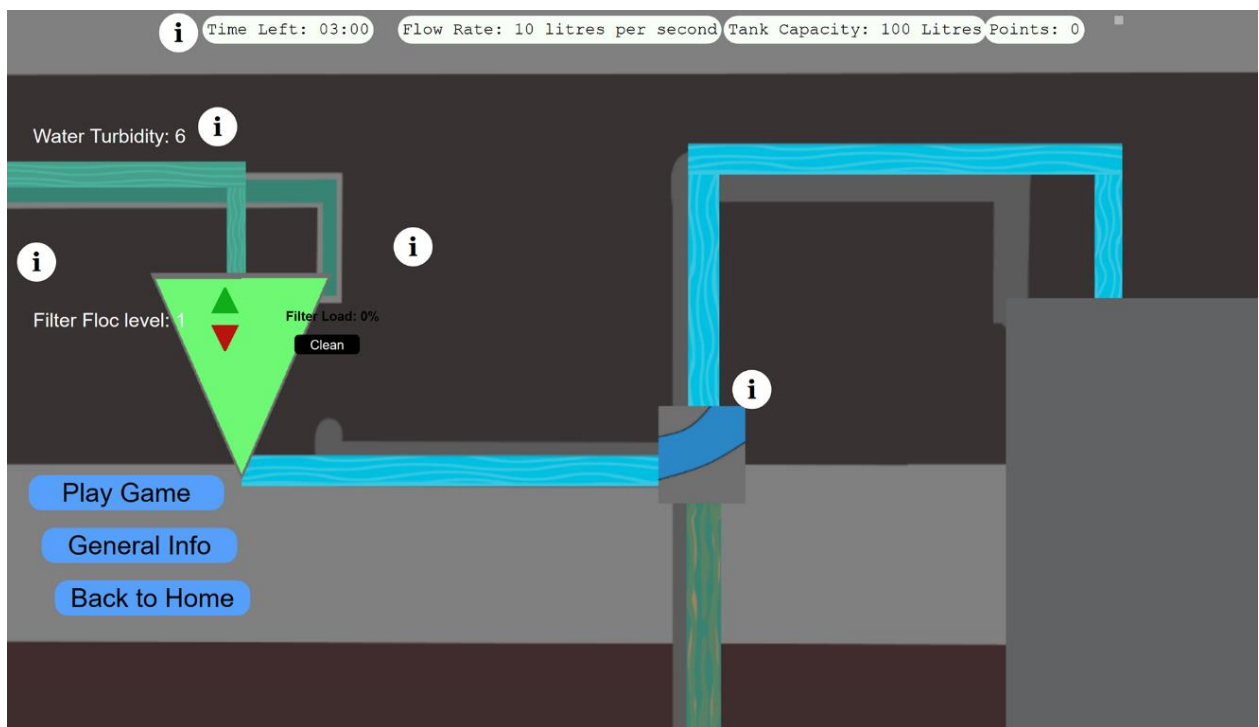


Figure 21: Example of the camera not resizing correctly. This picture was taken on a screen resolution of 1440 X 900.

The above screenshot shows the result of the tutorial page when the game was played at lower resolutions. This problem occurred due to the camera position not updating correctly on a window resize. To help prevent this, a decision was made, to fix the camera to a specific aspect ratio. The camera width was set to 1920 and the height was set to 940. This ratio matched the position of the camera when the window width and height were used.

3.3.1. Animated water textures

During the implementation of the GUI, an added feature that was not decided upon initially was added. This feature concerned, animating the water textures for the game. This was decided upon to show the user where the water was flowing, as an additional visual indicator of the functionality.

This feature was implemented by using the methods, "OffsetX and OffsetY". These methods can be used to offset a texture on an object by a certain value.

These methods were used within the game, to have the water texture offset by 0.03 pixels to its original location. This function is then called within the animate function so that the texture moves by this value each second, this makes it look like the texture is moving on the object.

3.4. Conclusion

Overall, the implementation of the designed features went as intended. The initial designs of the features did not consider some errors with the implementation of the proposed functionality. However, most of the errors that were discovered were fixed during the implementation process.

Some of the features designed took longer to implement than initially intended. This could be due to the features not being designed as thoroughly as they should have been or the technology available with the chosen languages not supporting the designed functionality.

In conclusion, the implementation stage of the project went well and allowed for some unforeseen functionality being added, which improves the overall flow of the game.

4. Testing

4.1. Overall Approach to Testing

The approach taken to testing the system was to test the features of the game separately. This consisted of testing the User Interface, Game Functionality, and the Storage systems separately. The aim of testing the system this way is to test the features the same way they were designed and implemented.

The features were tested manually by playing the game as a normal user would. This was the chosen method, as it allowed normal game mechanics to be played and tested so that the intended functionality could be perfected.

Unit tests were initially considered to be implemented into the system, however, due to time constraints, this could not be achieved. Therefore, the system was tested manually by playing the game and comparing the results to the results specified within the code.

In addition to manual developer tests, external user tests were made. These tests concerned asking individuals outside the project to play and evaluate the game produced. These tests were made throughout the project and were useful in making additional features and changes to the original design.

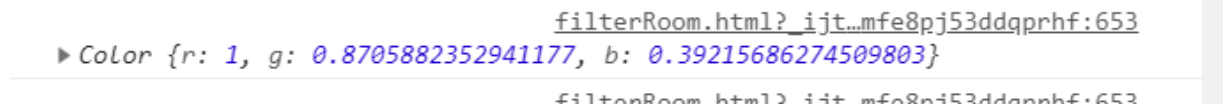
4.2. User Interface testing

Testing for the UI consisted of manually playing the game and viewing all the graphical features. The aim of these tests was to make sure that elements within the game that display information to the user all function correctly.

The features tested were the game timer, additional text labels, and the animated textures. The approach taken to test these features concerned manually playing the game, while checking that the text labels display the correct data specified within the code.

The testing approach followed the white box testing method, as the information displayed to the user on the screen, was compared to the data that was specified to appear within the code. This testing method could not be executed on the animated textures, due to the expected output being specified within their respective functions.

In order to test the filter colour change, it was decided that the material hex colour would be logged into the console log. The code displayed within the console log could be compared to the hex colour chosen within the code. This allowed an easy and efficient way of testing that the correct colour was being displayed to the user.



```
filterRoom.html?ijt...mfe8pj53ddqprhf:653
▶ Color {r: 1, g: 0.8705882352941177, b: 0.39215686274509803}
```

Figure 22: Example of filter hex colour logged in the console being compared to the value in the text string.

A similar approach was taken with the game timer in which, the value for the game timer was printed to the browser console and then compared to the value displayed within the game.

The testing approach followed for the graphical features consisted of viewing the objects and textures to see if they behave as expected. These tests could not yield solid results due to the animations being hard to document. It was ultimately decided that the tests for the animated textures could not be properly undertaken due to the lack of comparable evidence.

The tests executed on the games GUI returned mostly positive results. Using the console log to compare string values was useful to test if the text was displaying the correct information, however, the texture animations could not be properly tested. In order to test the texture animations, some Unit tests would need to be created, but this could not be achieved within the project time scale.

A test case table was created for the User Interface tests in which, the description of the test, the steps required to test the features and the results are present. The test table can be found in Appendix B1.

4.3. Feature Functionality testing

Testing the functionality of the game consisted of playing the game and testing all the interactive elements within the game. The aim of these tests was to test the buttons that were present in the game to see if the expected result was performed.

The approach to testing the feature functionality was to test each individual button present within the game, document the result and then compare the result gained to the expected result. In order to test the function of adding flocculant to the water, the button linked to this function was tested.

This test consisted of clicking the button through normal game play interactions and checking the result. The result was then documented within a table and checked against the expected input. The table that was produced is available in Appendix B2.

The majority of the functionality tests gave positive results from this method of testing. However, the functionality of the features implemented would benefit more from controlled unit testing instead of manual game play tests. This could not be achieved within the projects time span but could be revisited in future builds of the system.

4.4. Storage testing

Testing the storage system implemented consisted of manually testing the storage system, by playing through the game as a normal user. The aim of these tests was to discover if multiple scores could be saved correctly through normal gameplay.

The approach that was taken to testing the storage system concerned saving multiple scores with different string inputs on different web browsers. The aim of these manual tests was to see if the storage system would work on multiple web browsers, as well as making sure the data was stored correctly.

To ensure the storage feature worked for a range of web browsers, the game was played and tested on Firefox, Google Chrome, Microsoft Edge and on mobile Safari. In order to check that the cookies had been created successfully, the score table on the home screen was checked.

If the cookies had been created successfully, the score table would have some values displayed. In conjunction, when the cookies had been deleted, the table data would be erased.

Additional testing concerning, the removal of special character was also undertaken. These tests concerned, imputing a range of sanitized characters into the text input field. This test should result in having no cookie stored or displayed within the score table.

Storage testing resulted in positive results. It was found that the cookie storage system worked correctly on all the browsers the game was tested on. The results show that every score that was stored while playing the game, was displayed on the home screen once the game had ended.

Additionally, the cookies which were stored within the browser session were only deleted after the expiry date had passed and remained visible after the browser session was refreshed. A table of test cases was created for this system and is documented within Appendix B3.

4.5. User tests

This section will focus on the feedback given by external play testers. Three individuals were asked to test the final version of the game. In addition, an earlier version of the project was presented at Science Week (12th March 2019), which was tested by a range of different people.

The testers were asked to try every section of the system including the tutorial page, gameplay features and storing their scores.

The testers were asked on their thoughts about the game and how enjoyable the system was to play and navigate through. This section highlights the experience of the tester with the game, focusing on the parts of the game that work, and any errors that were encountered.

4.5.1. Science Week Feedback

During science week, many people tested the game when it was halfway through development. The feedback gained at science week suggested that a re-design of the user interface would help a younger audience navigate the game.

The issue that most testers had with the UI concerned the information being hidden from the user. This feedback aided the decision to make the game information present at all times to the user.

The most common feedback given, concerns the users not being to understand what the aim of the game was. This feedback was very useful and inspired the creation of the tutorial page. Due to this feedback, the design and implementation of the project focused on the gameplay aspects of the design.

The feedback gained from Science week helped focus the remaining weeks of the project on the areas that were lacking at that stage of development.

4.5.2. Game Tester 1

Thoughts the system

The game tester stated that the Tutorial page UI was easy to follow and use. The information available was said to be educational and useful for working out how the game functioned.

The user noted that the information pop-up for the flocculant appeared unexpectedly when the buttons were being tested.

The gameplay was easy to understand and use after reviewing the tutorial page. It was mentioned that two minutes feels like the ideal time to keep focus onto the game, if the game was any longer the interest would be lost.

Additionally, the game functioned well with no noticeable issues with the UI or logic. However, the rapid spikes in Turbidity that occurred twice during the game, effected the overall flow of the game.

The educational message of not wasting water was easily understood through the information provided within the tutorial page. It was noted that the information was not overly complex and complements the gameplay.

4.5.3. Game Tester 2

Thoughts the system

The information provided within the tutorial page is clear and precise. They commented that the functionality of the game is easy to understand and follow, even from a brief look at the tutorial.

The user thought that the game was good with simple graphics and smooth gameplay, it does not feel too “busy” and every button has its place.

The instructions were mentioned to be present are clear and easy to follow. Overall the game is enjoyable and gives you plenty of time to play.

The play tester stated that the was educational aspects are integrated well into the gameplay, they were not aware of what “Flocculant” meant before playing the game.

The tester said that overall, the game is a fun and enjoyable way to learn about where clean water comes from.

4.5.4. Game Tester 3

Thoughts on the system

The third tester stated that the tutorial page shows good and easy to follow information which is acceptable for any age group. The ability to test the buttons yourself during the tutorial is a good addition.

It was mentioned that the gameplay is clean and easy to follow. The idea of paying to clean the filter creates a good competitive edge to the game. The game emphasizes the concept of keeping the water clean at every stage of the game and how hard that can be.

This tester ended by saying the overall, game is a fun and interesting experience to play. The educational message behind the game is well thought out and lends itself well to the game play.

4.6. Known Bugs

This section will feature a list of bugs that were found during testing.

4.6.1. User Interface bugs

- **Minor** – “water level” of holding tank can experience sizing errors when the browser has been tabbed out.
- **Minor** – Animated water texture moves in the wrong direction on occasion if the browser window has been changed.
- **Major**- Game cannot be run from the folder directly. The game requires a server to run as intended due to a problem with the file paths. Some browsers block the use of images and scripts that are not run on a server.
The game can be played on any web browser by using a version that is stored on a server (9).

4.6.2. Functional bugs

- **Minor** – Information pop-up concerning the Flocculant on the tutorial appears randomly when the arrow buttons are pressed.
- **Minor** – Information pop-up concerning the weather on the tutorial page appears randomly.
- **Minor** – Filter load gains points quicker than intended. If the browser is refreshed, the filter load value may increase at double speed. For example, if 2 points are being added initially, after refresh, 4 points are being added.

5. Critical Evaluation

This section will review the work that has been achieved during the project and if the outcome matches the project criteria. An evaluation of the design, implementation, technologies used, and any further enhancements will be discussed within this section.

5.1. Project aim

The aim of this project was to create an educational game for public engagement. The game would need to be fun and engaging while teaching the target audience an educational message using gameplay.

The aim of this project has been achieved with the creation of a fun and engaging game which conveys the message of “cleaning water is hard” to the game user. The final product has been given good reviews by the game testers and members of the public as discussed within section 4.5.

However, the game does not fit the project aim perfectly. The gameplay that has been developed can become predictable and repetitive quickly if the game is played over and over. As the concept of the gameplay is simple, the complexity of the system has been sacrificed for accessibility.

This is not necessarily an issue as the game was meant to be developed with the aim to be accessible to the general public, but the gameplay is not as interactive as it was initially designed to be.

The use of HTML and JavaScript as the chosen programming languages was an excellent choice to help reach the project aim. The accessibility of HTML lends itself well to this project, as the product can be played anywhere without the need for added technology.

In addition, the use of THREE.js was a huge asset to this product. Object user data provided by THREE.js allowed the implementation of the designed features to be much easier than initially thought.

5.2. Project Management

The use of Feature Driven Development lends itself well to the project and allowed for an easy to follow a development plan. The concept of breaking up the different sections of the system into features was very useful for this project.

As the aim of the project concerned making an educational game, the idea of splitting up the educational features and the gameplay features was ultimately a very good decision for this project. It allowed the more complex parts of the system to be designed and implemented separately to make sure that the feature worked correctly, before being added to the system.

This was very useful when planning what features should be designed first. Having an upfront feature list which documented the functional requirements of the project was very useful when deciding what features were necessary and which could be scrapped.

However, there is an issue with the methodology used. As the methodology focuses on the design and implementing features separately. There is a lack of an overall structure for the program. This was an issue which caused certain features to take far longer to implement than initially thought.

Having the features designed individually does not allow for the added consideration of how the feature would function within the overall system. In some cases, a feature was designed and developed to bring added complexity to the gameplay but was scrapped due to the feature not working within the final system.

This uncertainty was the biggest problem that occurred with FDD and led to unnecessary changes throughout the project. Overall, the use of FDD was ultimately positive for the project with the added flexibility of allowing changes to be made, throughout the project.

5.3. Design

The design of the system was extensive and ultimately went well. Having each feature of the system being developed before the project had properly started, helped to get an idea of the direction that the game was going to take.

The decision to use THREE.js to create a 2D simulation game went well, as the added features of texture animation and object creation helped design an overall theme of the game. The decision to create a water cleaning simulation game set in a water factory was an excellent theme to work with. This theme allowed for creative freedom with what could be done to create a fun and enjoyable game.

The design for the core features within the system were executed well, with all the features adding to the educational message of the game while being fun to interact with.

The system would have benefitted from additional design when considering the filter functionality. As the filter was meant to convey to the user that cleaning the filter is an expensive process, the filter feature was designed within this concept in mind. However, due to time constraints, this features design was rushed and does not correctly convey this message.

Having the process of cleaning the filter cost points to perform is decent. However, this feature does not make too much of an impact on the gameplay and could just be ignored.

In conclusion, the designs for the gameplay features were executed well and helped to convey the educational message behind the game. However, some of the designs for the features were rushed and led to basic and simple functionality which could be ultimately ignored during normal gameplay, without removing any important gameplay aspects.

5.3.1. GUI design

As the game produced conveys its functionality to the user using mainly graphical elements and text labels, the design for the GUI was the most important feature to design correctly.

Overall, I think the decision to have the gameplay buttons available to the user constantly through the game was a good decision. This decision ultimately removed any unnecessary confusion with having the user interface hidden behind menu systems.

The design for the GUI was done well overall, but there were some issues with the design chosen. The game screen could be too cluttered with buttons and information when played for the first time. Having the tutorial page extremely important to the system shows that there is a problem with the UI design.

The aim of the game can be hard to understand initially, due to the UI not being descriptive as to what each button does. The layout of the UI causes this problem, as it can be unclear as to what the text labels and buttons correspond to. The UI worked well if the user has played the game before, but for first-time users, it can be confusing as to how the game functions.

Overall, the GUI is designed well with the inclusion of animated texture to show where the water is flowing, but the UI can be confusing to work out for first-time users and needs to be improved.

5.4. Testing

The approach to testing taken during the project was decent and allowed the main functionality of the game to be tested under normal circumstances. The testing process was not as extensive as it could have been due to time constraints.

The system would have benefited from having controlled unit tests so that any uncaught bugs could be fixed. From the testing done on the game, there were some noticeable bugs present within the game as discussed in section 4.6.

Having given more time within the project, some extensive tests on the code and algorithms would have been developed. The testing performed was good for testing normal game conditions, but the lack of unit tests leave uncertainty to whether the features are fully working as intended.

However, the inclusion of user tests was overly positive for the program. Having “real users” test the system throughout the development process, helped create or approve upon new features of the game to make the system more enjoyable.

The user tests helped to discover some bugs within the system that were not apparent to the developer. In addition, the chance to get different opinions on the game helped with the decision-making process of certain features and how the overall system should function.

If the testing process was to be revised, the inclusion of different types of tests could be extremely beneficial to the project. Creating tests that investigated the algorithm logic and the lower levels of the system will help in perfecting the game in the future.

5.5. Future Development

The work achieved within this project has been extensive and led to the creation of a fun and engaging game that teaches an important topic to the public. However, the system that was created could be expanded upon in the future to deliver an enhanced experience for the users of the system. The potential changes that could be made to the system are listed below.

- Improved User Interface

- The user interface that is currently used within the game does not have an overall structure and can be confusing for new players to understand. This can be improved upon by making a more defined menu system and exploring new concepts of displaying information to the game users.

The feedback that the user is given could also be improved, by having message alerts and sound effects to convey actions performed.

- Addition of level system

- It was initially planned that a level system would be implemented into the game, but due to time constraints, this was not achieved. The system would involve having the user start with one variable to change to allow for clean water, which is how the current game functions. If the water was cleaned correctly, the scene would change to add more variables or another filter.

The initial idea with the level system was to have the user start with just Turbidity values, and then have an added PH value as time progressed. This system could also include a difficulty setting, which could involve erratic variable data or less time to fill a quota for water.

- Improve game tutorial

- The tutorial page that is currently used within the game, focuses on telling the user the game functionality through text. This system can be improved upon in the future by having the tutorial more interactive. This could involve the user having to play a section of the game, with specific conditions. This would help the user understand what the aim of the game is and how everything works in a fun and interactive environment.

- Database storage

- The method of storing user scores could be altered to include database storage. This would require linking a MySQL database to the HTML game file, which allows for the points to be stored within the database. This could be a good change to the game as it adds more competitive elements to the game.

If the users of the game can see the scores of users from different systems, the incentive to beat the highest score managed on the game, not only adds replayability, but also adds more of a competitive edge to the game play.

5.6. Conclusion

In conclusion, the product produced for this project has met the aims set out by the project supervisor. A fun, engaging and educational game has been produced, which has been given positive reviews from game testers throughout the project and as a final product.

This game created is not perfect and lacks some important features which would make the game a more enjoyable experience.

The methodology chosen for the project worked well, due to the added flexibility of developing the different features of the system separately. This did create the issue of having a lack of an overall architecture to the system, however, the decision to use this methodology yielded positive results.

6. Annotated Bibliography

- [1] Dwr Cymru (welsh water), inc. "Dwr Cymru Education" 2019. [Online]. Available: <https://www.dwrcymru.com/en/Education.aspx>. [Accessed 22/03/2019].
- This is the main website for Welsh Water. The page that was accessed was the Education section of the website. It was used to get information on the water cleaning process, so that the actions and features within the game could be as accurate as possible.*
- [2] FantasyGameClub, GooglePlayStore, inc. "Pure Mineral Water Bottle Factory Application" November 22, 2018. [Online]. Available: https://play.google.com/store/apps/details?id=com.fgc.Pure_Mineral_Water_Bottle_Factory. [Accessed 25/03/2019].
- This application was found on the google play store and was used to get an idea of how similar games within the industry are built. The game was played with the intention on gaining inspiration for the features that are present within this application.*
- [3] Sablo Games, GooglePlayStore, inc. "Fresh Water Factory Construction: Drinking Games" 13 February 2018. [Online]. Available: https://play.google.com/store/apps/details?id=com.sablo.mineral.bottled.fresh.water.factory.plant.construction.simulator&hl=en_GB. [Accessed 25/03/2019].
- This application was the second water-based application found on the google play store. The game was played with the intention on gaining inspiration for the features that are present within this application.*
- [4] Unity, inc. "Unity Home Page" 2019. [Online] Available: <https://unity.com/>. [Accessed 21/03/2019].
- The Unity website was accessed to get an idea of what types of software could be produced using the Unity platform. Some research into products that members of the Unity community have made was undertaken when using this website.*
- [5] w3schools, inc. "JavaScript Cookies" 2018. [Online] Available: https://www.w3schools.com/js/js_cookies.asp. [Accessed 02/04/2019].
- The w3schools website on JavaScript Cookies was accessed for research purposes, regarding how to make a successful cookie storage system. The research led to the analysis of code which allows for the creation of browser cookies.*

- [6] Ubisoft GB, inc. "Assassin's Creed Odyssey" 2019. [Online] Available: <https://assassinscreed.ubisoft.com/game/en-gb/home>. [Accessed 26/03/2019].
- The website displays information about the game that researched. The website itself was not used for the research undertaken, but it does offer information regarding the software which was researched.*
- [7] Mr.Doob. Three.js. "Core – Raycaster" 2019. [Online] Available: <https://threejs.org/docs/#api/en/core/Raycaster>. [Accessed 15/03/2019].
- The section concerning raycasting on the THREE.js website was accessed for research purposes. During this research it was learned how the raycasting method worked and how it could be implemented.*
- [8] Mr.Doob. Three.js. "three.js home" 2019. [Online] Available: <https://threejs.org/>. [Accessed 24/03/2019].
- The home page of the THREE.js website offers a range of systems and games that were built from the community. The download link on the left side of the website was used to access the THREE.js library of functions for use within the project.*
- [9] users. aber. "Index-of-mal80-Major-Project 2019. [Online] Available: <http://users.aber.ac.uk/mal80/MajorProject/>. [Accessed 28/04/2019].
- This website hosts the game that has been produced for this project. The game is hosted so that it can be run on web browsers other than FireFox, without manually loading the game from the file.*
- [10] GitLab, inc. "mal80-CS39440-Major-Project" 2019. [Online] Available: <https://gitlab.dcs.aber.ac.uk/mal80/CS39440-Major-Project> [Accessed 12/02/2019].
- GitLab was used for version control throughout the project. The report and technical work were uploaded to this project at the end of every feature's iteration. The project can only be accessed by validated users.*
- [11] Wordpress, inc. "Mal80 Blog - CS39440: Major Project" 2019. [Online] Available: <http://users.aber.ac.uk/mal80/wordpress/>. [Accessed 03/02/2019].
- This blog was used to keep a record of the progress that was made during each week of the project life span. The tasks that were finished and started were documented in an informal writing style.*

7. Appendices

A. Feature List

Filtration of water

Filter Properties

This feature concerns the values that will be assigned to the filters separately. The properties that have been chosen consists of the “flocculation levels, PH levels and the overall water turbidity”.

These properties will be used to calculate an overall value called “water quality” in which, the value will have an influence on whether the water is good enough to produce “points” for the player or needs to be ran to waste.

These properties will be unique per filter and will need to have a calculation algorithm to consider the values from the different filters into one variable that will be easy for the player to indicate if the water is okay or not.

Filter load

This feature will concern the idea of having a filter within the game become dirty and clogged up with waste over the course of the game.

This can be highlighted by having the filter change colour slightly or by having a “clean” percentage that goes down over time.

The idea of this feature is to convey to the player that the filters become dirty and action needs to be taken in order to keep the filters and in conjunction, the water that is filtered clean for consumer usage.

This feature will happen automatically over the course of the game, however different weather effects can change how fast the filters become dirty as well as, elements talked about further in this document. Actions can be taken to help reduce the amount of “waste” in the filters throughout the game.

Cleaning filters

This feature would require the user to click on the filter that they want to clean.

This feature can have several mechanics, such as having the cleaning process cost money and reduce the amount of flow going to the holding tank for a few seconds to help simulate taking filters off the network.

This would work by having the player press a button when they think it is necessary to do so based on a percentage that is always shown to the player.

To get this feature to function, mouse controls will need to be implemented to allow the user to press on the button to allow the function to occur. Some calculations regarding the flow of water would also need to be considered to show a significant impact on the games state.

Adjusting Values

Flocculant values

This feature will be used to influence the water Turbidity and the water flow (below). The idea of this feature is to give the water filters a value, which can then be added to or taken away from when a button is pressed.

The idea for the flocculant, is for it to show that adding this liquid to the filters helps reduce the amount of stuff in the water (turbidity) and help clean the water so it can be used by people.

This value will need to be constantly changed as “new” water is sent into the centre, this will be done by having a red arrow and a green arrow as buttons.

This feature on its own is to simple, so to help emphasise the importance of the flocculant value to the player, the amount of flocculant within the filter can help effect the filter properties as well.

Water Turbidity

This feature will be linked to the lake object that will be created with the game. This feature refers to the real-world properties of water turbidity, which is the stuff within the water like sand.

The water Turbidity will be randomised throughout the game at a steady rate and it is the player job to help keep the water turbidity low so that the water quality remains okay. When the Turbidity is too high, the water quality can be deemed as “too bad” which can mean that the flow of water (discussed below) reduces massively, or the water is no longer allowed to be sent to consumers.

Having the high Turbidity will result in having a low score for the player, as you will not be sending any clean water so you will have no points. The value can be reduced by performing actions referring to the feature above (flocculant).

Water Flow

This feature consists of having a value present throughout the game at all times which shows the player the volume of flow of the water that is being sent to the holding tank.

This feature will be essential to the game play as it will be used to help calculate how many points and water will be sent to the player and holding tank respectively. The idea with this feature is to have a value which has effects on the points earned, while also being affected by other sources such as the filter load.

The flow of water will help show the player that the focus of a water treatment centre is producing clean water to homes at a steady rate. So, by having the flow effect the points earned and being reduced by the cleanliness of the filter.

The feature will aim to show the player that everything needs to be maintained in order to get the best amount of points within the game. Water Flow can be shown by having a value present on the screen for the player.

Water Management

Sending Clean Water

This feature will be the main source of getting points within the game and the centre of the game play. The idea with this feature is to have a button or a system which represents clean water being sent to customers of the water treatment centre. This feature will consist of a button that needs to be clicked when clean water is ready to be sent to the customers.

The player will have control under whether the water is okay to send to people or if it needs to be discarded. Sending clean water will increase the amount of water in the holding tank and give the player some points to help illustrate that sending clean water is a good thing and need to be sent consistently.

Discarding dirty water

This feature will give the user the option to discard water that cannot be filtered to a healthy state. This will give the user the option to reduce the amount of health risks that may occur. This feature will not have any educational merit but will create a more streamlined experience for users of the game, by having the option to discard water that is not ideal for the villages within the game.

However, it can be used to show the user that discarding water costs a lot of money, so making sure that this process is only done under extreme circumstances will be good for the user to be aware of.

This feature can be implemented by having the option to discard the water be in the form of a button as the water is being analysed just before it is sent to the main water system.

Buying Water

The idea for this feature would be to allow the player to buy in water when the holding tank threshold goes below a certain value. This feature will have consequences in the form of “costs money” to perform the action which would reduce the players over all points, however this action would also refill the tanks water supply.

This feature can be implemented by having a button that is only visible/functional when the tank threshold goes below a certain value.

The feature is to signify that buying water in should only be done in extreme circumstances, which will be illustrated by having the button only usable under certain conditions.

Point system

Gaining points

Gaining points is one of the key features of my game and adds a new layer to the game play. This feature can be implemented by having points added to the score board when clean water is sent to the holding tank and used by people.

This can signify that it is good to make as much clean water as possible to get the most points possible. The way that you would gain points is by sending clean water to the tank, however if the points gained varied with the total flow of the water being sent, the point system would have more depth.

The idea with this would be to get the highest flow of water to get the highest points, this will help aid the educational aspect of keeping the filters clean, to keep the flow of water high.

This feature can be implemented by having points added to the score variable when certain actions are performed.

Losing Points

This feature is to help illustrate that cleaning water and maintaining machinery costs money. This feature will work by having certain actions reduce the players points.

One of the ideas for this system to be used for would be cleaning the filter. Cleaning the filter when it becomes too dirty to clean water costs money and disrupts the flow of the water to holding tank. This feature will work in conjunction to the filter cleaning in the form of, cleaning the filter reduces the players points.

This is feature will help make the game more challenging by having points get reduced from actions such as cleaning the filter, adding large amounts of flocculant or buying in water. This feature will also be teaching the user that these processes cost a lot of money and should therefore only be used in tight situations.

B. Test Tables

1. User interface testing table

Test ID	Description	Steps	Expected result	Pass/Fail
TC1	Testing the timer for the game ticks down each second.	1. Start the game from the home menu. 2. Look at the timer.	The time value should decrease by a second from 02:00 until the timer reaches 00:00	Pass
TC2	Testing the tank capacity value goes down over time.	1. Start the game from the home screen 2. Check the tank capacity text to see if the value goes down.	The tank capacity value should go down over time regardless if water is being sent to the tank or to waste. 1 point should be removed each second.	Pass
TC3	The Turbidity value should change every 6 seconds during the game.	1. Start the game from the home menu 2. Wait 6 seconds to test the value change.	The Turbidity value should be present on the screen from start up. This value should have a different randomised number every 6 seconds.	Pass
TC4	Checking if the water quality variable changes based on the Turbidity.	1. Start the game from the home menu 2. Use the arrows to alter the Turbidity value. Green adds Turbidity, red removes.	The water quality label should display a different status based on the Turbidity value. If the Turbidity is below 5, the status should be "good". If the Turbidity is above 5, the status should be "bad".	Pass
TC5	Weather icon changes depending on weather.	1. Start the game. 2. See if the weather icon changes every 15 seconds.	The weather icon in the top left should change between "sunny" and "raining" every 15 seconds.	Pass

TC6	Tank water rises if water is being sent and falls if water is run to waste.	1. Start the game. 2. Change the water quality.	The blue section of the holding tank should go down when the water quality is bad, to show that no water is being sent to the tank. The blue section should increase if the tank capacity is below 100 and water is being sent.	Partial – see section 4.6.
TC7	Analysing the filter to see if the filter colour changes over time with the filter load.	1. Start the game. 2. Add flocculant using the green arrow.	The filter colour should change from green to red gradually as the filter load increases. This colour change should start at 30% load. The colour should change with every additional 20% load, up to 100%.	Pass
TC8	Checking if the water animation is functional.	1. Start the game. 2. Manipulate the water quality using the arrow buttons. 3. Check to see if the pipes have an animated water texture.	The water pipes should have an animated texture depending on which way the water is flowing.	Pass

2. Functionality Tests

Test ID	Description	Steps	Expected result	Pass/fail
TC9	Accessing the game tutorial	<ol style="list-style-type: none"> 1. Load the program. 2. Click on the "tutorial page" button. 	The scene should swap to the "tutorial page" when the button is pressed.	Pass
TC10	Accessing information on the tutorial page.	<ol style="list-style-type: none"> 1. Click on the "tutorial page" button. 2. Click on a "i" button to view information. 3. Click on the "close" button. 	<p>Each "i" button on the tutorial page should display unique information when the button is pressed.</p> <p>These information buttons should be closable using the "close" button.</p>	Pass
TC11	Starting the game.	<ol style="list-style-type: none"> 1. Load the program. 2. Click on the "start game button". 	The scene should swap to the "filter room" and the game should begin when the button is pressed.	Pass
TC12	Adding values to the flocculant variable.	<ol style="list-style-type: none"> 1. Start the game. 2. Click on the green arrow. 	When the "Green Arrow" is pressed, the value for the flocculant level should increase by 1. In addition, the Turbidity value should reduce by 1.	Pass
TC13	Removing values from the flocculant variable.	<ol style="list-style-type: none"> 1. Start the game. 2. Click on the green arrow to add additional flocculant. 3. Click on the red arrow. 	<p>When the "Red Arrow" is pressed, the value for the flocculant level should decrease by 1.</p> <p>This action should only be available when the flocculant value is above 1.</p>	Pass
TC14	Resetting the filter load.	<ol style="list-style-type: none"> 1. Start the game. 2. Click on the "clean" button on the filter. 	When the "clean" button is pressed, the filter load value should reset to 0. In addition, a text pop-up should appear near the score variable.	Pass

TC15	Buying water	<p>1. Start the game.</p> <p>2. Click on the “clean filter” button, until the tank capacity is below 50.</p> <p>3. Click on the “buy water button”.</p>	<p>The buy water button should reset the tank capacity value to 100 and reduce the points value by 50 when pressed.</p> <p>This action should only be allowed when the tank capacity is below or equal to 50.</p>	Pass
TC16	Exiting the game	<p>1. Start the game.</p> <p>2. Click on the “exit” button.</p>	<p>The exit button should stop all the game timers and change the scene back to the home screen.</p>	Pass

3. Score storage tests

Test ID	Description	Steps	Expected result	Pass/fail
TC17	Gaining points during the game.	<ol style="list-style-type: none"> 1. Start the game 2. Use the green arrow to change the water quality to good. 3. Check the flow rate. 	<p>Points should be rewarded to the user for having “good water quality”. The amount of points gained should be linked to the value of the flow rate.</p> <p>For example, if the flow value is 10, 10 points should be gained when the water quality is good.</p>	Pass
TC18	Losing points during the game.	<ol style="list-style-type: none"> 1. Start the game 2. Use the red arrows to get bad quality water. 3. Check the flow rate. 	<p>Points should be lost during the game if the water quality is “bad”. The amount of points lost should be linked to the value of the flow rate.</p> <p>For example, if the flow value is 10, 10 points should be removed if the water quality is bad.</p>	Pass
TC19	Option to save the game is available.	<ol style="list-style-type: none"> 1. Start the game. 2. Play the game until the timer reaches 00:00. 3. Click on the “save score” button. 	<p>The option to save the score should be available from the end game menu.</p> <p>Once the “save score” button is pressed a pop-up should appear, giving the option to input a name and save the player score.</p>	Pass
TC20	Checking the score has been saved	<ol style="list-style-type: none"> 1. Play the game until the timer reaches 00:00. 2. Click on the “save score” button. 3. Enter a player and click “ok”. 4. Press the “exit” button to return to the home screen. 	<p>Once a score has been saved, the user should return to the home screen.</p> <p>On the home screen, the score table should have the name and score that was just saved, displayed.</p>	Pass

TC21	Saving the score for the game is only allowed once per game.	<ol style="list-style-type: none">1. Play the game until the timer reaches 00:00.2. Enter a name and then press "ok".3. Click on the "save score" button again after saving the score.	<p>Once the user name and score has been saved, the "save score" button should not function again.</p> <p>The button can be pressed, by the pop-up box should not open until a new game is started.</p>	Pass
------	--	--	---	------

C. Third-party code and libraries

THREE.js code library – This JavaScript extension was used to allow the use of 2D scenes, physics simulation and user data. All instances of this libraries use the “THREE.” method within the system. The files to allows for the THREE.js functionality was found on the creator’s website. (8)

D. Ethics Submission

Ethics number: 12221.

Below is the ethics form submitted for this project.

AU Status

Undergraduate or PG Taught

Your aber.ac.uk email address

mal80@aber.ac.uk

Full Name

Matthew Lee

Please enter the name of the person responsible for reviewing your assessment.

Wayne Aubrey

Please enter the aber.ac.uk email address of the person responsible for reviewing your assessment

waa2@aber.ac.uk

Supervisor or Institute Director of Research Department

cs

Module code (Only enter if you have been asked to do so)

CS39440

Proposed Study Title

Educational game for public engagement

Proposed Start Date

28th January 2019

Proposed Completion Date

3rd May 2019

Are you conducting a quantitative or qualitative research project?

Qualitative

Does your research require external ethical approval under the Health Research Authority?

No

Does your research involve animals?

No

Are you completing this form for your own research?

Yes

Does your research involve human participants?

Yes

Institute

IMPACS

Please provide a brief summary of your project (150-word max)

My project is an educational game for public engagement. The general public will be able to play and interact with my game project, in which they will learn how clean water is produced and sent to household taps.

I can confirm that the study does not involve vulnerable participants including participants under the age of 18, those with learning/communication or associated difficulties or those that are otherwise unable to provide informed consent?

Yes

I can confirm that the participants will not be asked to take part in the study without their consent or knowledge at the time and participants will be fully informed of the purpose of the research (including what data will be gathered and how it shall be used during and after the study).

Participants will also be given time to consider whether they wish to take part in the study and be given the right to withdraw at any given time.

Yes

I can confirm that there is no risk that the nature of the research topic might lead to disclosures from the participant concerning their own involvement in illegal activities or other activities that represent a risk to themselves or others (e.g. sexual activity, drug use or professional misconduct).

Yes

I can confirm that the study will not induce stress, anxiety, lead to humiliation or cause harm or any other negative consequences beyond the risks encountered in the participant's day-to-day lives.

Yes

Please include any further relevant information for this section here:

Where appropriate, do you have consent for the publication, reproduction or use of any unpublished material?

Yes

Will appropriate measures be put in place for the secure and confidential storage of data?

Yes

Does the research pose more than minimal and predictable risk to the researcher?

No

Will you be travelling, as a foreign national, in to any areas that the UK Foreign and Commonwealth Office advise against travel to?

No

Please include any further relevant information for this section here:

If you are to be working alone with vulnerable people or children, you may need a DBS (CRB) check. Tick to confirm that you will ensure you comply with this requirement should you identify that you require one.

Yes

Declaration: Please tick to confirm that you have completed this form to the best of your knowledge and that you will inform your department should the proposal significantly change.

Yes

Please include any further relevant information for this section here:

E. Version control

For this project there were several platforms were used to keep track of the projects progress.

- A project **GitLab** was used for version control throughout the project, with regular uploads at the end of each iteration. The technical work and the report were regularly uploaded to the git repository. (10)
- A weekly blog was created using **WordPress** to keep track of the projects progress during the early stages of the project. The blog posts gave a brief overview to what has been done during each week of the project. (11)