**A level Computer Science**

**PROJECT**

Advanced Water Tech Demo

Codename: Saturnian

Table of Contents

[Problem Analysis 3](#_Toc130350839)

[What is the problem? 3](#_Toc130350840)

[Why would a computational approach be the most suitable for this? 4](#_Toc130350841)

[Thinking abstractly 4](#_Toc130350842)

[Thinking decompositionally 4](#_Toc130350843)

[Thinking procedurally 5](#_Toc130350844)

[Who are the stakeholders? 5](#_Toc130350845)

[Questionnaire 6](#_Toc130350846)

[Individual questions 6](#_Toc130350847)

[Survey responses and analysis 9](#_Toc130350848)

[Question 1 9](#_Toc130350849)

[Interview 1 10](#_Toc130350850)

[Initial contact 10](#_Toc130350851)

[Questions 11](#_Toc130350852)

[Transcript 13](#_Toc130350853)

[Research 13](#_Toc130350854)

[Comparison of similar solutions 13](#_Toc130350855)

[Solution 1 - Minecraft 13](#_Toc130350856)

[Solution 2 - WebGL Water Demo 16](#_Toc130350857)

[Solution 3 – Biome Simulator 18](#_Toc130350858)

[Program specification 20](#_Toc130350859)

[Features I wish to include 20](#_Toc130350860)

[Features I wish to avoid 22](#_Toc130350861)

[Limitation of chosen solution 22](#_Toc130350862)

[Solution Development & Testing 23](#_Toc130350863)

[Milestone 1 – Main Menu 23](#_Toc130350864)

[Development Log 23](#_Toc130350865)

[Test Plan 24](#_Toc130350866)

[Development Checklist 24](#_Toc130350867)

[Dev Log 24](#_Toc130350868)

[UI Design 24](#_Toc130350869)

[Milestone 2: Making the web authentication API 27](#_Toc130350870)

[Milestone 3: Creating a client user interface to download the project 35](#_Toc130350871)

[Original Proposal 37](#_Toc130350872)

[Code 38](#_Toc130350873)

[Unity Scripts 38](#_Toc130350874)

[PlayerMovement.cs 38](#_Toc130350875)

[PerlinBasicCalc.cs 39](#_Toc130350876)

[PerlinCalc2.cs 40](#_Toc130350877)

[Other scripts 41](#_Toc130350878)

[Server.py 41](#_Toc130350879)

[Client.py 47](#_Toc130350880)

[Client-GUI.py 47](#_Toc130350881)

New game idea: 'Dangerous Depths'

Dangerous Depths combines the natural aspect of survival with an unnatural scenario of randomly generated terrain and a doomsday scenario of catastrophic floods and severe water level changes. Your mission is to survive until rescued. Each time this mission is attempted, your surroundings will be completely unknown, with different temperatures and scenery. Good luck! Oh, and avoid those creatures whilst collecting powerful tokens.

# Problem Analysis

## What is the problem?

As someone who enjoys playing open-world games like Minecraft and action-shooter games like Fortnite, I have always dreamed of make a similar game of my own but with a few tweaks and incorporating several of my favourite things like water, geographical structures and imaginative abilities. I have always been obsessed with how water interacts with the environment, how physical geography is interrelated and how imagination interacts with reality, so creating a game or simulation-type application linking these aspects would Be amazing.

This will also serve a purpose, being interesting to many people who enjoy subjects like Physics and Maths due to the nature of water’s physical properties, as well as relief geographers and researchers who may benefit from studying how water has an impact on a physical area.

Whilst many existing solutions are similar to this idea, including Realflow or Psunami, none neither easily allow editing of the terrain involved nor are particularly beginner-friendly to use. Furthermore, these solutions have a high learning curve, are extremely pricey (Realflow costs $2,500 for a starter license) and are generally not applicable to most of the population – who would want to spend days learning a non-transferrable skill like this? A cross between a realistic simulator and a game would solve this. The game aspect would be encouraging for a much larger audience, having the potential of being enjoyable, interesting, and aesthetically pleasing, while the simulation aspect having elements of realism and physical utility.

## Why would a computational approach be the most suitable for this?

Simulations require many different physics algorithms to calculate relative positions, and when this is combined with the sheer number of molecules that water contains, and how these interact with each other, it becomes essential that the computer, relative to its viewport, all procedurally and automatically calculates this. Furthermore, a large aspect of my project is terrain generation, which requires intensive computing power to compute substantial number of mathematical vectors and relative positions. To achieve this, the 2D/3D Perlin noise generation map technique will be used, as this substantially reduces the required time to code and program how terrain is generated, while also using an efficient and well-documented algorithm.

Without a computer, simulating these physics in a randomly generated environment would be a very laborious task: people such as scientists will have to use a physical water container and make observations based on how this interacts with a specific object, with random generation being nearly impossible to recreate due to the nature of humanity. On the other hand, CPUs can easily, efficiently, and quickly process billions of instructions per second, while allowing the end user to have total control over what occurs in the simulation. Physical limitations such as gravity in the real world may also be altered when a solution is being programmed which can open a large variety of more experiments which simply would not be possible.

### Thinking abstractly

Abstraction is the process of removing excessive details of an algorithm or program, resulting in something that only has its key features, hiding information that is too detailed or irrelevant to the end user. In my project, this will involve:

* Creating a menu which allows the user to easily control what occurs in the simulation, instead of editing exact values in the code.
* Include only basic and necessary sounds which indicate specific things to the user.
* Having a simple yet powerful GUI which the user can interact with powerfully yet conveniently.
* Controlling music and sound effects

### Thinking decompositionally

Using decomposition will be essential. This is because decomposition involves breaking down a complex problem into smaller parts that are easier to understand, while being able to solve these individually.

* Generating infinite 3D terrain with interactions depending on its position could be turned into a simple Perlin noise map, without need to create an independent algorithm
* The physics calculations are likely to be handled directly by the engine, in a separate asynchronous process. Being asynchronous, this ensures that any additional processing time required by the physics engine does not cause detriment to the main game or visual processing engine.
* The menu and how this interacts with other parts may be challenging to create, as turning a 2D interactive scene into a 3D scene with completely new controls may take a while. To do this, I could separate the 2D and 3D aspects into different scenes, then bridge the two with a loading screen.

### Thinking procedurally

This is a method of thinking in sequence, in order and logically, such as like a flowchart.

Diagram

Description automatically generated

This is a simplified flowchart showing how each step is calculated in the Unity project. There are two separate systems involving the simulation and the game, which branch off from the main menu GUI. Although these systems are different, with one involving the control of a dedicated Player object while the simulation involves just a positioned camera, they share some features such as the 3D Terrain generator as well as the Structure and Location resolver, which generates and positions randomly generated structures within the project scene. In Unity, the method void Update() is called on every new frame, which allows for post processing and independent fidelity rendering to occur independently on every frame which may or may not involve new positioning of the viewport as well as what is occurring within the scene individually.

## Who are the stakeholders?

The stakeholders in my project are anyone who enjoys playing games which involve an element of survival, exploration and somewhat realistic graphics combined with an element of fiction.

In this case, my end user will be Ben Ford, as I am unable to ask every single person who enjoys gaming – over 3 billion worldwide! Ben has been complaining about not having a new, enjoyable game to play with pleasing aesthetics, as he is bored with most games and just wants to enjoy himself when not studying for his further education course.

## Questionnaire

### Individual questions

In my questionnaire, I will ask the following questions. This is to ensure that I get a breadth of ideas about people’s favourite game genres, what they find enjoyable, their devices and playtime habits. To appeal to a wider audience, the results that I receive from the questionnaire will be considerable in what features I add and other gameplay types that people may ask for. In addition to these general ideas, I have also added some questions directly to do with the pitch of my game, and people will be asked about their direct opinions on this, as well as anything that they find likeable and dislikeable about the brief description of the game. To ensure that the description is as fair as possible, I have deliberately made the name, description and details obscure so that the responses do not become significantly deterministic and repetitive, thus the responses are expected to be quite varied and therefore increase potential ideas of which to include.

#### Section 1 (1-10)

1. What's your favourite game genre?

Response type: single choice of 17 potential choices

Justification: this gives me an idea as to what proportion of people would be interested in the overall specifically, and whether creating something like my idea is possible in the first place, if it is not picked highly. I expect this project to be for a niche audience, so it is not expected to be in the top position.

1. What attracts you to this genre specifically?

Response type: tick boxes with 10 varied reasons

Justification: this is to give more fine-grain information as to why people choose to play their specific genre. I may then be able to use this information to understand what people are looking for when choosing what to play, whether it be with friends, due to its graphical fidelity, how satisfying it is or its complexity.

1. What is your favourite video game right now?

Response type: short-answer text

Justification: A variety is expected due to the nature of the subject. This will mostly help with specificity later, and games with particularly high responses will be notable as they are likely to have a specific feature which makes them stand out from others and, as such, may be implemented into my project.

1. Add any other information to help respond to the above question.

Response type: (optional) short-answer text

Justification: if people feel that there is information about this genre which I have not included or think that there may be other reasons as to why they specifically enjoy this type of game; this gives the opportunity for them to give their input.

1. How much free time do you have per weekday on average, in hours?

Response type: integer

Justification: With schoolwork, revision and other aspects of life taking part for the majority on weekdays, this makes playing games lower down on the list of peoples’ priorities. By asking how much time the average person can spend, and then taking the mean of these values, I will be able to see the optimal duration of a single game session, and given the fact that people typically choose to play multiple games per day, I can calculate how long a game to be and optimise the experience of players to better correspond to this.

1. How much free time do you have per weekend day on average, in hours?

Response type: integer

Justification: Similarly to the previous question, I expect that this integer will be higher by up to 6 hours since people do not have school and take a break from doing work and homework during the weekend. Combined with other factors such as playing with friends being more likely to occur on the weekend, I will be able to get a value as to how long people are likely to spend on a single game, and within this, how many rounds they are likely to play. The responses to earlier questions, particularly Q3, will aid me in this calculation, as games such as *Fortnite* and *Apex Legends* typically have rounds of around 20 minutes, aside from *Creative* games and modes which may last for hours.

1. Do you have any restrictions as to how much you can play/access to devices?

Response type: short-answer text

Justification: Any restrictions explained by an individual may outline anomalous results in the previous two questions and as such may be ignored. This may also help identify any parental restrictions or time restrictions, which may contribute to my calculations involving the optimal duration.

1. What platforms do you play games on?

Response type: selection of 8 platform types (iOS, Android, PC, Mac etc.)

Justification: This is to allow me to target my game to the platform with the most users. I expect platforms such as Linux to have fewer responses as opposed to a platform such as Windows. Furthermore, I will be optimising my program to run on a specific operating system such as Windows via some edits within the Unity game engine, without me needing to create standalone executables specifically designed for more niche operating systems such as Unity.

1. How much would you be prepared to pay YEARLY for your favourite game, if you had to?

Response type: choice of conditions (0, 0 < 25, 25 < 50, 50 < 100, >=100)

Justification: There are two main reasons for this. The people who likely are to spend more money to continue to play their favourite game are most likely to be the ones who are more insightful, as they may care about the experience and title that they choose to play and what it has which justifies this larger expense. This can then be used to find the specific features that makes people more willing to use that game and what to implement in my own to potentially justify a price in the future, although it will be free for all to test initially.

1. If you wanted to create something, how would you do it?

Response type: multiple choice of 3 (real life, game, dedicated software)

Justification: This question is more orientated towards the simulation side of the project, as opposed to the game side. If there is a high proportion of people using a game/dedicated software, this shows that my project idea may be of some use as it is acts as both a dedicated simulation tool (meeting the criteria for dedicated software) and has a game aspect (meeting the criteria for inside a game).

#### Section 2 (11-14)

After completing Section 1, the page will change to Section 2 with the following information.

This text below is displayed to the user before answering questions: (Note - it has been intentionally made abstract and vague to promote people thinking about multiple possibilities, and then they write down the ideas they have later!)

New game idea: *Dangerous Depths*

Dangerous Depths combines the natural aspect of survival with an unnatural scenario of randomly generated terrain and a doomsday scenario of catastrophic floods and severe water level changes. Your mission is to survive until rescued. Each time this mission is attempted, your surroundings will be completely unknown, with different temperatures and scenery. Good luck! Oh, and avoid those creatures whilst collecting powerful tokens.

1. What do you like about this idea?

Response type: long-answer typed text

Justification: after having read the description and having time to digest this idea, I would like for people to elaborate on their take of the positives in this idea. I’m looking for a variety of similar responses so that I can identify the exact aspects of the idea that people are likely to play. This may then help me during development as I can focus more on the details which I know people will enjoy more than other features.

1. What do you dislike about this idea?

Response type: long-answer typed text

Justification: just like the previous qestion, this will give me valuable insights as to the elements which people are less excited about. I hope to see a repeated amount and a pattern regarding this, and I will be able to adjust what I think about these specific aspects and refine them or remove them completely to make the experience better overall. However, I have deliberately made the idea prompt vague and abstract. I have a feeling that the elements which I have not fully described or elaborated on will be noticed by the respondents and be put in this section – if this is the case, these responses can be ignored.

1. What could be added or changed to make this appeal more to you?

Response type: long-answer typed text

Justification: I hope by positioning this after Question 2, people will take their dislikes and reflect on what could be adjusted in their opinion. This too will provide significant, detailed and precise information regarding optimisations that can be made and help me ‘fine tune’ things which may not be as good as they could be. By making the description abstract, responses to this specific question will be very varied depending on peoples’ perspectives and take on this description. I hope that this will include a variety of new ideas – potentially some that I have not thought of or in any other section – in these responses, so I am particularly looking forward to reading and analysing these responses.

1. Would you like to help eventually test out the game and maybe do an interview?

Response type: yes/no

Justification: I will need this to see who would like to take part in alpha/beta testing which will be able to provide me with detailed logging and/or tracebacks which I will be able to use and to improve my code. Logs may include system specifications and FPS count over time so optimisations can be made on my end. Critical errors and tracebacks are expected in this phase and people who answer ‘Yes’ to this will be able to provide me with these detailed files to help me alleviate these errors.

## Survey responses and analysis

### Question 1

## Interview 1

### Initial contact

For interview purposes, my end user will be Ben Ford, as I am unable to ask every single person who enjoys gaming – over 3 billion worldwide! I will be communicating with him throughout my development by sending beta builds, and other trusted people I know will be sent more stable builds of the Unity project. I understand that my end user has recently bought a Windows PC, so he is a suitable stakeholder as he represents both someone who enjoys playing games, designing how games work and understands the difficulty involved with the complexity of game engines.

Ben will check each stage that the project goes through; I will ask him for advice if any complications arise, so we can decide the best way to implement solutions and decide how to fix issues, while ensuring that as many end requirements will be met as possible.

Text

Description automatically generated*Initial contact messages via Discord.*

As I will be collecting large amounts of information from my end user throughout the duration of the project, I will need to make sure that both of us know the desired output, and ideas and methods as to how this goal can be reached.

Information that I may collect include:

* The user’s system specifications
  + This may include CPU cores and clock speed, RAM amount, secondary storage type and amount,
* Logfiles generated by Unity when the coded solution is in alpha and beta testing.
  + This will aid development especially if there are any bugs within the code, as well as more granular optimisations that may be able to be made thanks to tracebacks available within these files, such as including entities being rendered as well as framerate counts.
* Ideas as to what to include and how to include this inside the project
  + This will be essential
* What they are particularly interested in, e.g. hobbies
  + This will act as an example for my overall target audience, giving me an opportunity to see what people interested in my project my be interested in, generally
* Many information already answered in the questionnaire, as well as reasoning
  + This will give me a chance to develop any answers which may be vague, as well as any other inputs and resolutions proposed from someone else to consider as well as my own opinion.

### Questions

After considering the above generalisations as well as information, which will be derived from the questionnaire’s answers, I have decided to ask the Ben in the interview the below initial questions:

1. What is your current solution to the problem?

Expected answer – Blender/a current solution in use

Reason for asking – so that I can confirm how the current problem is being tackled and see whether my idea may be feasible.

1. What is wrong with your current solution?

Expected answer – limitations of current solution

Reason for asking - so that I can work out what features should be edited, created and removed in my solution.

1. What device will you want the program on?

Expected answer - the name of the device being used

Reason for asking - so I can start to identify what language I will be using to program the solution as well as any device-specific optimisations I will need. This could be, for example, adding touch controls for mobile devices or adding controller support for Xbox or PlayStation consoles.

1. What operating system are you currently using on your desktop/laptop?

Expected response: Windows 7/10/11, MacOS X or (unlikely) Linux

Importance: This ensures that the game I create is not compatible with the OS of my target users, but also that older operating systems may also be compatible with the game. I will need to learn Swift if the user answers iOS or iPadOS as well as Kotlin (Java) if the user answers Android.

1. What are the specifications of your system?

Expected answer - the specifications of the device being used

Importance - so I know what the device being used is capable of, and how much storage a typical user is prepared to allow a project to use on their system

1. What is the resolution of your display?

Expected answer – the monitor’s resolution (1920x1080 possible)

Reason for asking – this will allow me to fine tune resolutions to use in textures, and in graphical user interfaces, as any higher resolutions would be unnecessary and lower resolutions may appear pixelated.

1. After seeing this idea, what is wrong with this solution?

Expected answer – the difficulties and bad parts about the current solution.

Importance – by sharing his difficulties and problems more specifically, I will be able to keep these in mind for when I create my solution. This may also be used as criteria later on to include while coding the solution. Anything that is said will be noted down and may further be asked about the details as to why this is wrong and what could be amended.

1. After seeing this idea, what features would you like to add?

Expected answer – requested features, if any

Importance – this will be critical to ensure that my initial ideas are not omitting any critical features and other

1. Are you familiar with the Unity development engine and/or the C# code language?

Expected answer – familiarity with the engine and language (“No” to both possible)

Importance – if the end user knows how to develop Unity programs or is fluent in the C# programming language, I may be able to host the code on a collaborative IDE such as Repl.it so the user can find, report and even fix bugs without my input. This could then be merged into my original code using as system such as Git.

1. Are you familiar with designing game art? Have you ever designed a graphical user interface (GUI)?

Expected answer – any design and experience (a little experience possible)

Importance – if the end user is familiar with game art, they could provide valuable information as to how to add and improve a GUI within my project. As I have had no experience with any sort of art this may prove to be extremely useful, depending on the feedback.

1. Are you prepared to install alpha testing software on your device?

Expected answer – yes/no depending on willingness

Importance – if the end user wishes to help me develop the game as well as any other scipts quickly, then they should answer yes to this. Depending on how much they trust me or trust their system with handling unoptimised beta/alpha software, I’ll send them early tester files. I have preciously made scripts which execute perfectly on my main PC but not on other systems due to factors such as directory expansion errors to user profile folders, so alpha testing software will reduce the chances of errors in the final release.

### Transcript

## Research

To gather information on features I should as well as optimisations I can make, I have come up with a range of ideas as well as several comparisons of different solutions, including both advantages and disadvantages

* What is your favourite video game and why
* What do you look for in video games? Visuals, audio, storyline, multiplayer, physics
* Favourite game genres?
* Why do you play games?
* How much free time do you have per day?
* How much time do you have per day where you play games?
* What platforms do you own?
* Which of these do you game on?
* If you would like to create something, what would you do? Use game/app (blender)/real life moulds? If so, what?
* “How much are you willing to spend on a game?” (“How much are you willing to spend on a game? : gaming”) (Having seen the gameplay, visuals etc)
  + Only if on offer/free
  + Any price if it is worth it!
  + <£1, <£5, <£10, <£20, >£50,
* Would you be willing to pay for a monthly/annual subscription to play your favourite games?
* How to you get games?
  + Download them from online stores (steam/epic games), rent them, buy physical copies, Pirating & torrents
* Is connection speed & download time an issue when playing games? (e.g. high ping)
* Is computer power an issue when playing games? (e.g. low FPS due to low-end PC)

## Comparison of similar solutions

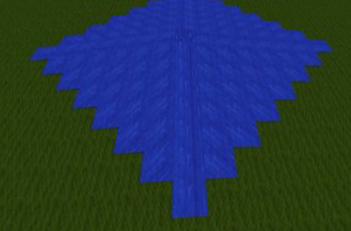
### Solution 1 - Minecraft

Minecraft is a game written in Java for the Java Edition on PCs, or on consoles, mobile devices as well as PCs in the ‘Bedrock’ edition, written in C++. It can be either a sandbox or a survival game, thanks to its variety of game modes.

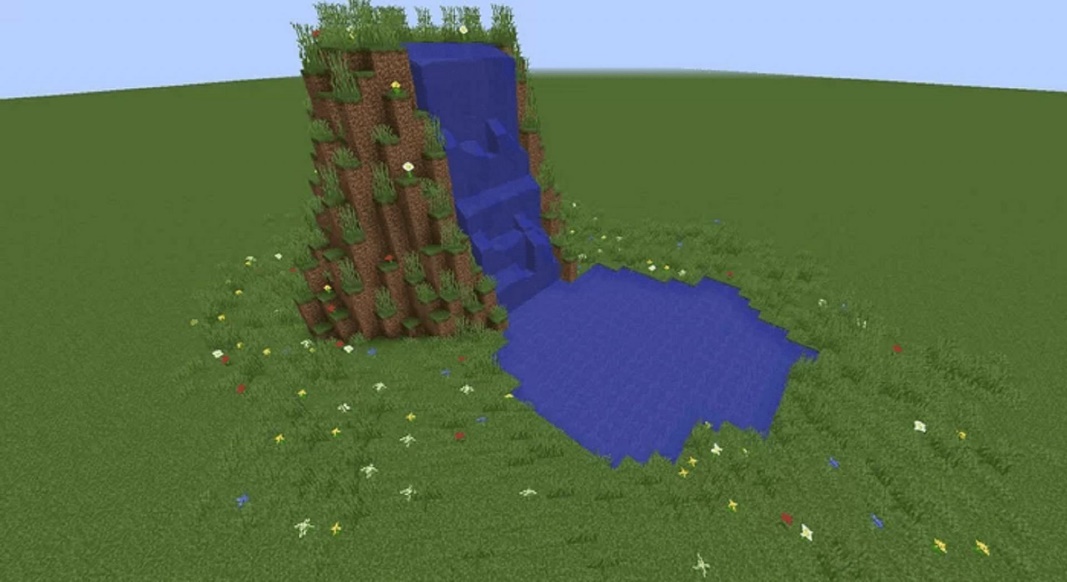
In the “Creative” sandbox mode, there is no element of being attacked by entities which may spawn in the world, compared to the “Survival” mode in which the player has to survive within the world, by gathering resources, building structures and interacting with different items. The Creative mode allows the player to access all blocks which are used to build structures, as well as unlimited items which help the user to interact with the terrain and manipulate it. The Creative mode also allows the user to access commands, which can be used to spawn entities with customised attributes, or to teleport to areas of the terrain using XYZ positioning, similar to how debugging works in traditional high level languages.

The terrain in the game itself is made of 3D procedurally generated, using a custom implementation of Perlin noise which adds another dimension and temperatures of areas within this noise, or ‘biomes’, which dictate the size of and whether different areas of the terrain should be a desert biome (hot), grasslands (medium) or a taiga biome (cold). The game generates ‘chunks’, which are 16-by-16 block areas of terrain, with a 384-block high Y-axis, and then generates structures within these areas such as caves, villages or mineshafts. This chunk data is stored in a custom, proprietary data format called MCA, which store these chunks, as well as position of entities within the chunk, and allow for the player to place or break blocks. If the chunk data is already generated, then the game will read this MCA files’ data instead of generating it again.

However, the game utilises a pixelated and blocky art style, which makes dynamic physics with water and other fluids impossible without the use of client modifications.



The stock game has considerably basic visuals, as the game itself is known for its 16-bit textures and basic graphics. Water is no exception; the game simply calculates the central ‘source’ block, and, if the surrounding terrain is flat, it will extend 7 blocks north, south, east and west, with each successive block being 0.125 block heights lower than its preceding water flow height. If the terrain is not flat, the optimal route to the lowest point of the terrain is calculated from each flowing block, again lowering by 0.125 heights for each successive block. When the water flow, even on the final 0.125m high flow, meets a drop of a block, this new lower location will act like a source block, repeating the flow again. If the drop is over 2 blocks, a ‘waterfall’ is made.

****

Simple water falling from a 17-block tall hill

This can be written in a flowchart format:

However, with client modifications or ‘mods’, water becomes much more realistic. This is achieved by using Ray Tracing (RT) or Path Tracing Global Illumination (PTGI) techniques.



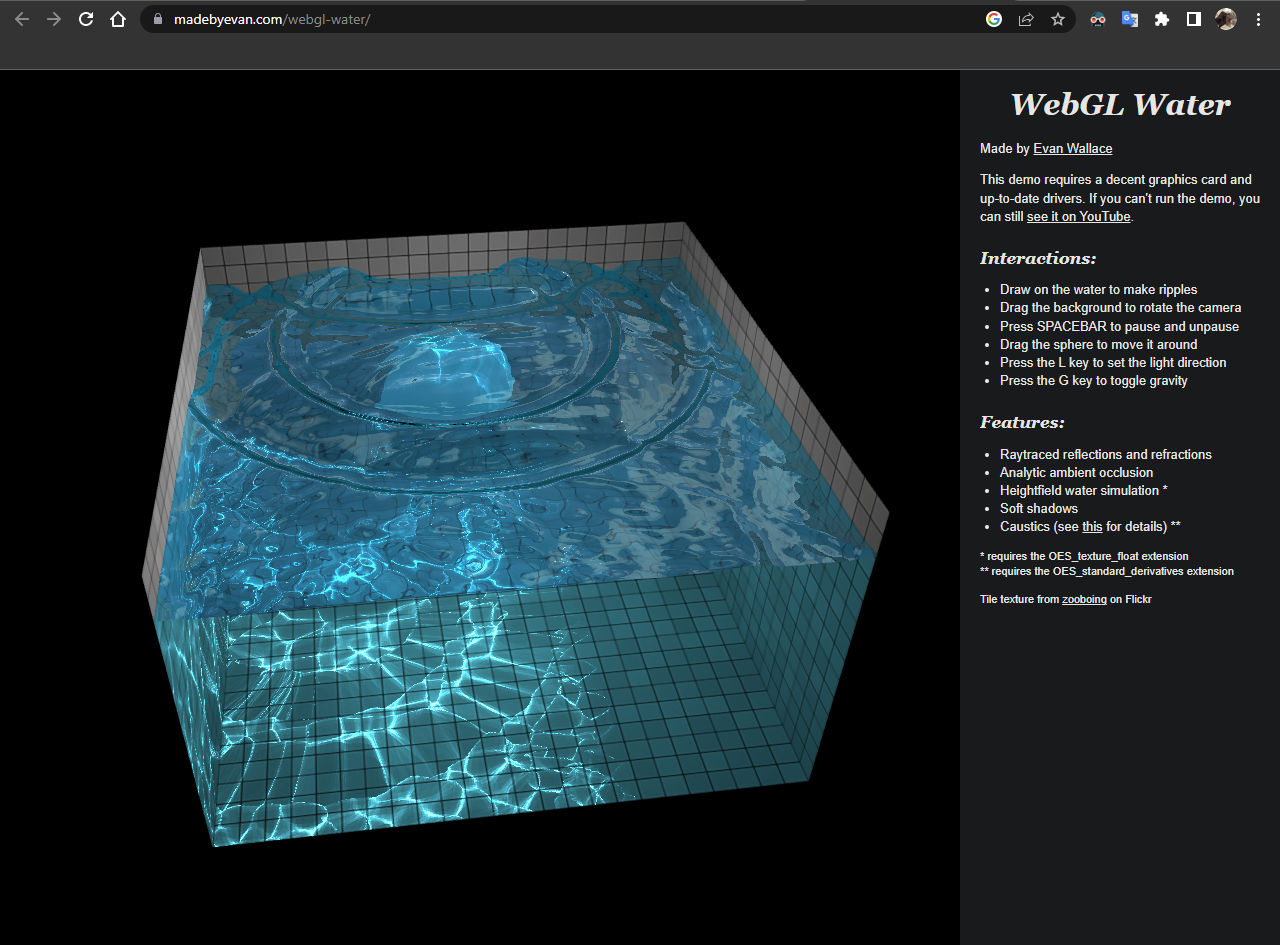
Simulated ray tracing over water using Continuum shaders with the Iris mod

In my project, this could be implemented using the Unity HDRP Raytracing implementation, which could then be togglable behind a specific setting from within the menu, similar to how Minecraft’s ray tracing mode is toggled. <https://docs.unity3d.com/Packages/com.unity.render-pipelines.high-definition@15.0/manual/Ray-Tracing-Getting-Started.html>

|  |  |
| --- | --- |
| **Advantages** | **Disadvantages** |
| Simple and easy to understand fundamentals  Can play and interact freely with any element of the terrain  Many different game modes – creative, survival, spectator, adventure  Connecting to multiplayer servers is easy with access to hundreds of more user-generated content (UGC) as well as game modes.  With a little knowledge, the streamed textures and interactions can be changed and complex mods can be created, from 8K hyper-realistic textures to interacting with APIs to recreate the real-world  No connection needed after the initial download to play in singleplayer worlds | The game costs £30 to download on a PC, and £6.99 on mobile app stores  For my specific purpose, there is a solution of limited effectiveness with reasonable water physics for a game designed around pixelated blocks.  Aspects of the game take a very long time to learn and can potentially be distracting to new players  The UI is very simplistic and has very little customisation and graphics options without downloading client modifications  Customising the code is nearly impossible for the average user, especially on versions which are not the Java PC edition  The game is a large download, and looking for mods and textures online has a risk of downloading malware, or may cost more money to obtain |

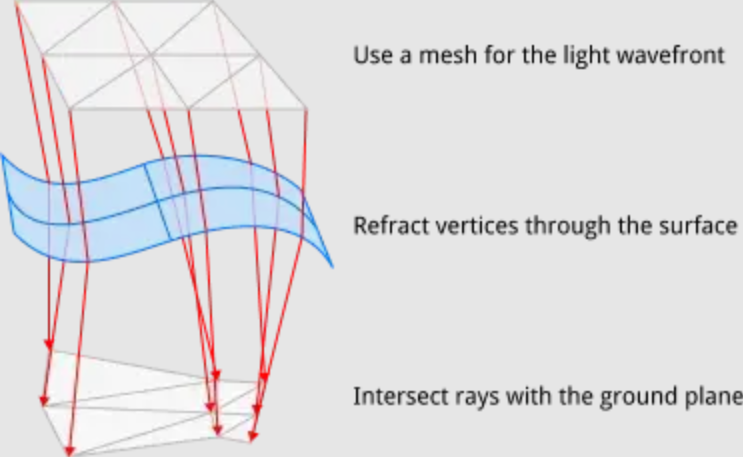
### Solution 2 - WebGL Water Demo

<https://madebyevan.com/webgl-water/>

***Screenshot of the page after having toggled gravity and dropped a ball in the pool.***

This is a dedicated in-browser water demo written in JavaScript, HTML and CSS. It is a pool of water rendered with reflection, refraction, caustics, and ambient occlusion. The pool is simulated with a height field and contains a sphere that can interact with the water's surface, as well as having noticeable impact on the reflections on the tiles, called ‘caustics’. The demo is also easily moveable with the mouse, or by dragging on mobile-based devices.

Furthermore, dragging around on the surface or tapping results in a small ripple, contributing to its realism. The best part about this demo is that when waves and a concave motion, makes the surrounding area dimmer, while light being focused using convex light bending results in the focal spot being much lighter than its surroundings. As these ripples collide, the outcome of the caustics is very realistic as it utilises real-time approximations or ‘path tracing’ to calculate how different light rays diffuse. Light spreading out within these waves, or in.

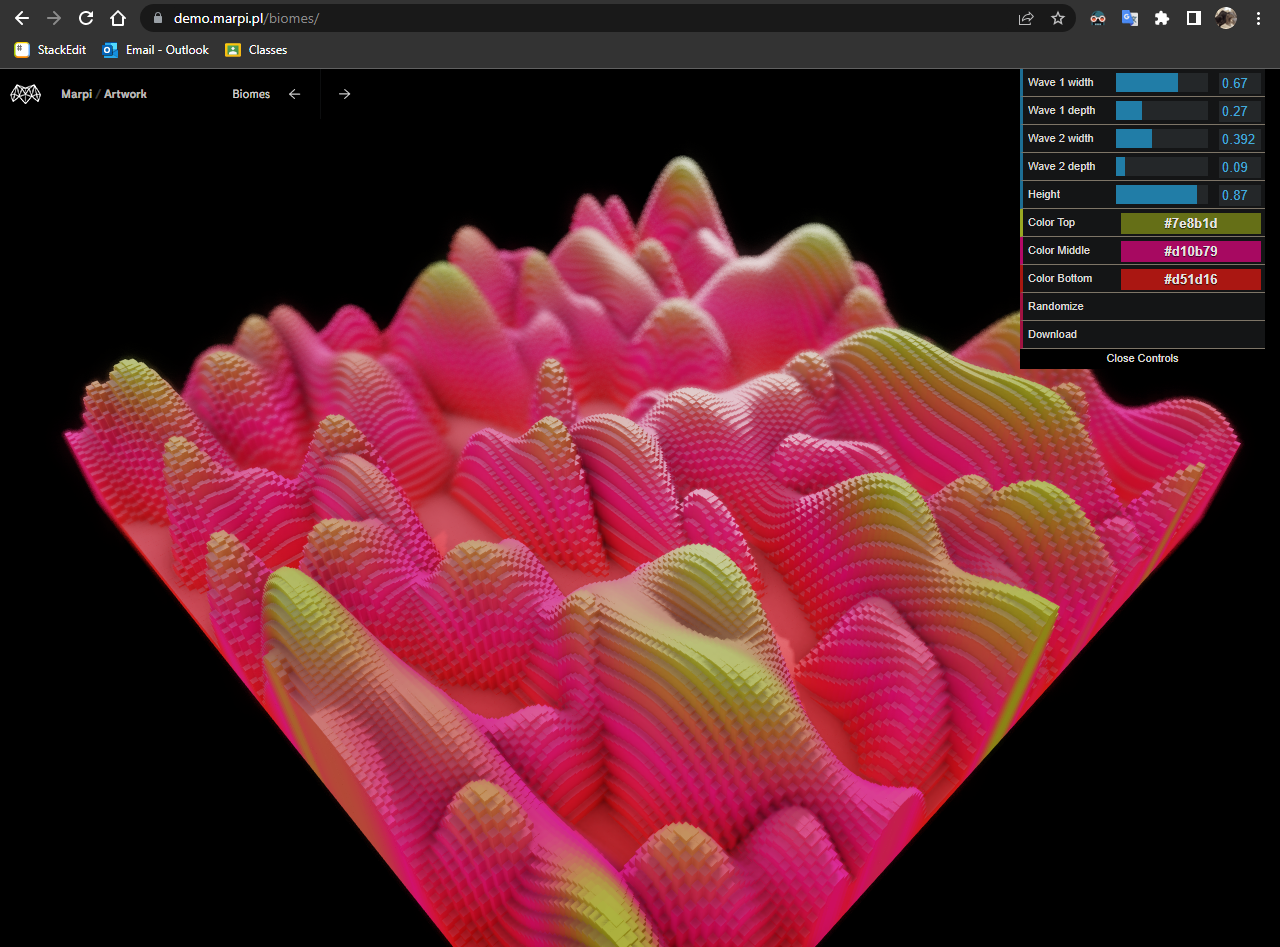


***Diagram used to describe the visualisation by the author of the site.***

|  |  |
| --- | --- |
| **Advantages** | **Disadvantages** |
| Written for in-browser display, so compatible across multiple devices (cross-platform) and unlikely to have changes unless browser engines change as Firefox, Chromium-based browsers and Apple devices which use Swift all support Just-in-Time (JIT) JavaScript code compilation in-browser  Highly backwards compatible as it was made on an older standard (in around 2013) it will be compatible for many systems, no matter the OS  Easy to interact with the interface and produces advanced outputs and real-time calculations, despite this ease  Has a sphere inside the demo, allowing for more complex interactions  No large downloads are required to install or any standalone apps which could add clutter to a user’s desktop | No settings for amount of water, changing physics settings and options like the effects of gravity.  Only a small, cube area, which can only be moved with the mouse and no other inputs are allowed.  There are no adjustable graphical settings, for example changing ray tracing to path tracing  Requires access to the Internet to load the web page  No element of ‘replicability’ and only is interesting for a few seconds, before the user has completed everything possible in the demo  WebGL requires a lot of computing power, which can be expensive.    WebGL is not as powerful as dedicated software for water simulation.    WebGL can be difficult to learn and use for inexperienced developers.    WebGL can be slow and laggy if not optimized properly. |

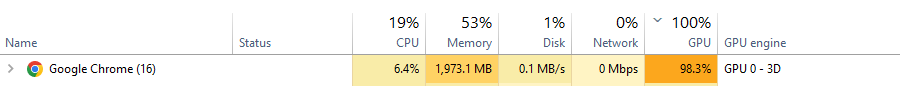
### Solution 3 – Biome Simulator

<https://demo.marpi.pl/biomes/>

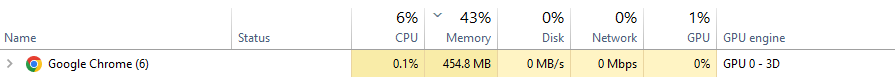
***Screenshot of the page with randomised settings.***

This biome simulator is another web-based demo but has a much larger variety of options available to the end user. Initially, the view is of a 2D plane which can be moved using mouse drags. To change the settings of the biome simulation, in the top right of the screen, there is a choice of useful settings such as which allow for different formations above this plane including the width and height of two different waves, one going vertically and the other going horizontally.

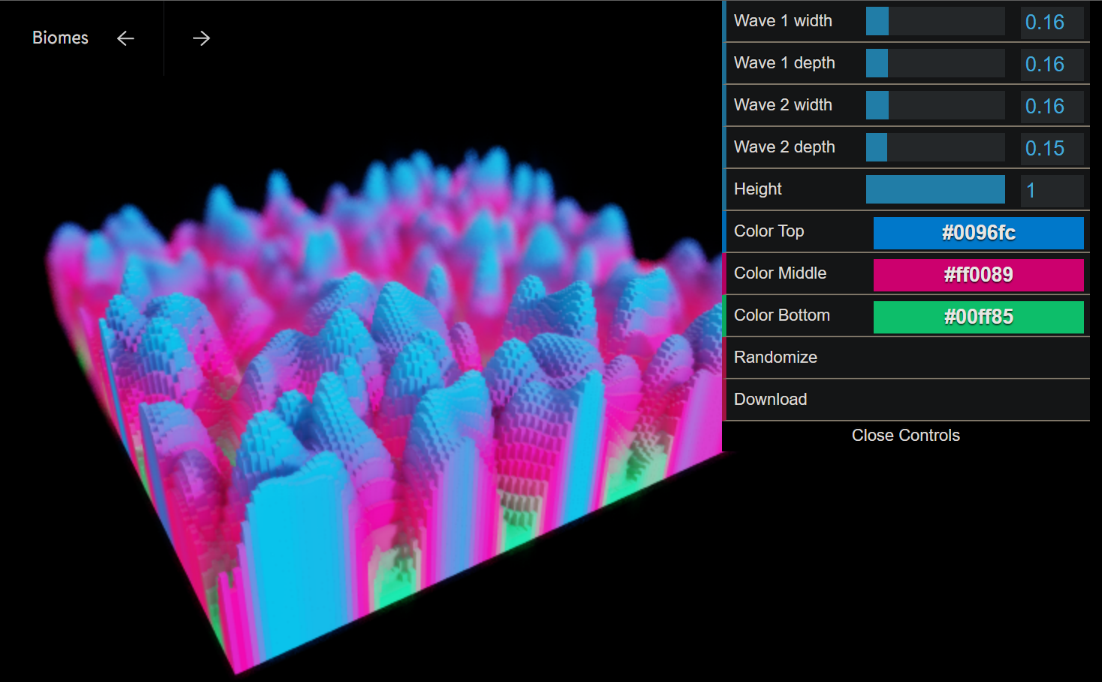
However, the simulator has very little in terms of graphical settings. Although the simulation itself is aesthetically pleasing, this comes at a performance hit. It implements a type of ray tracing implemented in WebGL, resulting in high performance usage to calculate relative positioning and lighting within the scene.

***Screenshot of Task Manager, showing resource usage when the scene is being rendered.***

As Chrome and most modern browsers use hardware acceleration, it renders this WebGL rendering code from the GPU. Whilst this reduces the workload for the CPU, it still shows a noticable input delay and framerate drops, as the GPU is being fully utilised. The system this is running on has an Intel i5-1145G7 @ 2.60GHz CPU, with Intel Iris Xe Graphics as an iGPU.



***When the tab is closed. The computer now likes me!***



***A screenshot of the page with modified settings, setting both wave depths and widths at 0.16 and height set to 1.***

## Program specification

Researching other similar solutions to the problem presented to me allowed me to view how others have gone about it, giving me ideas into what to avoid and what I should perhaps try to implement myself. The information gained from the user's current solution will probably be the biggest influence of my end product out of all the solutions I researched as well as the responses to my survey.

## Features I wish to include

* Easy to use GUI – WebGL Water Demo
  + This ease of use and intuitiveness makes this easy to understand for all ages, yet a powerful feature that is both visually pleasing and satisfying to use.
* Game aspect – Minecraft
  + This enables more advanced features to be added such as power ups and items to use
  + The environment will be populated with realistic physics and Perlin noise generated terrain. The user will be able to move around the environment using the WASD keys and interact with objects in the environment using the mouse. The user will also be able to collect items and use them to progress through the game.
* The ability to adjust settings – Biome Simulator
  + This ensures that the user can adjust settings, or can define specific settings for the game to use
  + The user will be able to adjust the graphics settings, sound settings, and control settings. The user will also be able to adjust the physics settings, such as gravity, friction, and air resistance. The user will also be able to adjust the Perlin noise generated terrain settings, such as height, frequency, and octaves.
* A 3D layout – WebGL demo/Biome Simulator/Minecraft
  + Essential for any realistic physics system. A 2D game will not be appropriate in this situation as an interactive environment with all my features is not possible in 2D.
* Interaction with SDKs
  + The user stated during the interview that they like to use Discord for sharing what they are currently doing, so the Discord Game SDK will allow for the game they are playing to be displayed on the user’s profile, as well as other aspects such as time elapsed, what they are doing within the game and an option for people viewing the profile to download and play the game.
  + There is simple boilerplate code already existing so this can simply be copied, so this will not take much time to implement. Furthermore, companies who offer these SDKs will make money on having a wide range of applications, so if I have any difficulties implementing this, I will always be able to talk to a professional.
* Adjustable graphics settings
  + I would like to add a slider on the OOBE (out-of-box experience) or first-launch where the user can choose their graphics settings based on the system specifications of their device
    - This will range from ‘potato’ to ‘quantum computer’ to not just be humorous but to give a range of relatable settings for people who do not know how their device will perform.
  + Many programs include this to enable the user to change between preferring visual fidelity or framerates depending on their preference
  + As my program may include realistic graphics, then the program should allow for the user to include high resolution textures as well, as well as lower quality to increase framerates
* Automatically keep the program up to date, without the need for a user to input any URLs or download files manually
  + This is useful as it improves user satisfaction and convenience, especially for alpha and beta testers who would otherwise have to go to a website, download a ZIP file, extract it, etc.
  + This will also reduce the risk of file corruption errors and time wasted by the user who may not know how to properly read and follow instructions, provided that my code works and has no issues.
  + I already have made an existing Python executable program with PyInstaller (so it is a working .exe file) which can update itself, so I may be able to use this code to not just download and update itself, but also the project.
* Have a range of accessibility options so anyone can use the program
  + This may be colour blind modes or font resizing or bolding
  + May also include localisation, with other languages. This would be helpful for my French A-level too, as well as enabling many other people to access my game, or students to learn niche vocabulary within the game.
* Include a dedicated simulation mode
  + This will remove the game aspects from the original project and focus more on realism.
  + More customisable aspects in the GUI will need to be implemented here, as people studying Maths and Physics may enjoy experimenting with different potential solutions inside a given scenario.
  + This will also be more suitable for relief geographers who, not only require a better physics engine but also may be able to use this to simulate real-life scenarios and protect against natural disasters like typhoons and tsunamis based on the relief.

Other non-specific features:

Many demos include helpful tutorials and tips to help users get the most out of the experience, supplying step-by-step instructions on how to adjust the water's properties and create stunning visuals. It may also include a variety of pre-set options that allow users to quickly create stunning visuals without having to manually adjust the water's properties. These presets can be easily adjusted and viewed with just a few clicks.

## Features I wish to avoid

* Internet connectivity being essential to play, aside from the initial download
  + This ensures that an unstable connection will not interrupt what is occurring within the program

## Limitation of chosen solution

* By limiting Internet connectivity, people will be unable to join multiplayer
  + Playing with friends or with other people is the biggest reason in my survey as to why people play their favourite game and its specific genre, so by doing this I will not be able to satisfy an ideal feature
  + This would take an exceptionally long time to code and to program anyway, as well as using services such as Amazon Web Services (AWS), Unity Gaming Services or Epic Online Services (EOS) may cost excessive amounts as well as taking a long time to understand and code their respective documentation

Program Specifications

**Laptop** – Intel U series chip, 6th gen or older. No discrete graphics processing unit.

**Potato** – Intel U series chip, 7th generation or older. Embedded GPU

**Toaster**  – Intel i3-7100 or equivalent. GTX 550 equivalent or newer.

**Decent** – Intel i5-7500T or equivalent. GTX 970 equiv. or newer.

**Gaming PC** – Intel i5-7700 or equivalent. GTX 1060 3gb or newer.

**RTX On** – Intel i7-9700 or equivalent. RTX 2060 or newer.

**NASA Supercomputer** – Intel i9-12000K or newer. RTX 3090 or newer.

**Quantum Computer** – Intel i9-13900KF or newer. RTX 4090 or newer.

# Solution Development & Testing

My initial plan is to use Unity to create and render the 3D environment. It provides many tools to allow for building to different operating systems, a convenient scene system and also various optimisations and enhancements in its engine compared to if I was to create a rendering system from scratch. Unity also has a plethora of documentation and online tutorials was well as assistance to help me if (when) I encounter any bugs in my code.

## Milestone 1 – Main Menu

### Development Log

To start with, I believed that having the main menu in the main scene would be beneficial to have as it could then be accessed at any point. However, I realised as I was working on the project that this actually made switching scenes much more inefficient as I would have to place code for the main menu screen in each individual scene, as opening up assets from another scene increases memory usage and wastes processing power. I would need to destroy a GameObject and as a result the coroutine and features applied to it would be stopped. As a result, I had to make the MainMenu into its own separate scene. A side effect of this was that I could now use a loading screen more easily, which is a positive,

By doing this, I also realised that having everything in dedicated scenes is beneficial, so I created new scenes for SaturnianMain (the game scene) and SaturnianCine (Cinematic rendering mode/realistic physics). I also kept assets shared between the two in a separate folder called SaturnianSharedAssets which makes it easier to load and determine whether code and assets should be in both.

The directory now looks like this:

├───Scenes

│ ├───SaturnianCine

│ │ └───# HD assets and render pipeline here

│ └───SaturnainMain

│ │ └───# game assets and audio here

│ └───MainMenu

├───Assets

│ └───SaturnianSharedAssets

| └───Scripts

| └───# a ton of C# scripts

├───Builds

│ └───Alpha

│ | └───# alpha builds

│ └───Beta

│ | └───# beta builds

│ └───Public

│ | └───# public/more stable builds

└───SaturnianClient.unity

### Test Plan

## Development Checklist

* Basic FPS movement.
* Menu screen with clickable buttons
* Exit button
* Settings area with placeholder values
* 3D terrain system
* Water physics
* Realistic water physics
* Added grass, stone textures
* Texture changing depending on the Y level
* Button to start water flow
* Item pickup on ground
* GUI
* Health bar and damage bar
* Effects system such as speed
* Enemy characters

## Dev Log

24/04/2023: After testing throughout 4 different versions of Unity and creating many sample scenes and following tutorials, I have created my project which I will work on from now on.

Finished, implemented and heavily modified this video: https://www.youtube.com/watch?v=f473C43s8nE

Tutorial: <https://www.youtube.com/watch?v=xCxSjgYTw9c>, up to 30 seconds in.

Created a Git repository with the project’s data. I’m not sure whether I will do follow the semantic commit guidelines, but it might be useful. I aim to write something down in each commit message which will help me to write it up if I just have a day of straight development. Using this command I was able to link my auto saving OneDrive to the GitHub repo so I can keep a better track of what I am editing in the repo and at the same time on OneDrive which I can access in school. mklink "D:\Unity\Csproject\CompSci Project.docx" "D:\OneDrive - Notre Dame High School\CompSci Project.docx"

Solution design

## UI Design

Creating Terrain

Terrain generation in Unity can be complicated. Whilst using Perlin noise like in Minecraft seemed to be a good idea, implementing this would have taken far too long. Furthermore, trying to draw a Perlin scene in high resolution in 3D procedurally is a nightmare – many developers have posted about their experiences with hours long loading times and various issues. On top of this, coding the solution here would have been literally painful – I don’t particularly like maths, but Perlin noise involves a significant amount of complex mathematical operations. The algorithm is based on a combination of trigonometric functions, gradients, and interpolations, which can be difficult to grasp and implement correctly. Moreover, tweaking the parameters of the Perlin noise function to achieve the desired result would have required a lot of trial and error, which can be time-consuming and frustrating.

Despite its challenges, Perlin noise remains a popular technique for generating natural-looking textures and landscapes in video games and other digital applications. Its ability to create realistic and organic patterns has made it a valuable tool for developers looking to add depth and complexity to their virtual worlds.

As a result, I decided fairly early on not to use custom coded Perlin noise and a terrain system that would require high amounts of high-level code in a language that I am not too familiar with. Instead, the Unity Terrain Toolbox, created by Unity, offers a high-level suite of terrain changes. At 1.3 GB for the stock asset, it will add considerable file size to the end product, but it is definitely worth it as it will save significant time, probably several dozen hours.

Graphical user interface, application

Description automatically generated

This is perfect for my project and for helping me with my Geography A level course also.

Graphical user interface, text, application

Description automatically generated

Thermal redistribution is present in permafrost landscapes in real life, so this will help me with that too.

Noise generation is also possible, with Perlin included.

A picture containing text, indoor, screenshot

Description automatically generated

Creating FPS Movement

## Milestone 2: Making the web authentication API

Whilst thinking about ideas for my project, I thought about what I would like to add to make it as user-friendly as possible. I quickly identified a ‘pain point’ which is present in my previous programs I have created, which was the initial download and update process. Sending executables to people, then when I make an update, I used to send a new version of the same executable via Discord or a link to my webserver. This was incredibly tedious, as some users would not want to download and run this and accept the security warnings (as I am not a signed developer with a CA certificate; this would cost lots of money) and the various prompts to download it.

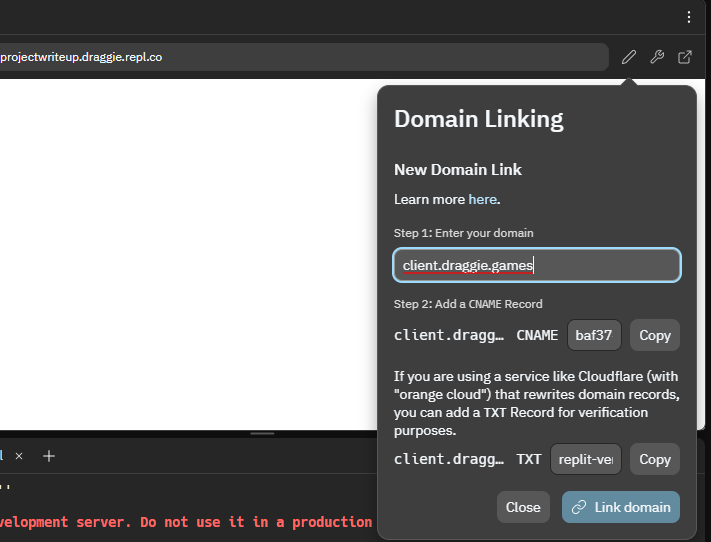
To fix (or at least hope to fix) this issue, I will create a Python script which runs constantly in the background and is executed at startup to ensure this. This script should then interact with some API endpoints on a webserver and allow the user to check their entitlements and, if there is sufficient information related to the user’s account, then it will automatically download the update.

Firstly, I will need to decide what webserver I will use as well as the programming language. I have experience with Replit, it is free, and I know how to make a functioning website. I also own a domain, draggie.games, hosted on the Cloudflare network, which I will be able to assign my Repl (the server which runs the code) to make it look professional and be extremely fast through Cloudflare’s edge server CDN, which I have spent time perfecting how to use. However, as it is a new domain, I will need to reconfigure a few settings. However, I have never created a functioning API with interactible endpoints, so I will need to learn how to do this.

Firstly, to link my domain to my Repl, I will need to use DNS routing and domain linking. As long as there is a listener running on a port, Replit will allow the domain linking function.

We will need to add a CNAME record to link to my existing domain. I did this by opening up the CLoudflare dashboard, navigating to ‘DNS’ and adding the following record:

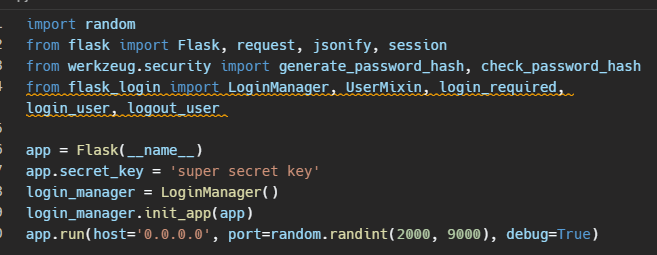
client.draggie.games CNAME baf37e21-28b1-4ccf-865c-b302967b4275.id.repl.co



This CNAME, or canonical domain, record is used instead of an A record which points to an IP address, and is when a subdomain or less commonly a domain is an ‘alias’ of another domain. All CNAME records must point to a domain. Here, client.draggie.games points to a subdomain on Replit’s network, which then holds the A record which allows the actual data of the server to be sent across the Internet.

I will be using the Flask module to make and receive web requests to specified endpoints on the server.

Note: code in this section is work-in-progress and comments have been omitted to condense space. The final code with all data is at the end of this document



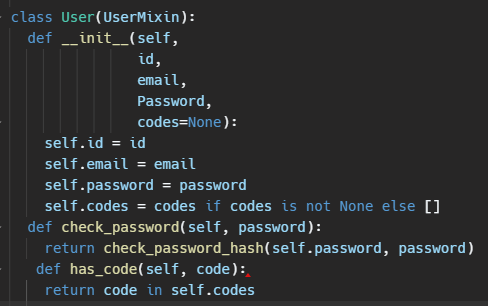
As I have created a Flask app before for a fairly basic use case, I knew that I should run the app on a random port for Replit to realise that it should listen for web traffic. I also set it to run on 0.0.0.0, which makes the script bind to all available IP addresses, so that I will not have to waste time finding out the server’s IP address, and update this if it changes.

Debugging just makes the code reload much faster and whenever changes are made to the source Python file, speeding up efficiency.

The security module and login manager were new to me, but I read and digested the documentation fairly easily. From prior classwork, as I will be allowing people to create passwords, storing these in a plaintext database is not really a possibility. To help with this, used using Werkzeug’s security module to generate encrypted, secure SHA-encrypted hashes of passwords to store in a database which I will make later.

The login manager made authentication on the server much easier as it allows for various checks to be made quickly and efficiently. For example, it can read from my database and parse an incoming POST request to authenticate a user for one particular session.

After the initial import tests, it was time to create the user login class which has the attributes we need, such as username, email and password.



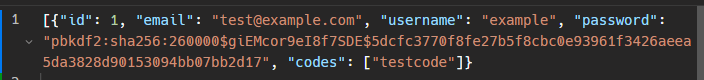
Here, we define the User class with a UserMixin parameter which is used by Flask to securely authenticate users. This provides various properties which returns with associated values, such as is\_authenticated(), is\_active(), is\_anonymous() and get\_id() which are fairly self-explanatory. I will primarily be using the get\_id() method which resturns the unique ID for the object associated with the User instance.

I have also added the ‘codes’ object as a list which is associated with the User instance, which will allow the user to redeem codes to their account. However, firstly, I need to have a database of all users. After a few hours of searching through documentation, I found that the best way to achieve this result was the following:



Here, the load\_users function returns a list of all users read from the JSON file, users.json. I needed to add the try, except condition here in case the file somehow gets corrupted during an operation, if it does not exist (which I used frequently during testing to quickly reset it) or if there are no users actually in the file. In any of these scenarios, the first user is added to the list with a test email and password, generated from the generate\_password\_hash function imported from the werkzeug.security module.

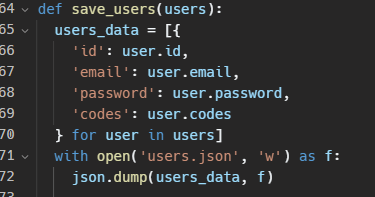
Inside of users.json, the following is created on running the above code:



The id starts at one and increments for every user in the database added. As you can see, the password key has a sha256 value of which its plaintext value will not be leaked if the JSON file is publicised. However the email and username is not encrypted, which may be an issue later that I can fix.

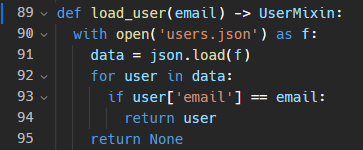
I decided to make the database in JSON format as it is not only easy for humans and machines to interpret, but also the most common ‘language’ (well, notation) for transferring data across the Web and the Internet as a whole. It also looks good when compressed as well as prettified!

In order to save users’ accounts in the future, I will also create a save\_users() function which will allow users to be added to the database of all users, with their own unique properties



The load\_user function used to load a user based on their email address. It iterates through the JSON object loaded through the ‘users.json’ file and then if the email associated with the user matches the inputted, the user object isreturned which holds all the other values associated with the user such as the id, password, and all codes redeemed.

Encrypting the emails here would be computationally intensive but good for privacy. It will probaly be something which I implement later down in the development pipeline.



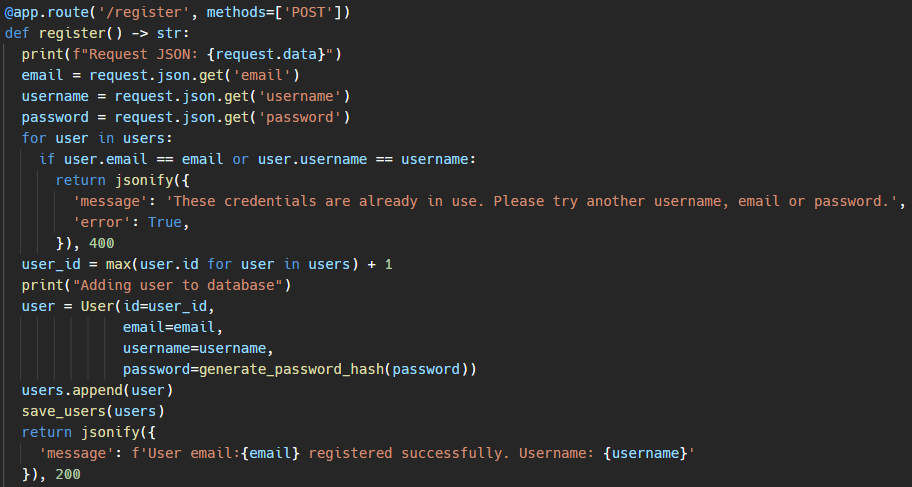
This all worked flawlessly first time which was extremely surprising!

Adding a second user to the database again works perfectly:



Finally, after implementing the basic user load and save functions I could work on the actual endpoints which can be contacted by clients. My intital goal is to have 4 endpoints that are as RESTful as possible. Flask itself is a microweb framework that does not have a particular architecture like REST. As I cannot directly implement REST, I’ll implement RESTful features instead, so there are clear HTTP methods to indicate the type of action being performed and returning JSON data as a resposnse to the user, for reasons outlined earlier.

The register endpoint I created after around an hour:

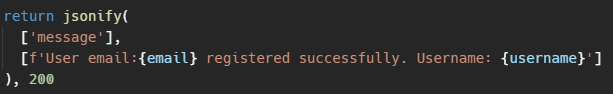


This was my first time creating a POST endpoint and my first time creating an interactive database which allowed for things other than predefined values.

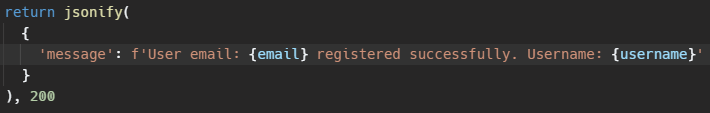
The decorator here specifies the route ‘/register’ to listen to, with all incoming POST traffic being handled by this function. If the traffic is not a POST request, such as something like a GET request, Flask will return a 401 Method Not Allowed response and not run the function below.

For testing purposes immediately after, the data of the request is printed to the console. It then parses all the relevant values from the data payload, including the username, password and email address for later use. After this, a for loop is entered, looping through all users in the users list until a matching email or username is found. If this is found, then a 400 Bad Request JSON object is returned as the register endpoint should disallow duplicate account creation. If there is no duplicate emails or usernames, then the user\_id = max(user.id for user in users) + 1 statement will get the largest value of all user IDs and assign this to the variable user\_id.

This variable is then used used to add an account to the database JSON through the User class. When I initially wrote the code, I missed out a few things like the curly brackets for the JSON response. I was returning a JSON message as a list instead of a dict which took me a few minutes to figure out what was going on! This is what I was doing:

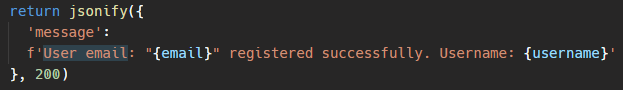


Whereas the correct implementation is:



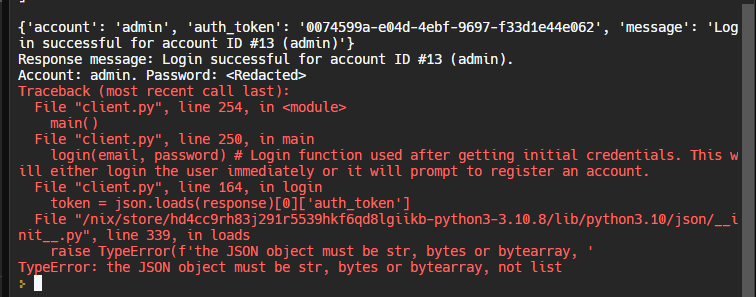
I can add various more JSON keys and their respective values to this, for example adding the username as a key and a value outside of the ‘message’ key to prevent sensitive data being displayed on the user’s machine.

I probably won’t do this for the initial registering, however for generating access tokens in the ‘login’ route may need this, to avoid the risks associated with showing plaintext passwords.

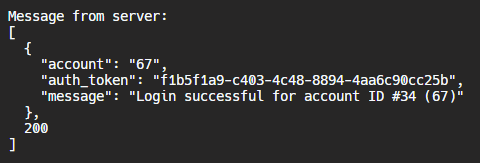


However, this is where I encountered an error. After adding the correct attributes to the returned JSON object, the login client code began returning an error. After having added the status code in particular, the JSON object interpreted by the client was now a list, with index 0 having the required JSON payload, and index 1 containing the HTTP status code.

This was the error:



I needed to extract the Response object directly received from the server in my client code, so I printed out the following as soon as it has received a message:



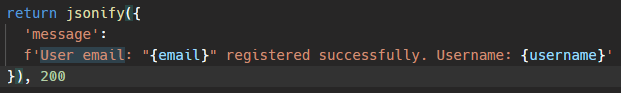
The Response object returned was now wrapped in a list with two attributes, including the payload as well as the status code, which was new. As I had taken a break from coding the project due to Easter holidays and a 1-week French school exchange, I figured that a dependency must have been updated to add this or something similar. To fix this, I had to modify all instances of getting an attribute from any Response object by adding a list index of 0 to the expression. For example,



would now be:



After writing a regex expression to change and add the required text to the code, some other features broke unintentionally. Specifically, all the methods in the ‘login and ‘add\_code’ routes were no longer working and throwing an error. Going back through the code I had just modified, I realised that there was a typo. The closing bracket was placed after the HTTP status code, instead of before. This meant that the HTTP status code integer was being sent as part of the payload, and because it was outside of the JSON object, it was being added as a list. It wasn’t any dependency being automatically updated by Replit, it was indeed my lack of practice for 24 days!



I quickly updated my client code back to what it should be and reverted the mass regex change earlier, as well as updated the server-side code to fix the bracket issue. It was fixed!

Speaking of tokens, implementing access tokens would be ideal.

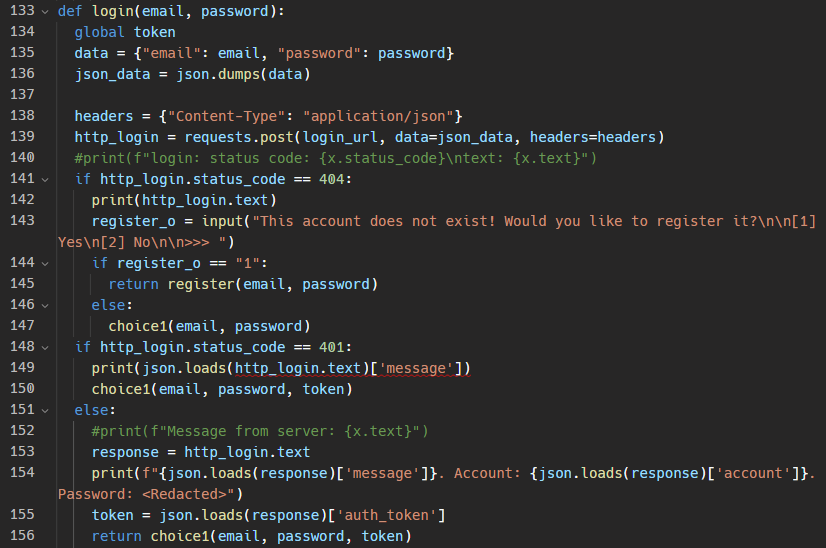
|  |  |  |  |
| --- | --- | --- | --- |
| Identifier | Data Type | Scope | Purpose |
| User | class | global | Represents a user object with attributes such as id, email, username, password, codes, token, token\_expiration and methods such as check\_password, add\_code, has\_code |
| id | int | instance | Unique identifier of a User instance |
| email | str | instance | Email address of a User instance |
| username | str | instance | Username of a User instance |
| password | str | instance | Password of a User instance |
| codes | list | instance | List of redemption codes associated with a User instance |
| token | str | instance | Authentication token associated with a User instance |
| token\_expiration | int | instance | Expiration time of the authentication token associated with a User instance |
| load\_users | function | global | Loads user data from a JSON file and returns a list of User objects |
| save\_users | function | global | Saves User objects to a JSON file |
| load\_user | function | global | Loads a User object with matching email from a JSON file and returns it as a UserMixin |
| add\_user\_token | function | global | Adds a new authentication token to a User instance if it doesn't already have one, and returns the token |
| clear\_user\_token | function | global | Clears the authentication token associated with a User instance |
| get\_user\_token | function | global | Returns the authentication token associated with a User instance if it is not expired, otherwise generates a new token and returns it. |

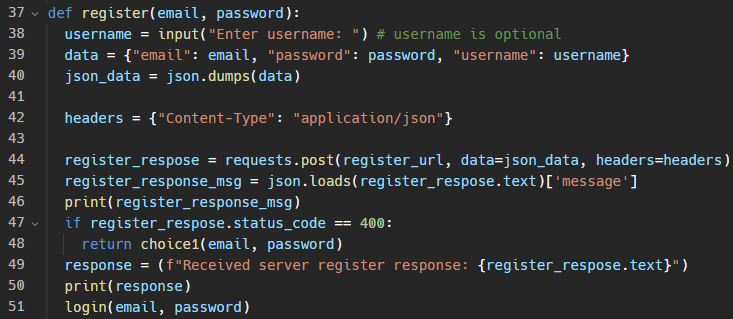
|  |  |  |  |
| --- | --- | --- | --- |
| Identifier | Data Type | Scope | Purpose |
| alpha\_key | function | global | This function retrieves an alpha key for a user, after checking the user's token from the request header, the function checks the token expiration time, validates the codes assigned to the user account and then returns an alpha key. |
| register | function | global | This function registers a new user by generating a unique user\_id, checking if the email or username is already in use and then adding the user information to the users list, which is then saved to a database. |
| login | function | global | This function authenticates a user by checking if the email and password match with the credentials stored in the users list. If the email and password match, a token is generated and saved with the user's information, which is then returned as a response. |
| logout | function | global | This function logs out a user by checking the user's email and password, clearing the user's token from the database, and then saving the updated user information to the users list. |
| add\_code | function | global | This function adds a code to a user's account by checking if the user's email and password match the credentials stored in the users list. If the credentials are valid, the code is added to the user's account and the updated user information is saved to the users list. |
| access\_data | function | protected route | This function is a concept of protected data retrieval that requires authentication with a token. It receives a code from the request and verifies if the user has the code, then reads data from a file and returns it as a response. This function is currently unused. |
| token | variable | local (function) | This variable stores the token retrieved from the request headers. |
| users | list of dictionaries | global | This variable is a list of dictionaries, where each dictionary contains the user's information such as id, email, username, password, token, codes and token expiration. The user information is used for user registration, authentication, and account management. |
| user | dictionary | local (function) | This variable is a dictionary that stores the information of the current user being iterated over in a loop. |
| email | variable | local (function) | This variable stores the email of the user retrieved from the request body. |
| username | variable | local (function) | This variable stores the username of the user retrieved from the request body. |

## Milestone 3: Creating a client user interface to download the project

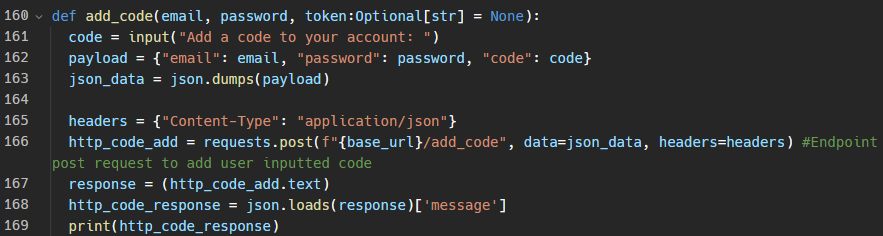
The DraggieGamesClient is a software interface built in the Python and Cython programming languages that allows users to authenticate with Draggie Games, a gaming ‘platform’ and service that I have created. The interface provides several endpoints for interacting with the platform's authentication system.

The main server base URL for Draggie Games is https://client.draggie.games. The first endpoint, /login, allows users to log in by providing their email and password as parameters. Upon successful authentication, the endpoint returns an authentication token. If authentication fails, an error message is returned.





The third endpoint, /add\_code, allows users to redeem a code by providing their email, password, and the code as parameters. The endpoint returns a message indicating whether the redemption was successful or not.

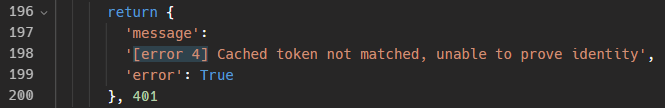


The fourth endpoint, /alpha, allows users to obtain an alpha key contained as an environment variable by providing their email and password as parameters. The endpoint returns a JSON dictionary containing an access\_key string and a decryption\_key and sha\_hash array. The access\_key can be parsed from the JSON and used to download a file from a URL. The downloaded file is then decrypted using the decryption\_key, and each file is checked against the sha\_hash object.

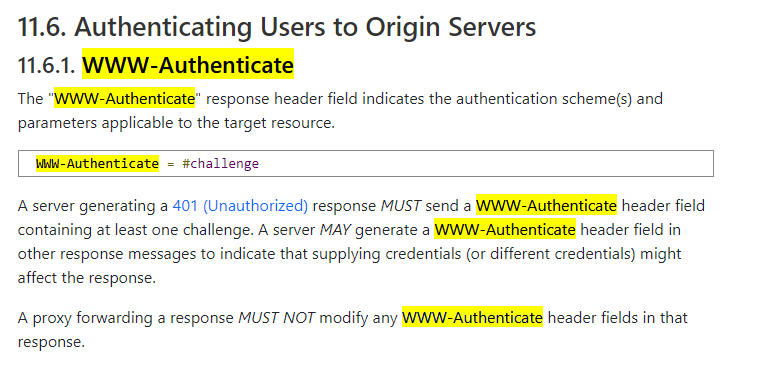


The final endpoint, /logout, allows users to log out by providing their email, password, and authentication token as parameters. Upon successful logout, the endpoint returns None.

After initially drafting the messages which do not specify response codes, I read the HTTP specification, more specifically RFC 9910, available at <https://httpwg.org/specs/rfc9110.html>. This dictates what HTTP status codes should be used with examples of scenarios, more specifically available at <https://httpwg.org/specs/rfc9110.html#status.codes>. Using this specification, decided to add the recommended status codes and headers. For example:

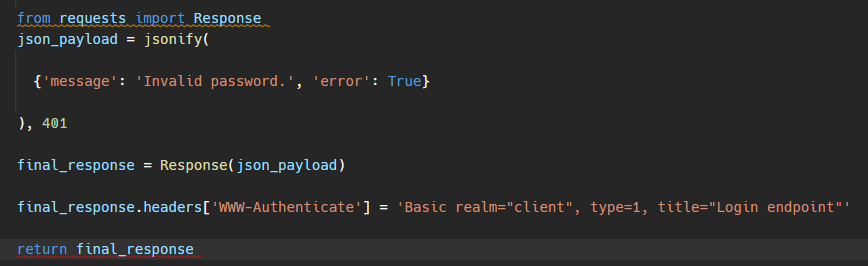


Should be replaced by specifying a ‘401’ at the end of the dictionary, and add a ‘WWW-Authenticate’ header.

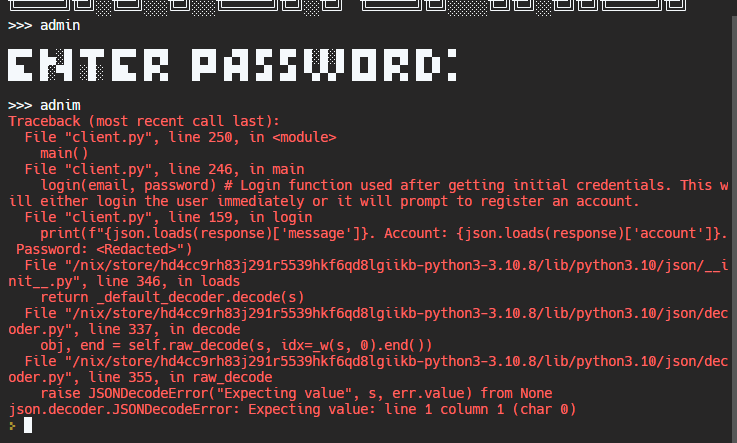


*Screenshot from the HTTP spec page, showing part of 1.16.1.*

However, I ran into an issue. As I was just returning dictionaries which could be interpreted by JSON easily on both the client and the server, I had not changed the headers at all. Using [this Stack Overflow article](https://stackoverflow.com/questions/25860304/how-do-i-set-response-headers-in-flask), I changed my code from the above snippet to this:



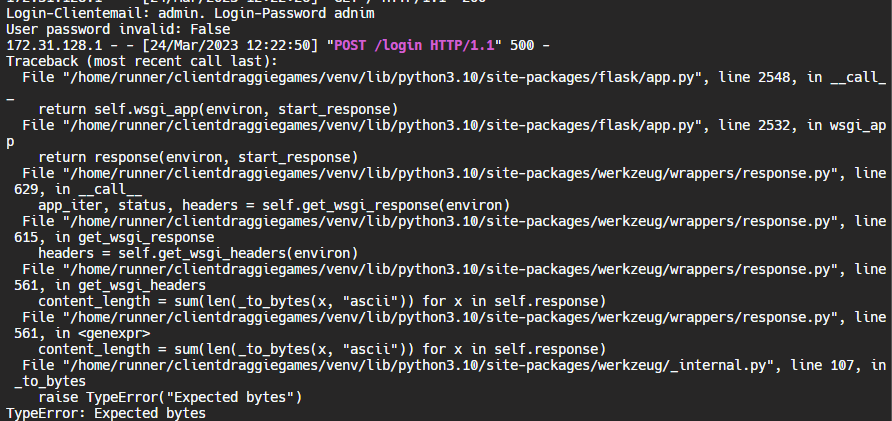
However, this seemed to be a little troubling. On changing the server script to this, attempting to log in to an existing account produced this error:



Whereas the correct login is:



The server’s console printed this output:



Throughout the process, I was using Git on top of Replit’s built-in history feature to commit and save the Python code after meaningful edits have been made.

# Original Proposal

*Advanced Water Tech Demo*

Brief overview of the project

My goal for this project will be to create a water simulating tool and game.

In this program, the user will be allowed to generate water in a specific area of 3D terrain, simulate rainfall, or simulate how water will rise up from lower areas of ground. It will also allow the user to change perspectives, place structures and edit the simulation’s terrain, as well as walk on it.

Main project objectives/success criteria

* Automatically generate a simulation
* Generate random terrain
* Create a GUI for users to interact with
* Have a list of presets which can be selected instead of random generation
* Also include a game mode, as well as the simulation mode, to appeal to a wider range of stakeholders
* Have a range of accessibility options so anyone can use the program
* Automatically keep the program up to date, without the need for a user to input any URLs or download files manually

Programming Language(s) used

To create this program, I will be using the Unity engine, which utilises C# code.

(This will ensure that the program will be able to have a GUI and stable framerates when being rendered, while also being a high-level language for ease of reading and writing. Additionally, if my school’s computers allow for it, I will use the Unreal Engine instead: it is more optimised for my specific tasks)

Why is this a suitable project?

I believe that this is a suitable project as it will be interesting to many people who enjoy subjects like Physics and Maths due to the nature of water’s physical properties, as well as relief geographers and researchers who may benefit from studying how water has an impact on a physical area. Furthermore, the game aspect of this will appeal to more users and not just students and professionals, as it will be engaging and dynamic. It is also suitable for myself as it will help me to learn how game engines work, and C# is a new language to me, so I’d like to get better at it. Penultimately, it is suitable for other reasons, as stakeholders will easily be able to suggest new ideas and help, during testing and evaluation, as it is a desktop-based application. Finally, many other features could be added to the program, such as generating images to display the terrain, connecting to a server to download updates, or adding ray-tracing.

# Code

## Unity Scripts

### PlayerMovement.cs

For a quick way to test if the monobehaviour property is correctly assigned we can use this. It continually adds a basic rigidbody force depending on the position in the scene. This can be attached to any GameObject to test if the rigidbody is applied to it correctly. The expected output is the object moving around the scene in a positive or negative X orientation every time the scene updates.

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

public class PlayerMovement : MonoBehaviour

{

    public Rigidbody rb;

    // Start is called before the first frame update

    void Start()

    {

        Debug.Log("Hello, world!");

        rb.useGravity = false;

        rb.AddForce(0, 20, -20);

        Debug.Log("Modified the initial rigidbody force towards centre");

    }

    // Update is called once per frame

    void Update()

    {

        // this ensures that we gradually travel to the centre of the scene

        // fix the bug later

        while (rb.XAxisPos > 0) {

            Debug.Log("X axis pos greater than 0, applying inverse force");

            rb.AddForce(-1; 0; 0);

        } else {

            rb.AddForce(1; 0; 0);

            Debug.Log("X axis less than 0, forcing inverse force")

        }

    }

}

### PerlinBasicCalc.cs

This was an early primitive test version of perlin noise mapping with some taken from online tutorials. https://answers.unity.com/questions/938178/3d-perlin-noise.html

using Mathf;

public static float PerlinNoise3D(float x, float y, float z){

    y += 1;

    z += 2;

    float xy = \_perlin3DFixed(x, y);

    float xz = \_perlin3DFixed(x, z);

    float yz = \_perlin3DFixed(y, z);

    float yx = \_perlin3DFixed(y, x);

    float zx = \_perlin3DFixed(z, x);

    float zy = \_perlin3DFixed(z, y);

    return xy \* xz \* yz \* yx \* zx \* zy;

}

static float \_perlin3DFixed(float a, float b)

{

    return Mathf.Sin(Mathf.PI \* Mathf.PerlinNoise(a, b));

}

### PerlinCalc2.cs

Note: This is not actively used for any MonoBehaviour but it can be assigned to something to test it.

//Start of C# script

using UnityEngine;

public class PerlinNoise : MonoBehaviour {

    //Creating a 2D array of floats to store the height values

    public float[,] GeneratePerlinNoise(int width, int height) {

        float[,] noiseMap = new float[width, height];

        //Generating a random seed

        System.Random random = new System.Random();

        Vector2[] octaveOffsets = new Vector2[8];

        for (int i = 0; i < octaveOffsets.Length; i++) {

            float offsetX = random.Next(-100000, 100000);

            float offsetY = random.Next(-100000, 100000);

            octaveOffsets[i] = new Vector2(offsetX, offsetY);

        }

        //Defining the noise parameters

        float amplitude = 1f;

        float frequency = 1f;

        //Looping through the noise map

        for (int y = 0; y < height; y++) {

            for (int x = 0; x < width; x++) {

                float noiseHeight = 0;

                for (int i = 0; i < octaveOffsets.Length; i++) {

                    //Calculating the sample points for the perlin noise

                    float sampleX = (x + octaveOffsets[i].x) / width \* frequency;

                    float sampleY = (y + octaveOffsets[i].y) / height \* frequency;

                    //Getting the noise value from Perlin

                    float perlinValue = Mathf.PerlinNoise(sampleX, sampleY);

                    noiseHeight += perlinValue \* amplitude;

                    //Reducing the amplitude and frequency for the next octave

                    amplitude \*= 0.5f;

                    frequency \*= 2f;

                }

                //Assigning the noise height to the noise map

                noiseMap[x, y] = noiseHeight;

            }

        }

        return noiseMap;

    }

}

//End of C# script

## Other scripts

### Server.py

The server code runs a Flask application in Python that allows for POST and GET requests to be listened to. If there is not a specific endpoint accessed, it will just display “Hey!” in plaintext. The login, register, get\_code endpoints all require some form of JSON payload else it will return a ‘400 Bad Request’ error. The request must also contain a valid username and password, or access token or it will return a ‘401’ unauthorised header. This all complies with the recommended HTTP RFC 9110 Semantics specification. https://httpwg.org/specs/rfc9110.html#status.400

from flask import Flask, request, jsonify, session

from werkzeug.security import generate\_password\_hash, check\_password\_hash

from flask\_login import LoginManager, UserMixin, login\_required, login\_user, logout\_user

import json, random

from uuid import uuid4

# Initialize Flask app and login manager

app = Flask(\_\_name\_\_)

app.secret\_key = "super secret key"

login\_manager = LoginManager()

login\_manager.init\_app(app)

app = Flask("")

@app.route("/")

def home():

    return "Hey!"

# Define user model

class User(UserMixin):

    def \_\_init\_\_(

        self,

        id,

        email,

        username,

        password,

        token=None,

        token\_expiration=None,

        codes=None,

    ):

        self.id = id

        self.email = email

        self.username = username

        self.password = password

        self.codes = codes if codes is not None else []

        self.token = token if token is not None else None

        self.token\_expiration = (

            token\_expiration if token\_expiration is not None else None

        )

    def check\_password(self, password):

        return check\_password\_hash(self.password, password)

    def add\_code(self, code):

        self.codes.append(code)

    def has\_code(self, code):

        return code in self.codes

# Load users from file

def load\_users():

    try:

        with open("users.json", "r") as f:

            users\_data = json.load(f)

            users = [User(\*\*data) for data in users\_data]

    except FileNotFoundError:

        users = []

    return users

# Save users to file

def save\_users(users):

    users\_data = [

        {

            "id": user.id,

            "email": user.email,

            "username": user.username,

            "password": user.password,

            "codes": user.codes,

            "token": user.token,

            "token\_expiration": user.token\_expiration,

        }

        for user in users

    ]

    with open("users.json", "w") as f:

        json.dump(users\_data, f)

# Create initial user

users = load\_users()

if not users:

    user = User(

        id=1,

        email="test@example.com",

        password=generate\_password\_hash("password"),

        codes=["testcode"],

        username="example",

    )

    users.append(user)

    save\_users(users)

# Endpoint for registering new users

@app.route("/register", methods=["POST"])

def register():

    print(f"Request JSON: {request.data}")

    email = request.json.get("email")

    username = request.json.get("username")

    password = request.json.get("password")

    for user in users:

        if user.email == email or user.username == username:

            return (

                jsonify(

                    {

                        "message": "These credentials are already in use. Please try another username, email or password."

                    }

                ),

                400,

            )

    user\_id = max(user.id for user in users) + 1

    user = User(

        id=user\_id,

        email=email,

        username=username,

        password=generate\_password\_hash(password),

    )

    users.append(user)

    save\_users(users)

    return jsonify({"message": f"User registered {email} successfully"})

def load\_user(email):

    with open("users.json") as f:

        data = json.load(f)

        for user in data:

            if user["email"] == email:

                return user

        return None

import uuid

import time

def add\_user\_token(email):

    users = load\_users()

    for user in users:

        if user.email == email:

            if user.token is None:

                # Generate new token if user doesn't have one

                user.token = str(uuid.uuid4())

                user.token\_expiration = int(time.time()) + 86400

                save\_users(users)

            return user.token

    return None

def get\_user\_token(email):

    users = load\_users()

    for user in users:

        if user.email == email:

            if (

                user.token is not None

                and user.token\_expiration is not None

                and user.token\_expiration > int(time.time())

            ):

                # Token is not expired, return it

                return user.token

            else:

                # Token is expired or missing

                add\_user\_token(email)

                return None

    return None

@app.route("/alpha", methods=["GET"])

def alpha\_key():

    # Get user's token from request headers

    token = request.json.get("token")

    # print(f"Token: {token}")

    if token is None:

        return {"message": "No access scopes provided"}, 401

    users = load\_users()

    for user in users:

        if user.token == token:

            if user.codes is None:

                return {

                    "message": "Error: No codes have been applied to your account"

                }, 401

            for codes in user.codes:

                if codes == "ALPHA":

                    current\_time = int(time.time())

                    if current\_time <= user.token\_expiration:

                        # Return alpha key to user

                        return {"message": "success", "key": "alpha\_key"}

                    # Check if current time is within token's expiration time

                    else:

                        return {

                            "message": "Error: Token is expired, please sign in again"

                        }, 401

            else:

                return {

                    "message": 'Error: No valid authentication scopes have been found for codes on this account"s tokens.'

                }, 401

    return {"message": "Nothing happenned"}, 401

# Endpoint for logging in

@app.route("/login", methods=["POST"])

def login():

    client\_email = request.json.get("email")

    password = request.json.get("password")

    print(request.json)

    for user in users:

        print(user.email)

        if user.email == client\_email:

            print(f"User email: {user.email} is equal to client email: {client\_email}")

            print(f"Password: {password}")

            if user.check\_password(password):

                user\_token = get\_user\_token(client\_email)

                user = load\_user(client\_email)

                print(f"ID: {user['id']}, User: {user['username']}")

                return jsonify(

                    {

                        "message": f"Login successful for user #{user['id']} ({user['username']})",

                        "account": user["username"],

                        "auth\_token": user["token"],

                    }

                )

            else:

                print(f"User pass: {user.check\_password(password)}")

                return jsonify({"message": "Invalid password."}), 401

    return jsonify({"message": "Invalid email or password."}), 401

# Endpoint for logging out

@app.route("/logout")

@login\_required

def logout():

    logout\_user()

    return jsonify({"message": "Logout successful"})

# Endpoint for adding a code to a user's account

@app.route("/add\_code", methods=["POST"])

def add\_code():

    email = request.json.get("email")

    password = request.json.get("password")

    code = request.json.get("code")

    for user in users:

        if user.email == email:

            if user.check\_password(password):

                user.add\_code(code)

                save\_users(users)

                return jsonify(

                    {

                        "message": f"The code has been successfully redeemed to account: {email}"

                    }

                )

            else:

                return jsonify({"message": "Incorrect password"})

    return jsonify({"message": "User not found"}), 404

# Endpoint for accessing protected data

@app.route("/api/data")

@login\_required

def access\_data():

    code = request.args.get("code")

    if code is None:

        return jsonify({"message": "Code not provided"}), 400

    user = load\_user(session.get("user\_id"))

    if user.has\_code(code):

        with open("data.txt", "r") as f:

            data = f.read()

        return jsonify({"data": data})

    else:

        return jsonify({"message": "Unauthorized"}), 401

# Run the app

if \_\_name\_\_ == "\_\_main\_\_":

    app.run(host="0.0.0.0", port=random.randint(2000, 9000), debug=True)

### Client.py

### Client-GUI.py

This is very similar to Client.py but is a GUI version which took a lot more time! It uses PyQt4 as the graphics framework which acts as a wrapper for a C++ program.

1) My current solution to the problem is to use a game like Minecraft as a starting point.

2) What is wrong with my current solution is that Minecraft does not have the realistic water physics. Additionally, Minecraft does not have the same level of graphical fidelity or user interface (GUI) design as my game requires.

3) I would want the program on a laptop or desktop computer.

4) I am currently using Windows 10 on my laptop.

5) The specifications of my system are an Intel Core i7 processor, 8GB of RAM, and a 1TB hard drive.

6) The resolution of my display is 1920x1080.

7) After seeing this idea, what is wrong with this solution is that it does not have the necessary tools and technologies to create the game I am envisioning. It does not have the ability to create realistic water physics, random terrain generation, or a doomsday scenario. Additionally, it does not have the graphical fidelity or user interface design necessary for this game.

8) After seeing this idea, features I would like to add include realistic water physics, random terrain generation, a doomsday scenario, and high-quality graphics and user interface design.

9) I am familiar with the Unity development engine and C# code language.

10) I am familiar with designing game art and have designed graphical user interfaces (GUIs).

11) I am prepared to install alpha testing software on my device.