

Computing OCR A level coursework

Philip Eagles – 12/13CT RGSHW

Candidate number: 2347

Centre number: 52423

This documentation contains videos which are not currently viewable in Google Chrome and several other PDF viewers. Please open in Adobe Acrobat Reader/DC © with the flash player plugin to view any video content.

The PDF generation has also severely reduced video quality. The original files were full HD at 30fps, and the project itself will run at any resolution at a maximum of 60fps.

Gantt chart

A Gantt Chart will be frequently referenced throughout this documentation. This Gantt Chart is included as a zip file, which contains both the original spreadsheet made in Microsoft Excel® and an image of each sheet/tab (zooming in recommended).

The Gantt Chart is separated into 4 tables for better showing specific parts of the spreadsheet, all of which are shown in the master table.

The Gantt Chart references testing through numbers (e.g. 28.1). The corresponding test can be found in the development section. There is a page of hyperlinks that can be found [here](#).

Contents

- Analysis
 - Stakeholders
 - Problem Identification
 - Research the problem
 - Proposed solution
- Design
 - Problem decomposition
 - Solution description
 - Testing plan
- Development
 - Iterative development
 - Testing
- Evaluation
 - Testing
 - Solution success
 - Final product
 - Maintenance and development
- Bibliography

Analysis

Background information

- Canary Development Studios is a software development company that covers the UK (excluding Northern Ireland). They typically work by hiring entrepreneurs to join their company, and then offer them a full time job if they are satisfied with the results. This is to help increase productivity and originality in work, as entrepreneurs are typically good at spotting gaps in the market, something that Canary Development Studios can use to their advantage.
- Canary Development Studios was founded in 2004 as an indie games company by a group of three university students – Carl Marx, Lydia Baughman and Trent Wayne. Since then they have expanded to become a national company that employs over 200 workers in 9 different cities. Their most famous project has been Revolved, their debut project, selling over 179,000 copies virtually.
- Since then they have also created utility software that helps to optimise system usage and bot times, as well as helping to make software packages to aid school learning (such as plugins to visually represent programming languages like C, Java and VBA).
- In more recent years they have chosen to work more and more with school aged children, trying to help educate students on programming techniques.

Email communications

As can be seen over the next few pages, I have been contacted by a representative from Canary Development Studios to create a project for them. They have made several very clear specifications, but have left large parts of the program up to me to decide how they will run, look and function.

This means that I will have to choose what I will be doing, justify why I have chosen this approach, and how I can achieve this.

Email communications

Dear Mr Eagles,

We at Canary Development Studios have heard that you have offered your services to create a project tailored to your employer's needs. We were wondering if we could be such employers, and whether your experience in web based applications could be used by us to produce a game. We wish to feature the project as a showcase for a student session that will be coming up, where the students will be given a fully functioning program and must attempt to alter and change the code (or "mod" it). This means that we will require you to code in such a way that a person with basic programming skills will be able to look at your code and understand what it is doing.

If you choose to accept our proposal, then we will also be asking for full documentation, including a manual or tutorial on how to use the game.

Yours sincerely,

Joseph Munt

(Head of External Affairs for Canary Development Studios)

Dear Joseph Munt,

I would be more than happy to accept your offer for working for Canary Development Studios, your proposition sounds very interesting and I believe that I am up for the challenge of making code that can be modified for further use.

Would it be possible for me to know more about the kind of people that will be using the program (such as age and nationality) to help create a project that is more applicable to them? Additionally, any plans for what the project should be like would be much appreciated.

Yours sincerely,

Philip Eagles

Email communications

Dear Philip Eagles,

The project is aimed at secondary school students in the UK in years 10 and above. We were hoping that you could create a maze game with multiple predetermined levels in it. We wish for there to be a client side save system as we wish to make some of the "mods" by the students change the save system to be server based, as we believe that this will be most beneficial for teaching them PHP which is a valuable skill. This means that the save system should use a technique that would not be too different to that of PHP, namely an SQL database stored on the client's machine. The project will be saved on a local network, so the users will not have to use the same machine to have all of the save data accessed. I realise that this may be limiting to what you can therefore achieve, but this is a necessary requirement.

Another requirement is that you use javascript and not jQuery - we are attempting to teach the students how to program in the original form, rather than using the jQuery library. Again, I can understand that this may be a change from your normal work, however I hope that you will be able to manage this.

Our last requirement is that it will be possible to "inject" more levels into the code. We are hoping that the students will be able to customise almost all attributes of the project, and creating their own levels would be highly useful. Another idea that we would like the students to be able to do is work on procedural level generation, so again we ask that you do not include a feature for this.

Beyond this, we are happy to let you have relatively free reign on what you do for the project. With the exception of jQuery, we are more than happy for you to use external libraries of scripts.

In terms of the user's architecture, we wish for it to be cross platform compatible, hence the idea of a web-based application. We are asking that you aim to have the program run on Google Chrome as this is one of the better internet browsers in our opinion. Feel free to use features exclusive to this browser, as long as they are relatively stable.

For art style, may I request that there is a relatively simple theme of drop shadows for the main menu, and just black and white for the mazes themselves?

Best wishes,
Joseph Munt
(Head of External Affairs for Canary Development Studios)

Email communications

[REDACTED]
Dear Joseph Munt,

I can of course refrain from using PHP, despite the obvious fact that this may limit some of the features. I propose that I will instead use webSQL which is a now depreciated client side SQL database creator and editor based off SQLite3. This will create a physical file of the database on the system. As you mentioned that the main browser will be Google Chrome, this will not be an issue as this is implemented to its full extent.

I am glad that you do not wish for me to use jQuery as I prefer to code in "pure" JavaScript. I will make sure to use logical function and variable names, as well as including comments in my code to explain what I am doing so that a new user can read the code and understand what is happening.

I will create a part of the program that explains how to create user levels, and will make sure to only have a set amount of premade levels, but make it possible to add more. This will mean that the users will be able to mod and create their own elements.

I will make sure to stick to your art scheme, and will send you a rough draft of the stylism once I have completed further analysis of the project (such as interviews).

Best wishes,
Philip Eagles

Dear Philip Eagles,

Thank you very much for your co-operation. Please send us the results from your questionnaire and regular updates on how you are progressing with the design and programming.

Best wishes,
Joseph Munt
(Head of External Affairs for Canary Development Studios)

Client requirements

- Create a maze game
- Have multiple predetermined levels
- Have it possible to add code to the project
- Have it possible to add new levels to the project
- Must not include procedural level generation
- Use HTML and JavaScript as the main programming languages
- Primarily black and white art style
- Cannot use PHP or jQuery
- Use SQL tables to save data
- Prioritise Google Chrome as the main platform
- Create a tutorial or user manual on how to use the game
- Create a documentation writeup (this document)
- Targeted end user's are aged 14-36 years, and of no specific gender or nationality
- Targeted end client is 14-18 years old, and of no specific gender or nationality

My assumptions

- Use webSQL to create databases
- Use code comments
- Use meaningful variable names
- Use meaningful function names
- Have a user login system running on a network
- Have individual user accounts with saved data
- The project will be deployed through Canary Development Studios website to registered users, where it will be available to download
- The end user will be casual gamers
- The end client will have prior programming knowledge
- The user will have their times taken to complete each level saved

Why use a computer?

- This project cannot be done without a computer as it will need very precise timing on level completion times.
- Additionally, the visual aspect of the game will not fit into any existing board game styles, or indeed any other kind of game (short of creating a life sized maze for players to run around in which is impractical as well as expensive) other than using a computer.
- It will also be quicker to use a computer to sort times to find the quickest time than using pen and paper.

Project importance

- As I will be working for a large corporation, I need to ensure that my work is to a high standard. Failure to do this could result in Canary Development Studio (CDS) losing money, so should be avoided at all costs.
- CDS are asking for on-going progress on the project, meaning that I will need to submit my work to them at regular intervals for approval. One of the first plans that they want to see is the GUI plan.
- Further communications have provided me with the insight that work will be assessed by a board of managers from various departments (design, programmatic interpretation, coding, testing etc.) and I will be given approval based on this. They will not be giving me in depth feedback, but rather plan on simply stating whether they are happy with progress or not.

Current similar projects

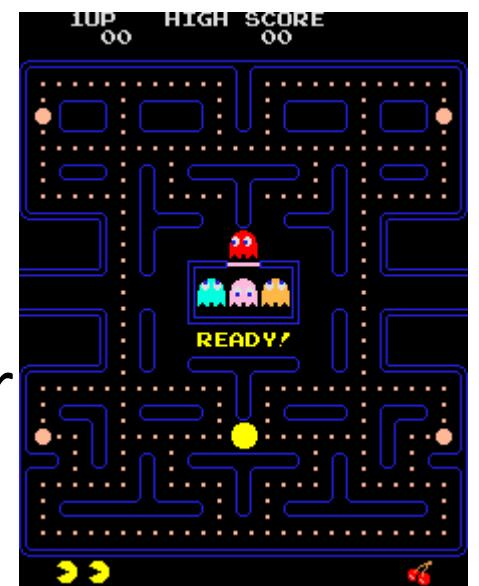
- A top down maze game is nothing new and can be traced back to the classic arcade game Pac-Man released in 1980 (and further back too, but this is a well known example). This means that I will need to do something to make the game unique and stand out in the market.
- Games with timers in them are also not a new concept. Even older racing games have options to do timed laps/courses which can display a scoreboard of fastest to slowest times.
- Typical controls for any game on a PC are either the arrow keys or the w, a, s and d keys to move the player. Many games also make use of the spacebar, for example to jump over an obstacle. However, as this is a top down game, it is difficult to see the need to jump in it.

Current similar projects – 1

A top down maze game is nothing new and can be traced back to the classic arcade game Pac-Man released in 1980 (and further back too, but this is a well known example). This means that I will need to do something to make the game unique and stand out in the market.

As can be seen, it has a built in score and enemies that chase the player around, as well as a life system so that the player can have multiple attempts at getting a new record.

Many users stop playing this game because it only has 1 level design, so become bored of the game. Knowing this, I can see that I should include multiple different levels rather than just 1 level that is always used



Current similar projects – 1

The original version of Pacman used a joystick to control the player, but has since been adapted to directional keys on a keyboard (e.g. arrows or wasd). These controls are more modern and work just as well, as allow for quick changes in direction.

The ghosts in the game do not use pathfinding, and instead simply choose a “random” direction at each junction, meaning that they can appear to chase the player with very little computation. More modern versions have introduced a feature where the ghosts can lock onto the player when they have been directly “seen” (i.e. not through a wall), and are harder to get rid of. This presents a more challenging scenario which some players prefer.

Current similar projects – 1

Pacman has a scoring system of collecting small white dots that are at regular intervals around the level. These, when collided with by the player, increase the user's score. When the player loses, the score is compared to other previous scores, and if high enough can then be saved to a highscore table where the user can choose a name to go with the score. Older versions of the game were limited to 5 characters, but newer versions have removed this restriction.

Additionally, after a random number of seconds, a bonus collectible may spawn (in the shape of a fruit) which the player can collide with to earn a significant number of points. The fruit vanishes after a number of seconds if the player doesn't get to it in time.

Current similar projects – 1

Pacman has a life system which allows the player to make mistakes and not have to immediately restart the game. The default number of lives is 3, but different versions have different amounts. Some also have ways of gaining more lives, but this is not in the original. Lives can be lost by colliding with an enemy ghost.

Enemy ghosts can be temporarily chased by collecting a larger white dot, causing them to change colour and move slower. During this time, the player may “eat” the ghosts which will give them additional points. The ghosts then return to the centre of the screen and respawn and go back to chasing Pacman.

Current similar projects – 2

Games with timers in them are also not a new concept. Even older racing games have options to do timed laps/courses which can display a scoreboard of fastest to slowest times.

One example of this would be the 1992 game Super Mario Kart, where the player can compete with other players and the computer AI to try to complete a course in record time. It has 2 display options – a top down minimap of the whole level, and a third person view of the player and level. The time taken is recorded, and a record can be shown to the player.



Current similar projects – 2

This game has been modernised several times (the 9th release has been released for the Nintendo Switch ©), but the core principal has remained the same. The player must drive a car through a predetermined level and try to beat the other players and/or enemy AI. In more recent versions, you can choose different characters to influence the car's handling, acceleration and so on.

The enemy AI's difficulty can be chosen before the race starts, which determines how difficult it can be for the player to win. A harder AI means that they make fewer mistakes and also drive slightly quicker than before.

Current similar projects – 2

If the player wins a level, they have the option to save their time to a highscore table which can be compared to other player's. More recent versions of the game allow for sharing over wi-fi.

It is possible for the player to take damage in the game, for example by driving into an area of water. This results in a 3 second penalty, but puts the player back on the course so that they may continue the race.

There are also special items which can be collected and used to help the player or enemy AI. These include projectiles to damage a car, speed boosts and other damaging items. Some of these will chase the player down and are very difficult to avoid.

Current similar projects – 3

The idea of shadows is also not new. Taking the game Luminesca as a more recent example, we can see that a player casting light from itself as well as other non-player sources has been done before. This game is still in a developmental stage, so some elements of it may change over time, but the core principles will remain the same.

This game is a simple exploratory adventure game with some basic storytelling. It does not have any time features included.



Current similar projects – 3

Luminesca is currently in BETA development, but the project was first released in 2012. It is still being added to and optimised.

The game is set in an underwater cave system which the player can travel through in a linear fashion. The player may encounter obstacles that require them to flip a switch or similar to move out of their way and continue exploration. There are also enemy AIs that can chase the player down, and if they catch the player then they can cause the player to have to restart a given section. A player may attempt a section as many times as they would like, and can choose to go back and replay a section.

Current similar projects

- None of these systems have been combined together to create a unique game like I plan to do, meaning that there is an open space in the market for a product like this.
- Canary Development Studios seem to have already identified this, which is why they are funding the project.
- Canary Development Studios are aiming to make this game open source as a resource to aid programmers in learning from other people's unique style of coding, meaning that it can be modified to add extra features which they have the right to publish.

Questionnaire

- To help gather feedback, I will be asking several people what they think could be used in the project to make it interesting, unique and enjoyable for them, as well as other questions about their perception of the end user.
- They will be shown the email communication prior to being asked the questions so as to have a basic understanding of what is wanted.
- They will not be shown my notes, as this may influence their decision making process.

Questionnaire

- Who will be using this product?
- What is your idea of what the project will be like?
- What must it include?
- What must it not do?
- How do you want to control the program?
- Any suggestions or extra features?

Sample data 1 – anonymous identity

- I can imagine this product being used by mainly children to young adults who enjoy playing video games in their spare time. I do not think that it will be best suited to one gender, but rather enjoyable for a wide range of gamers. However, as it is also editable, I feel that the modders will be mostly teenagers between 13 and 20.
- I am imagining it as being a maze game where there are premade levels that the player must traverse. I think that the player should have a limited view and that there should be obstacles in the player's way. These obstacles are walls that stop the player from passing through them. There should be a time trial mode where the aim is to complete the level as fast as possible. It should record a high score which can be viewed by other users. Each user should have their own save for their own high scores and progress, but should be able to share scores with others.
- It must include a player and a maze. Parts of the maze far away from the player must be hidden. The user must be able to control the player and choose things like levels.
- It mustn't be too complicated, so tutorials would be helpful.
- I want to use the arrow keys to move the player, but also use the mouse for things like buttons to choose levels.
- Definitely the high score system, I think it would make the game more competitive and enjoyable. Collectibles would be interesting too, with things you can unlock by collecting more of them such as style changes or upgrades to help complete levels quicker.

Sample data 2 – Kiran Rajappan (student)

- Pretty much any people to be honest. Well, maybe not so much children under 8 or something, they wouldn't be able to understand it I feel.
- I want to have a game that I play in my spare time. It's got to have sick graphics and run at least 60fps (more is better). I want it to be able to run in 5k on my iMac. I want to be able to beat other players because I'm a pro and love playing video games.
- It must be easy to use but requires some practice to get amazing at. It must save my progress so I can compare how good I am with other people.
- It can't have someone else using my account, so give it passwords to make sure someone can't hack me.
- I want to use a keyboard like all PC games with the wasd keys to move.
- I guess it should have a shop for buying upgrades and stuff, like customising colours and making things go faster or easier to use. A highscore table is important too because I need to see who to beat.

Sample data 3 – Jean-Remy Duboc (Canary Development Studio employee)

- I think the target audience should be casual pc gamers, so not requiring high spec machines. It should be kept gender neutral and also aimed for people of any age. The modification side should be aimed at people with some programming experience so they shouldn't have to be hand held through how to change anything.
- I feel that it should be a top down maze game, where the player has limited view around them. This could be done with lights around the player to reduce visibility far away, otherwise the player can immediately see the full maze which is less interesting as it can be solved immediately.
- A player is the obvious starting point with a maze around it. A start and end point should be in the maze. The maze should have a time trial aspect, with a highscore that different users can see and compete for. Assuming there are different levels, there should also be a level choosing screen.
- Break obviously! It needs to be stable and run consistently without errors.
- I personally like to use the arrow keys to navigate in PC games, but I know that other people use the wasd keys so both should be optional.
- If it is aesthetically pleasing then that can make it more enjoyable. I envisage a very simple art style with mainly monochrome colours. Options to change the colour scheme would be very interesting, and maybe bonus levels that you unlock by completing achievements or something similar.

Sample data 4 – Lucy Adams (student)

- Any programmer wanting to learn to use someone else's code would be the obvious one, which would include most students in their later years as companies like people that are able to work together on code.
- A maze game with a top down perspective of each level, with more than 1 level.
- Obviously a player, maze, selection screen and timer. If there is a timer then there should also be a highscore screen with a player name to compare people's best results. Comments in the code would aid any programmer too
- I would hate to see a project where the programmer can't understand the purpose of a bit of code, so avoiding silly variable names would help.
- I want to be able to change the game file(s) to allow me to mod the code and add my own features. I want to be able to just do this and then rerun the program with the changes in place.
- A save file system with administrators would be pretty good in case someone forgets their password and can't log in.

Conclusions

- From this questionnaire, I have gathered that people would like to not be able to see the full level straight away as this detracts from the enjoyment of the level.
- Players wish to use the keyboard keys (either wasd or arrow keys) to move around.
- The code itself should have comments to make it easy to understand.
- Each level should have a predetermined start and end to it.
- Players would like password protected accounts to score the best times with their username.

Which programming language?

I will be making this in HTML and JavaScript as this has built in cross platform compatibility. It also has Scalable Vector Graphics (SVG) which will allow for the program to work in any resolution. HTML also has support for SQL which is very important for the format of the save files. I can save extra data as cookies which will mean that there is a file that is stored on the computer for a fixed length of time.

I will have to make sure that if I use any “experimental” features (particularly relating to CSS animations) that I implement them so that they will be reliable and not cause any issues.

I plan to use webSQL as my interface between SQL and HTML/JavaScript.

Hardware/software requirements

- The initial download of the file will require internet access, as the file will be downloaded from Canary Development Studios' official website after signing in with a valid account.
- A system administrator will need to save the file into a publicly accessible location, common to all users (e.g. “Program Files” folder on a Windows © device).
- I will be developing on a somewhat low-end laptop which will be used as a basic benchmark for whether the project runs fast enough. The specs of the laptop are 6 GB of RAM, 2 1.6 GHZ cores (Intel Celeron), a built in graphics processor (Intel HD Graphics, 256MB VRAM), and running Windows 10 (64 bit).

Hardware/software requirements

- Chromebooks (Canary Development Studios targeting platform) will typically have at least 8GB RAM, 2 2.0GHZ cores, a built in graphics processor with 512MB of VRAM, and a 64 bit processor, making them superior to my laptop, meaning that if my laptop can run the program successfully then there is no reason that there's can't too. The program will require a monitor, keyboard and mouse as the main I/O devices. I plan on needing speakers as well for audio output.
- The main browser to target will be Google Chrome, which is well optimised and highly capable for graphics processing. I will also try to make sure that Safari and Opera will function with the program, as this will allows for users to choose a different browser. I will not be working with the Internet Explorer or Mozilla Firefox families, as they do not support WebSQL so could not run the program.

Objectives – usability

All objectives on the project itself are shown in the Gantt Chart, and are best viewed in the “Objectives” tab/sheet.

Some example data is shown below.

Task ID	Main Task	Intermediate Task	Sub Task	Justification
1	Game Immersion			Increases enjoyment for the player by removing distractions and making the game more "real"
1.a		Fullscreen		Hide other windows so reduce distractions
1.a.i			Detect page size	Find the size of the currently rendered webpage
1.a.ii			Detect screen size	Find the size of the screen with the webpage in it
1.a.iii			Show message if page size is not screen size	Compare the sizes, and if not matching then the page is not fullscreen so a message is shown
1.a.iv			Hide message if page size is screen size	Compare the sizes, and if not matching then the page is fullscreen so the message is hidden
1.a.v			Pause game if page size is not screen size	If not fullscreen, then forces the player to be in fullscreen by preventing the game from playing
1.b		Sound effects		A common part of any game, makes the game more interesting and intuitive for the user
1.b.i			Play audio files	Causes audio to play
1.b.ii			Event based triggers	Audio will play under certain actions e.g. completing a level
1.c		Ambience effects		Background hum/wind sounds during gameplay to create an atmosphere
1.c.i			Play audio files	Causes audio to play
1.c.ii			Loop audio files	Makes a continuous background sound
1.c.iii			Fade in audio	Removes "hard" transition when ambience begins
1.c.iv			Fade out audio	Removes "hard" transition when ambience ends
1.c.v			Layer audio files	Allow multiple files to play at the same time
1.c.vi			Randomise audio selection	Create a more natural background effect

Design

Objectives – performance

Objective	How to check if works/exists	Importance (/10)
In-game shadows (raytracing)	Play a level, move	10
In-game acceleration (friction)	Play a level, move	10
Login loading time	Login	6
Game loading time	Launch game	7
Level loading time	Load level	6
In-game framerate	Play a level	9
Button animations	Click/mouse over buttons	8
Screen scaling	Run in different resolutions	10

Objectives – reliability

Objective	How to check if works/exists	Importance (/10)
Key pressing (which key)	Play level, move	10
Username length	Try too short username	10
Username characters	Try bad characters in username	10
Password length	Try too short password	10
Password characters	Try bad characters in password	10
Password hashing	Check SQL tables for hashed password	8
Password's matching (registration)	Try mismatching passwords	10
Control setting (changing keys)	Change control scheme, play level	10 (main objective 8)
Loading user data	Login with a pre-existing user	10
Saving user data	Logout with changed user data	10

Objectives – maintainability

Objective	How to check if works/exists	Importance (/10)
Meaningful variable names	Check new coder can understand variable purpose	10
Meaningful function names	Check new coder can understand function purpose	10
Comments in code	Check new coder knows what is happening in code	8
Logical use of functions	Make sure not to make many useless functions that rely on other functions to create messy code	7
Logical array structuring	Make sure arrays follow a sensible format that can be understood	5
Backup code	Keep code backups in case of serious error	10
Up to date documentation	Write documentation while coding	8

Computational methods

- The program will need to be broken down into the following methods, taking a divide and conquer approach:
- Main game
 - Moving the player
 - Player colliding
 - Screen scrolling
 - Loading the level
 - Clearing the level
 - Drawing the shadows
 - Creating the enemies
 - Making the enemies move
 - Passive wandering
 - Aggressive chasing
 - Taking damage
 - Collectibles
- Menus
 - Options
 - Sound
 - Controls
 - Level selector
 - Scrolling between pages
 - Triggering the game to begin
 - Help
 - In game tutorials
 - Shop
 - Negating currency
 - Changing appearances
 - Log in
 - Separate users
 - Access rights
 - Create new users
 - Validate username existing
 - Username and password check
 - Hashing

Computational methods

- Output types:
 - Time (elapsed)
 - Visual elements (e.g. GUI changes, game display)
 - Sound effects
- Input types:
 - Keyboard
 - Mouse
 - Time (begin/end)
- Processing methods:
 - Take a begin time and end time, and subtract from each other to find the elapsed time.
 - On a keyboard press, move the player in the screen.
 - On a mouse click, have the GUI respond to if a button is pressed.

Program tractability

- Due to the program being based around a fixed number of levels, the processing time will scale linearly (big O notation being $O(kn)$ where n is the number of levels and k is the scaling factor) with the number of levels for query results regarding level loading.
- Getting a record time will have a linear time complexity per level for the number of users (big O notation being $O(kn)$ where n is the number of users and k is the scaling factor).
- Settings and save files will also scale linearly, meaning that if there were 10,000 users then the program may take some time to query the user but this will work well for fewer users (which is what I am anticipating)

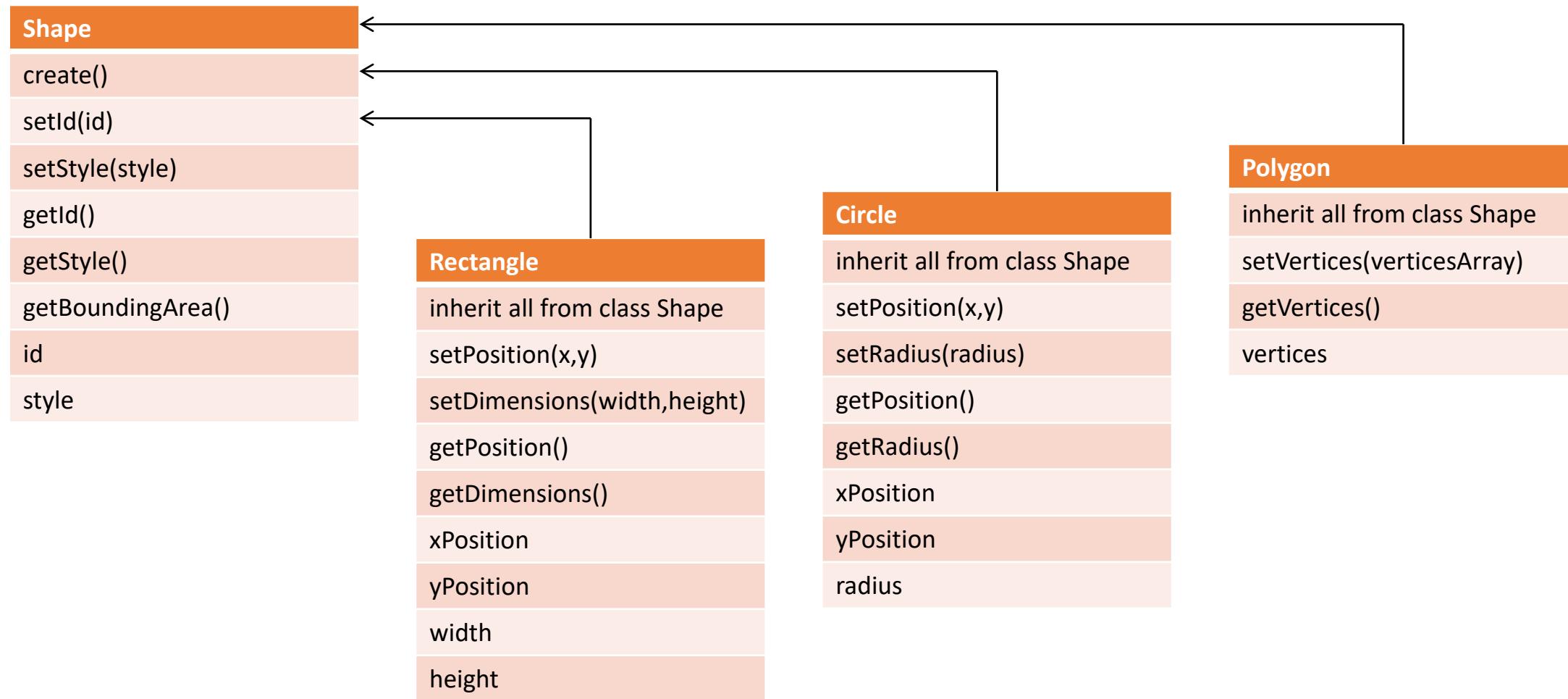
Possible limitations

- Enemies will be difficult to make path find around the level design.
 - Each level will be unique and so cannot share routes with each other.
 - Enemies will need to deviate from various routes to chase the player.
 - After some more time they need to return to the normal routes if they lose the player.
 - This means that I will not prioritise this, especially as it is not a pre-stated requirement from CDS.
- The shadows may be computationally expensive to render in real time.
 - Each shadow is a polygon, where each side needs to have a calculated gradient, start point and end point.
 - Each shadow will be redraw multiple times per second.
- Hashing may not be secure as all processes are carried out on the client side, so the code to generate the hash can be seen.
 - Hashing is normally done on a server where the code cannot be viewed.

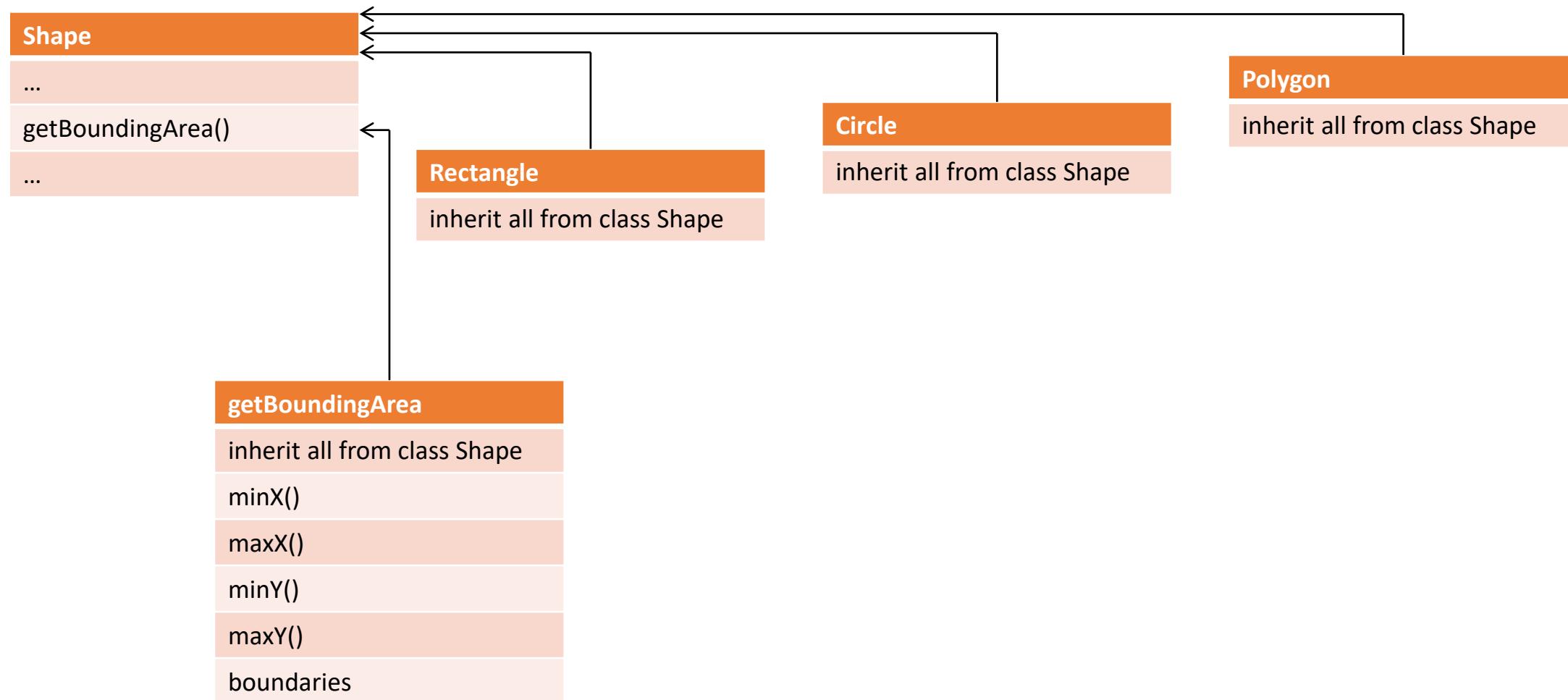
Scheduling plan

- | | | |
|---------------------|------------------------------|------------------------------|
| 1. Level generation | 9. SQL databases
(levels) | 16. User times |
| 2. Player movement | 10. Level exit | 17. User account
controls |
| 3. Screen scrolling | 11. Separate users | 18. Damaging obstacles |
| 4. Player collision | 12. Hashing | 19. Help menu |
| 5. Level shadows | 13. User login | 20. User settings |
| 6. Level selection | 14. User registration | |
| 7. Loading levels | 15. Main menu | |
| 8. Saving levels | | |

UML diagrams



UML diagrams



UML diagrams

ScreenCanvas	KeyBindings	Buttons	Users
create()	setDown(which, state)	create()	create()
appendShape(shape)	getKey(which)	setId(id)	setDefault()
removeShape(shape)	containsKey(which)	setText(string)	setId(id)
getShapes()	setKey(which, key)	setStyle(style)	setName(name)
setXScroll(x)	keysDown()	getId()	setAdmin(boolean)
setYScroll(Y)	keyBinds	getText()	getId()
getScroll()	states	isClicked()	getName()
setWidth(width)		remove()	getAdmin()
setHeight(height)		id	id
getSize()		text	name
shapes		x	admin
xScroll		y	
yScroll		width	
width		height	
height		clicked	
		style	

UML diagrams

PurchasableItem
create()
setId(id)
setPrice(price)
setTitle(title)
setDescription(description)
setBought(boolean)
getId()
getPrice()
getDescription()
Title
getBought()
id
price
title
Description
bought

Pseudocode – class generation (shape)

```
CLASS Shape
    PRIVATE id
    PRIVATE style
    PUBLIC PROCEDURE create()
        id := null
        style := null
    END PROCEDURE
    PUBLIC PROCEDURE setId(unique)
        id := unique
    END PROCEDURE
    PUBLIC PROCEDURE setStyle(newStyle)
        style := newStyle
    END PROCEDURE
    PRIVATE FUNCTION getId()
        RETURN id
    END FUNCTION
    PRIVATE FUNCTION getStyle()
        RETURN style
    END FUNCTION
    PRIVATE FUNCTION getBoundingArea()
        RETURN [style minX, style minY, style maxX, style maxY]
    END FUNCTION
END CLASS
```

Pseudocode – class generation (rectangle)

```
CLASS Rectangle INHERITS Shape
    PRIVATE xPosition
    PRIVATE yPosition
    PRIVATE width
    PRIVATE height
    PUBLIC PROCEDURE create()
        super.create()
    END PROCEDURE
    PUBLIC PROCEDURE setPosition(x, y)
        xPosition := x
        yPosition := y
    END PROCEDURE
    PUBLIC PROCEDURE setDimensions(w, h)
        width := w
        height := h
    END PROCEDURE
    PRIVATE FUNCTION getPosition()
        RETURN [x, y]
    END FUNCTION
    PRIVATE FUNCTION getDimensions()
        RETURN [width, height]
    END FUNCTION
END CLASS
```

Pseudocode – class generation (getBoundingClientRect)

```
CLASS getBoundingClientRect() INHERITS Shape, Shape.sub // gets the Shape's child element as well as the shape
    PRIVATE boundaries
    PRIVATE FUNCTION getBoundingClientRect()
        child := super.sub
        IF child.className = "Rectangle" THEN
            position := child.getPosition()
            dimension := child.getDimensions()
            RETURN [position[0], position[1], dimension[0], dimension[1]]
        ELSE IF child.className = "Circle" THEN
            position := child.getPosition()
            r := child.getRadius()
            RETURN [(position[0] - r), (position[1] - r), (position[0] + r), (position[1] + r)]
        ELSE
            nodes := child.getVertices()
            output := [null, null, null, null]
            FOR I := 0 TO I = nodes.length()
                IF nodes[I][0] < output[0] OR output[0] = null THEN
                    output[0] := nodes[I][0]
                ELSE IF nodes[I][0] > output[2] OR output[2] = null THEN
                    output[2] := nodes[I][2]
                END IF
                IF nodes[I][1] < output[1] OR output[1] = null THEN
                    output[1] := nodes[I][1]
                ELSE IF nodes[I][1] > output[3] OR output[3] = null THEN
                    output[4] := nodes[I][3]
                END IF
            NEXT I
            RETURN output
        END IF
    END FUNCTION
    PRIVATE FUNCTION minX()
        array := getBoundingClientRect()
        RETURN array[0]
    END FUNCTION
    PRIVATE FUNCTION minY()
        array := getBoundingClientRect()
        RETURN array[1]
    END FUNCTION
    PRIVATE FUNCTION maxX()
        array := getBoundingClientRect()
        RETURN array[2]
    END FUNCTION
    PRIVATE FUNCTION maxY()
        array := getBoundingClientRect()
        RETURN array[3]
    END FUNCTION
END CLASS
```

Pseudocode – level loading

```
PROCEDURE loadLevel(levelData)
    FOR each IN levelData
        IF each.shapeType = "rectangle" THEN
            shape := Shape.Rectangle.create()
            shape.setId(each.id)
            shape.setPosition(each.x, each.y)
            shape.setDimensions(each.width, each.height)
            shape.setStyle(each.style)
        ELSE IF each.shapeType = "circle" THEN
            shape := Shape.Circle.create()
            shape.setId(each.id)
            shape.setPosition(each.x, each.y)
            shape.setRadius(each.radius)
            shape.setStyle(each.style)
        ELSE
            shape := Shape.Polygon.create()
            shape.setVertices(each.verticesArray)
            shape.setId(each.id)
        END IF
        ScreenCanvas.appendShape(shape)
    NEXT each
END PROCEDURE
```

Pseudocode – player movement, collision

```
BEGIN
    xvelocity := 0
    yvelocity := 0
    xposition := 0
    yposition := 0
    player := Shape.Rectangle.create()
    speedMultiplier = 1
    WHILE playing = TRUE
        clearShadows()
        IF KeyBindings.getKey("up") = TRUE THEN
            yvelocity := yvelocity + (2 * speedMultiplier)
        END IF
        IF KeyBindings.getKey("down") = TRUE THEN
            yvelocity := yvelocity - (2 * speedMultiplier)
        END IF
        yvelocity := yvelocity * 0.9
        yposition := yposition + yvelocity
        player.setPosition(xposition, yposition)
        IF checkCollision() = TRUE THEN
            yposition := yposition - yvelocity
        END IF
        IF KeyBindings.getKey("right") = TRUE THEN
            xvelocity := xvelocity + (2 * speedMultiplier)
        END IF
        IF KeyBindings.getKey("left") = TRUE THEN
            xvelocity := xvelocity - (2 * speedMultiplier)
        END IF
        xvelocity := xvelocity * 0.9
        xposition := xposition + xvelocity
        player.setPosition(xposition, yposition)
        IF getCollision() = TRUE THEN
            xposition := xposition - xvelocity
            player.setPosition(xposition, yposition)
        END IF
        FOR I := 0 TO I = ScreenCanvas.length()
            createShadow(ScreenCanvas[I])
        NEXT I
    END WHILE
HALT
```

Pseudocode – key stroke detection

BEGIN

 WHILE TRUE // hardware is assumed to be a premade class used by
 the computer as part of the built in Input Output system of the OS
 IF hardware.peripheral.keyboard.keyDown.any() = TRUE THEN
 key := hardware.peripheral.keyboard.keyDown.which()
 IF KeyBindings.containsKey(key) THEN
 KeyBindings.setDown(key, TRUE)
 END IF
 check := KeyBindings.keysDown()
 FOR I := 0 TO I = check.length()
 IF check[I] <> hardware.peripheral.keyboard.
 keyDown.which() THEN
 KeyBindings.setDown(key, FALSE)
 END IF
 NEXT I
 END IF
 END WHILE
HALT

Pseudocode – collision detection

```
FUNCTION checkCollision()
    hitbox := player.getBoundingArea()
    marker := FALSE
    FOR each IN level_objects
        obstacle := each.getBoundingArea()
        IF ((hitbox.maxX() <= obstacle.maxX() AND hitbox.maxX() >=
obstacle minX()) OR (hitbox.minX() <= obstacle.maxX() AND hitbox.minX() >=
obstacle minX())) AND ((hitbox.maxY() <= obstacle.maxY() AND hitbox.maxY() >=
obstacle.minY()) OR (hitbox.minY() <= obstacle.maxY() AND hitbox.minY() >=
obstacle.minY())) THEN //checks if the player is within the bounding area
of the object
            marker := TRUE
            BREAK FOR
        END IF
    NEXT each
    RETURN(marker)
END FUNCTION
```

Pseudocode – screen scrolling

BEGIN

```
    canvas := ScreenCanvas.create()
    canvas.setXScroll(0)
    canvas.setYScroll(0)
    canvas.setWidth(hardware.peripheral.screen.getWidth())
    canvas.setHeight(hardware.peripheral.screen.getHeight())
```

WHILE TRUE

```
    canvas.setXScroll(xposition)
    canvas.setYScroll(yposition)
```

END WHILE

HALT

Pseudocode – shadows

Due to the complexity of this function, I will include it later on when the program is able to generate levels as this way I will have a better understanding of how the objects will move around the player.

To view the algorithm, please click the link below.



Pseudocode – clearing all shadows

```
PROCEDURE clearShadows()
    shapes := canvas.getShapes()
    FOR each IN shapes THEN
        IF each.id.character(0) = "s" THEN
            canvas.removeShape(each)
        END IF
    NEXT each
END PROCEDURE
```

Pseudocode – level clearing

```
PROCEDURE clearLevelShapes()
    shapes := canvas.getShapes()
    FOR each IN shapes THEN
        canvas.removeShape(each)
    NEXT each
END PROCEDURE
```

Pseudocode – hashing

```
FUNCTION hash(user, pass):
    counter := 0
    hash_user := 0
    WHILE counter < user.length()
        hash_user += user.character(counter).to_ASCII()
    END WHILE
    counter := pass.length()
    hash_pass := 0
    WHILE counter >= 0
        hash_pass += pass.character(counter).to_ASCII()
    END WHILE
    output = modulo((hash_user ^ hash_pass), 1234567890)
    RETURN output
END FUNCTION
```

Pseudocode – persistent users

BEGIN

```
    saved := OPEN("users.table")
    saves := saved.READ()
    users_array := []
    FOR I := 0 TO I = saves.length()
        users_array.push(saves[I].id)
    NEXT I
```

HALT

Pseudocode – logging in

```
BEGIN
    username := INPUT(username)
    password := INPUT(password)
    hashed := hash(username, password)
    loggedInUser := null
    IF users_array.contains(hashed) THEN
        loadSave(hashed)
        PRINT("login completed")
    ELSE
        PRINT("login failed")
    END IF
HALT
```

Pseudocode – loading save

```
PROCEDURE loadSave(id)
    FOR I := 0 TO I = saves.length()
        IF saves[I].id = id THEN
            loggedInUser := saves[I]
            BREAK
        END IF
    NEXT I
END PROCEDURE
```

Pseudocode – registration

BEGIN

```
username := INPUT(username)
password := INPUT(password)
password2 := INPUT(password2)
screenname := INPUT(screenname)
hashed := hash(username, password)
IF users_array.contains(hashed) THEN
    PRINT("error, please use a different username and password combination")
ELSE IF password <> password2 THEN
    PRINT("passwords not matching")
ELSE IF check_alphanumeric(password) <> TRUE THEN
    PRINT("password must be alphanumeric")
ELSE IF check_alphanumeric(username) <> TRUE THEN
    PRINT("password must be alphanumeric")
ELSE IF check_alphanumeric(screenname) <> TRUE THEN
    PRINT("screenname must be alphanumeric")
ELSE IF password.length() < 6 THEN
    PRINT("password must be at least 6 characters long")
ELSE IF username.length() < 6 THEN
    PRINT("username must be at least 6 characters long")
ELSE IF screenname.length() < 3 THEN
    PRINT("screenname must be at least 3 characters long")
ELSE
    createSave(hashed, screenname)
    users_array.push(hashed)
    PRINT("new account made")
END IF
```

HALT

Pseudocode – input validation

```
FUNCTION check_alphanumeric(string)
    flag := TRUE
    FOR I=0 to I=string.length()
        char := string.character(I).to_ASCII() // gets ASCII character
        IF ((char > 47 AND char < 58) OR (char > 64 AND char < 91
        OR (char < 96 AND char > 123)) <> TRUE THEN
            flag := FALSE
            BREAK
        END IF
    NEXT I
    RETURN flag
END FUNCTION
```

Pseudocode – creating save

```
PROCEDURE createSave(id, name)
    user := Users.create()
    user.setDefault()
    user.setId(id)
    user.setName(name)
    saved.APPEND("users.table", user)
END PROCEDURE
```

Pseudocode – generate button

```
FUNCTION newButton(id, text, style)
    button := Buttons.create()
    button.setId(id)
    button.setText(text)
    button.setStyle(style)
    RETURN button
END FUNCTION
```

Pseudocode – main menu (example menu)

BEGIN

 btnLogout := newButton("0", "log out", smallButtonStyle)

 btnHelp := newButton("1", "help", largeButtonStyle)

 btnPlay := newButton("2", "play", largeButtonStyle)

 btnSettings := newButton("3", "settings", largeButtonStyle)

 btnAccount := newButton("4", "account", smallButtonStyle)

 btnShop := newButton("5", "shop", largeButtonStyle)

 WHILE TRUE

 FOR each IN Buttons

 IF Buttons[I].isClicked() = TRUE THEN // trigger

 PARSE("button" + Buttons[I].getText() + "click()")

 END IF

 NEXT each

 END WHILE

HALT

Pseudocode – settings (example button)

```
PROCEDURE buttonsettingsclick()
    FOR I := Buttons.length() TO I = 0
        Buttons[I].remove()
    NEXT I // I decrements each iteration
    btnBack := newButton("0", "back", smallButtonStyle)
    btnDisplay := newButton("0", "display", standardButtonStyle)
    // and so on
END PROCEDURE
```

Pseudocode – settings (example button)

```
PROCEDURE buttonlogoutclick()
    FOR I := Buttons.length() TO I = 0
        Buttons[I].remove()
    NEXT I // I decrements each iteration
    loggedInUser := null
    btnLogIn := newButton("0", "log in", smallButtonStyle)
    btnRegister := newButton("0", "register", smallButtonStyle)
    // and so on
END PROCEDURE
```

Pseudocode – saving times

```
BEGIN
    IF finalTime < getSaveTime(loggedInUser.id, selectedLevel) THEN
        newSaveTime(loggedInUser.id, selectedLevel, finalTime)
        PRINT("New record set")
    END IF
HALT
```

Pseudocode – getting a save's time

```
FUNCTION getSaveTime(id, level)
    timed := OPEN("times.table")
    times := timed.READ()
    FOR I := 0 TO I = times.length()
        IF times[I].id = id THEN // gets the correct user
            subTimes := times[I].times
            FOR J := 0 TO J = subTimes.length()
                IF subTimes[J].level = level THEN // gets the time
                    RETURN subTimes[J].time
                END IF
            NEXT J
        END IF
    NEXT I
    RETURN null
END FUNCTION
```

Pseudocode – saving a user's time

```
PROCEDURE newSaveTime(id, level, time)
    timed := OPEN("times.table")
    times := timed.READ()
    FOR I := 0 TO I = times.length()
        IF times[I].id = id THEN // gets the correct user
            subTimes := times[I].times
            FOR J := 0 TO J = subTimes.length()
                IF subTimes[J].level = level THEN // gets the time
                    subTimes[J].time := time
                    BREAK
                END IF
            NEXT J
        END IF
    NEXT I
    timed.CLEAR("timer.table")
    timed.APPEND("timer.table", times)
END PROCEDURE
```

Pseudocode – displaying records

BEGIN

```
    text := getSaveTime(loggedInUser.id, selectedLevel)
    text2 := getRecordTime(selectedLevel)
    PRINT("own record: " + text)
    PRINT("overall record: " + text2.time + " (" + text2.user + ")")
```

HALT

Pseudocode – getting the record time

```
FUNCTION getRecordTime(level)
    timed := OPEN("times.table")
    times := timed.READ()
    min := null
    FOR I := 0 TO I = times.length()
        IF (times[I].times[level] < min.time OR min = null) AND
            times[I].times[level] <> 0 THEN
            min.time := times[I].times[level]
            min.user := times[I].user
        END IF
    NEXT I
    RETURN min
END FUNCTION
```

Pseudocode – shop transactions

```
PROCEDURE shopTransaction(item)
    IF loggedInUser.money > item.getPrice() THEN
        updateSave(loggedInUser.id, "upgrades." + item.getId(),
        TRUE)
        updateSave(loggedInUser.id, loggedInUser.money,
        (loggedInUser.money - item.getPrice()))
        PRINT("transaction complete")
        PARSE("upgrade" + item.getId() + "bought()")
        item.setBought(TRUE)
    ELSE
        PRINT("not enough money")
    END IF
HALT
```

Pseudocode – updating save

```
PROCEDURE updateSave(id, attribute, value)
    FOR I := 0 TO I = saves.length()
        IF saves[I].id = id THEN
            PARSE("saves[I]." + attribute + " := " + value)
            BREAK
        END IF
    NEXT I
    saved.CLEAR("user.table")
    saved.APPEND("user.table", saves)
END PROCEDURE
```

Pseudocode – speed (example purchase)

```
PROCEDURE upgradespeedbought()
    speedMultiplier := 1.5
END PROCEDURE
```

Variable/list plan

Type // Hungarian	Scope	Name	Description	Example data
Pointer // p	Global	pPlayer	A quick method of interacting with the player entity	n/a
Array // rg	Global	rgUser	The logged in user's details	[“UserID”, “name”, “admin”]
Array // rg	Global	rgUsers_list	All possible user's details	[["UserID1", "name", "admin"], ["UserID2", "name", "admin"]]
Array // rg	Global	rgLevels_list	All premade level's details	[["LevelID1", "name", "description"], ["LevelID2", "name", "description"]]
Integer // n	Global	nLevel	The currently selected level's position in the Levels_list array	2

Variable/list plan

Type // Hungarian	Scope	Name	Description	Example data
Array // rg	Global	rgLevel_objects	Shapes to be generated for a given level	[["shape1", "xposition", "yposition", "type", "style"], ["shape2", "xposition", "yposition", "type", "style"]]
Integer // n	Local	nCounter	A counter used in any loop	4
String // dw	Global	dwSelected_purchase	The name of the currently selected purchasable item	"Upgrade1"
Float // f	Global	fXposition	Current player position on x axis	437.31

Variable/list plan

Type // Hungarian	Scope	Name	Description	Example data
Float // f	Global	fYposition	Current player position on y axis	-543.1
Float // f	Global	fXvelocity	Current player velocity along x axis	10.324
Float // f	Global	fYvelocity	Current player velocity along y axis	-0.163
Array // rg	Global	rgRecord_times	Best times to complete a level across all users	["0.321", "21.32", "43.4321"]
Array // rg	Global	rgUser_times	The logged in user's best times	["0.623", "24.1", "43.4321"]
Float // f	Global	fCurrent_time	The logged in user's current time in a level	37.3215

Variable/list plan

Type // Hungarian	Scope	Name	Description	Example data
String // dw	Local	dwHashed	The result of a password hash	"Gt64w7gh7wF89hHgq78hGqE9ht439pL"
String // dw	Local	dwUsername	The user's username	"Charlie001"
String // dw	Local	dwPassword	The user's password	"1a2b3c4d5e"
Array // rg	Local	rgPoly_Points	The coordinates of the vertices of a polygon	[“13.3,3.43”, “14.5,1.2”, “11,12.3”]

Multi user plan

- This program will allow for multiple separate users. Each user can have their own data saved for settings, times taken, user information and so on.
- Each user can know their own high scores, and can view the “global” high score between all users (the best of the best).
- Each user will have a unique id which will be generated based off their username and password (via hashing).
- There will be the option to make a user an administrator so that they can reset other user’s passwords, and delete other accounts.

Gameplay

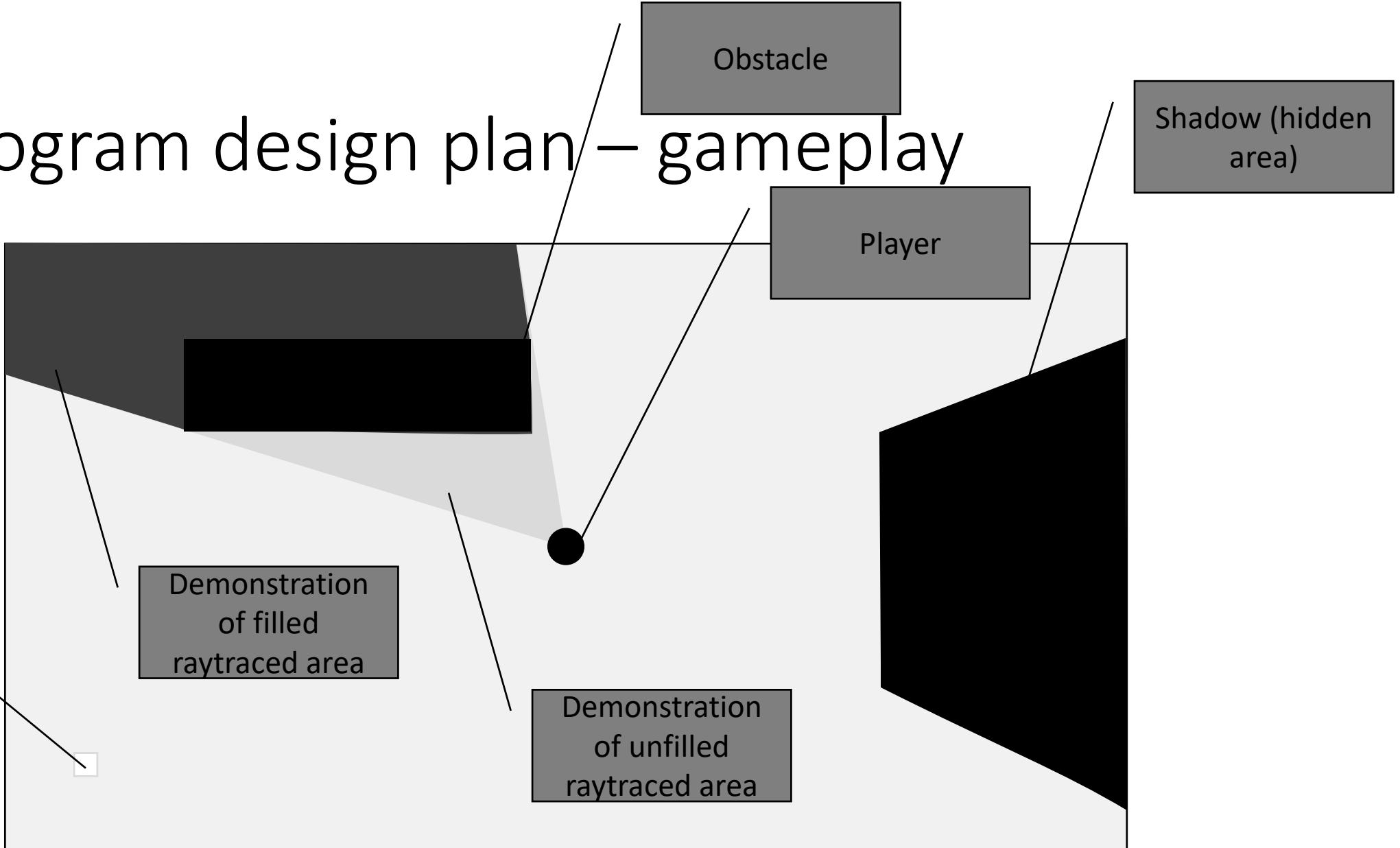
- There will be individual, pre-designed levels that can be played. These levels will become progressively harder as the user completes previous levels.
- The basis of each level will be a maze. As a result there should be dead ends and only 1 exit point for each level.
- Some of the levels can have parts that can damage the player. If the player takes too much damage, they will fail the level and have to restart.
- Things that will damage the player are specific walls/obstacles, and enemy AIs (they will track the player).
- The scenery should scroll as the player moves.
- The player should not be able to see behind obstacles/walls.

Gameplay

- There should be collectibles to promote exploration in the levels.
- The player should be able to pause the level at any time.
- The player should be able to reattempt levels to get a better time.

Initial program design plan – gameplay

Background: light grey
Player: black fill
Obstacle: black fill
Shadow: black fill
Collectible: white fill, light grey border



This is an idea of how part of a level may appear with further construction lines

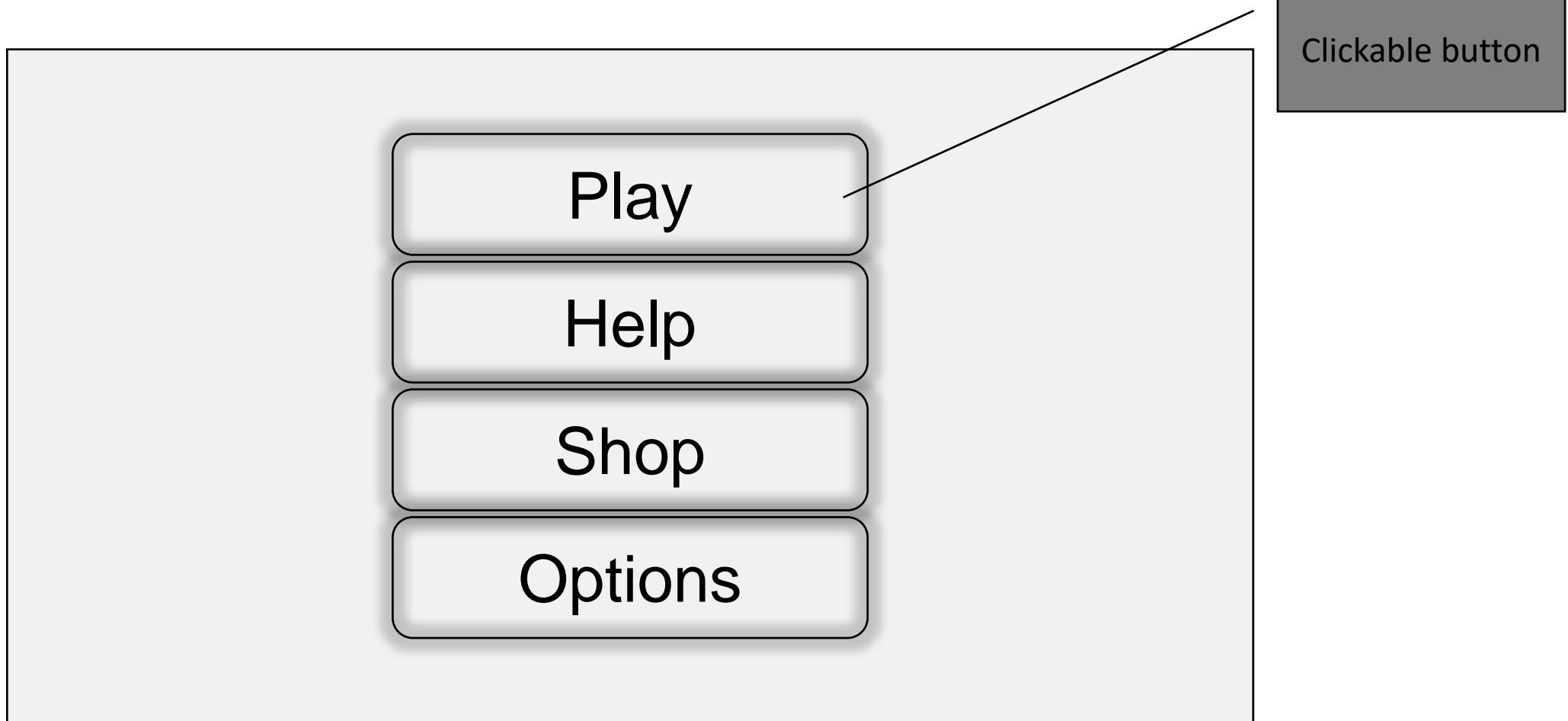
Initial program design plan – gameplay

- The colour scheme has been somewhat predetermined by CDS stating that it should be mostly black and white, and I have developed this to include the full greyscale spectrum, meaning that I have a more reasonable template to work from.
- I have chosen to make the player and shadows black as shadows are meant to hide what cannot be seen, and is logical and matches the real world. The player is black as it can collide with the obstacles that are covered by black shadows, so collision obstacles match.
- The background is light grey so as to prevent strain on eyes, as solid white backgrounds can become an irritant after a while.
- The collectible is white to stand out from the background and appear different from the obstacles and player.

User interface

- The UI style should be black and white or greyscale
- Buttons should be large and clear
- Backgrounds will be plain colour or gradient
- The style should be minimalistic for ease of use
- Drop shadows can be used for decoration
- Lethal objects can include red in their design
- The exit can be a different colour (e.g. green)

Initial program design plan – main menu UI



Initial program design plan – main menu UI

Font style: Arial (easy to read, common across all systems)

Font colour: Black (easy to read)

Font size: approximately 1/10th of the screen height (fills buttons, easy to read)

Button border: Black solid thin line (clear distinction between background and button)

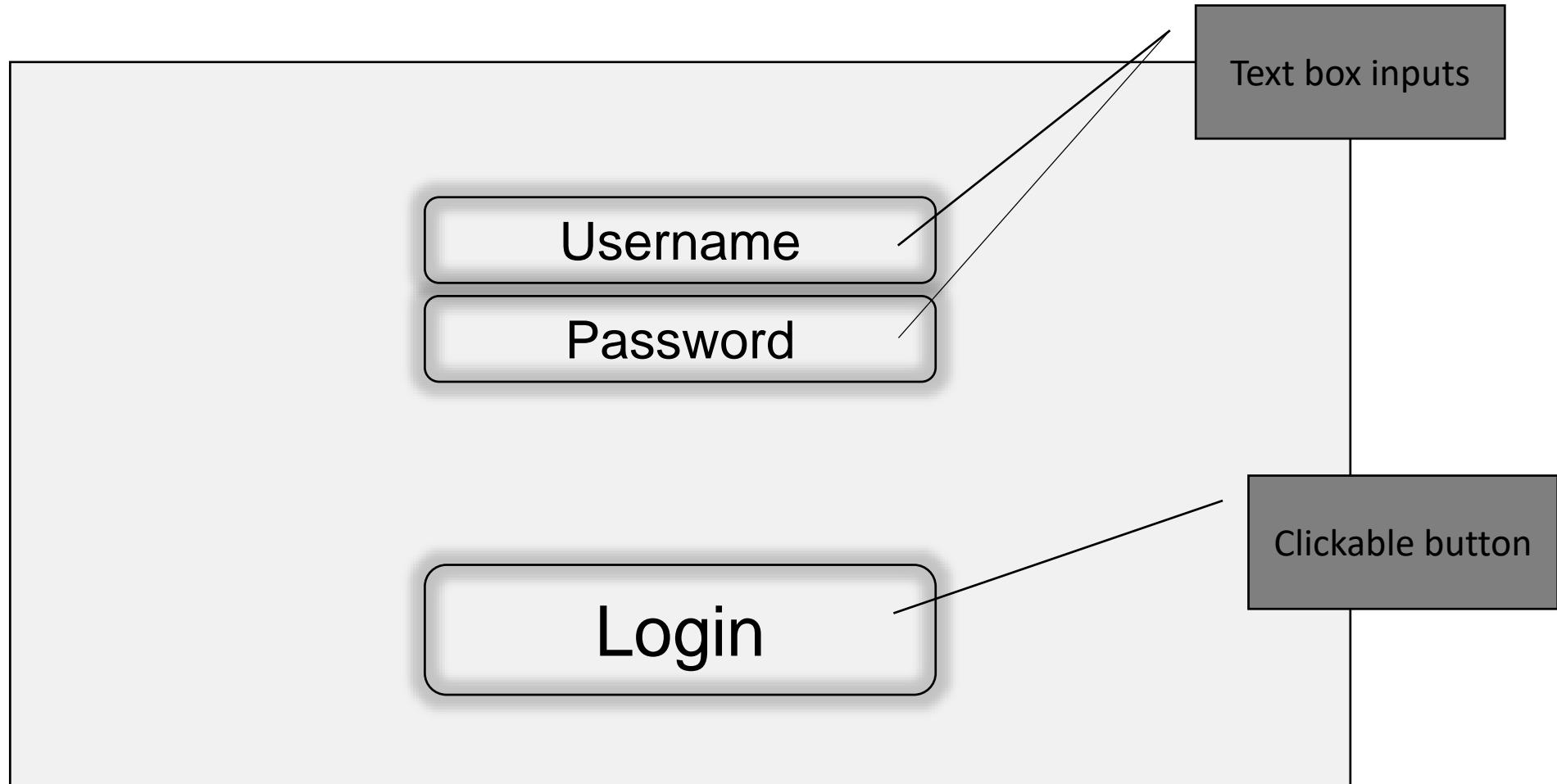
Button fill: clear (background unnecessary as button has a border)

Button drop-shadow: dark grey, gentle fading gradient, both inner and outer (emphasises that the element is not the background and makes it more distinctive)

Button size: approx. 1/5th screen height, 1/3rd screen width (fills the screen sufficiently without making it cluttered)

Widgets: play button, help button, options button

Initial program design plan – login screen UI



Initial program design plan – level selector UI

Font style: Arial (easy to read, common across all systems)

Font colour: Black (easy to read)

Font size: approximately 1/10th of the screen height [login], 1/16th screen height [text boxes] (fills buttons, easy to read)

Button border: Black solid thin line (clear distinction between background and button)

Button fill: clear (background unnecessary as button has a border)

Button drop-shadow: dark grey, gentle fading gradient, both inner and outer (emphasises that the element is not the background and makes it more distinctive)

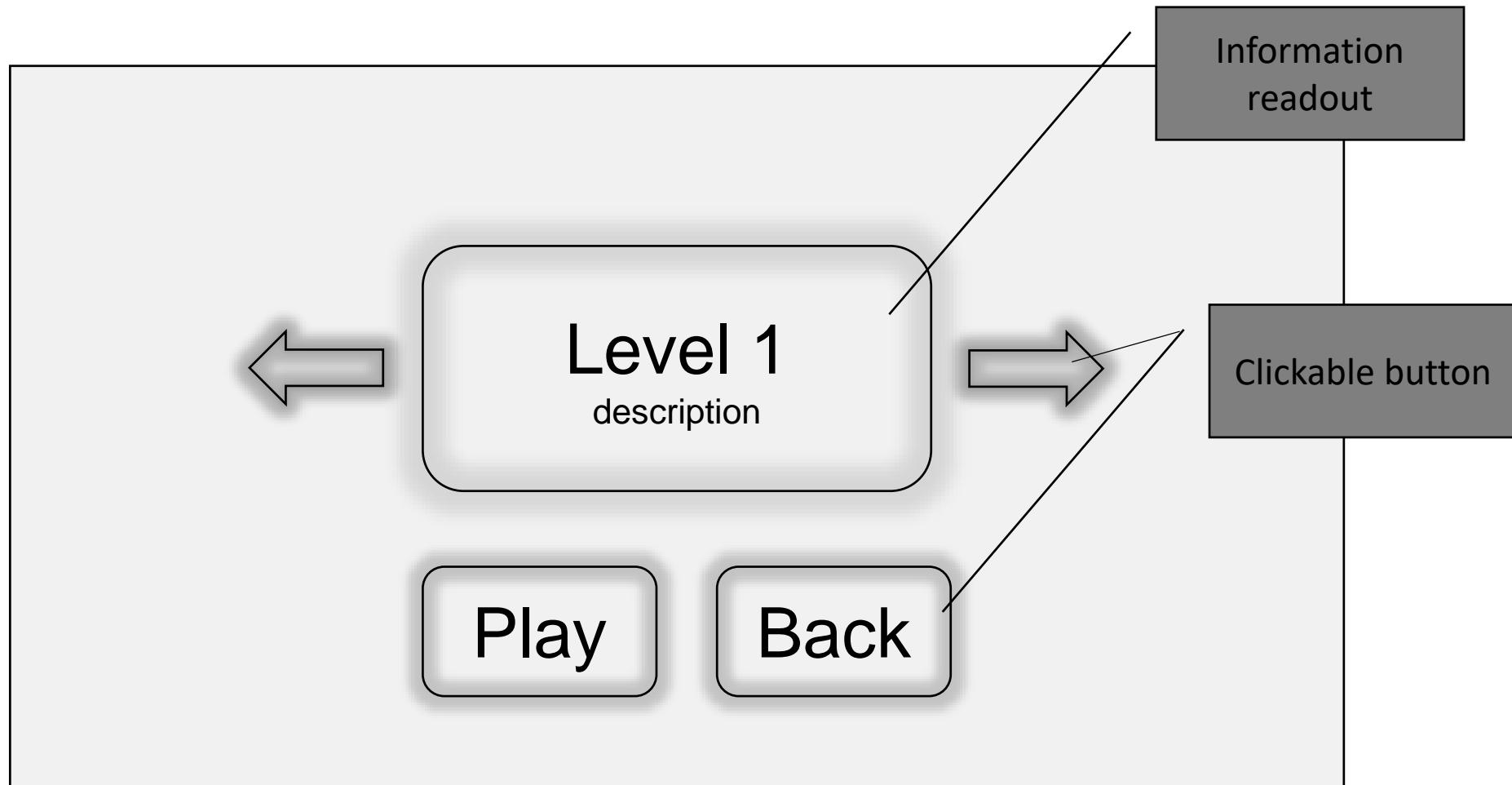
Button size: approx. 1/5th screen height, 1/3rd screen width (fills the screen sufficiently without making it cluttered)

Text input box: ditto button, 1/10th screen height (highly visible, follows same style choices as other elements)

Widgets: login button, username text field, password text field

The password field should show typed characters as '*'s or similar to ensure privacy. This is a common method of keeping passwords more secure.

Initial program design plan – level selector UI



Initial program design plan – level selector UI

Font style: Arial (easy to read, common across all systems)

Font colour: Black (easy to read)

Font size: approximately 1/10th of the screen height [buttons, title], 1/16th screen height [description] (fills buttons, easy to read, has sensible padding for description)

Button border: Black solid thin line (clear distinction between background and button)

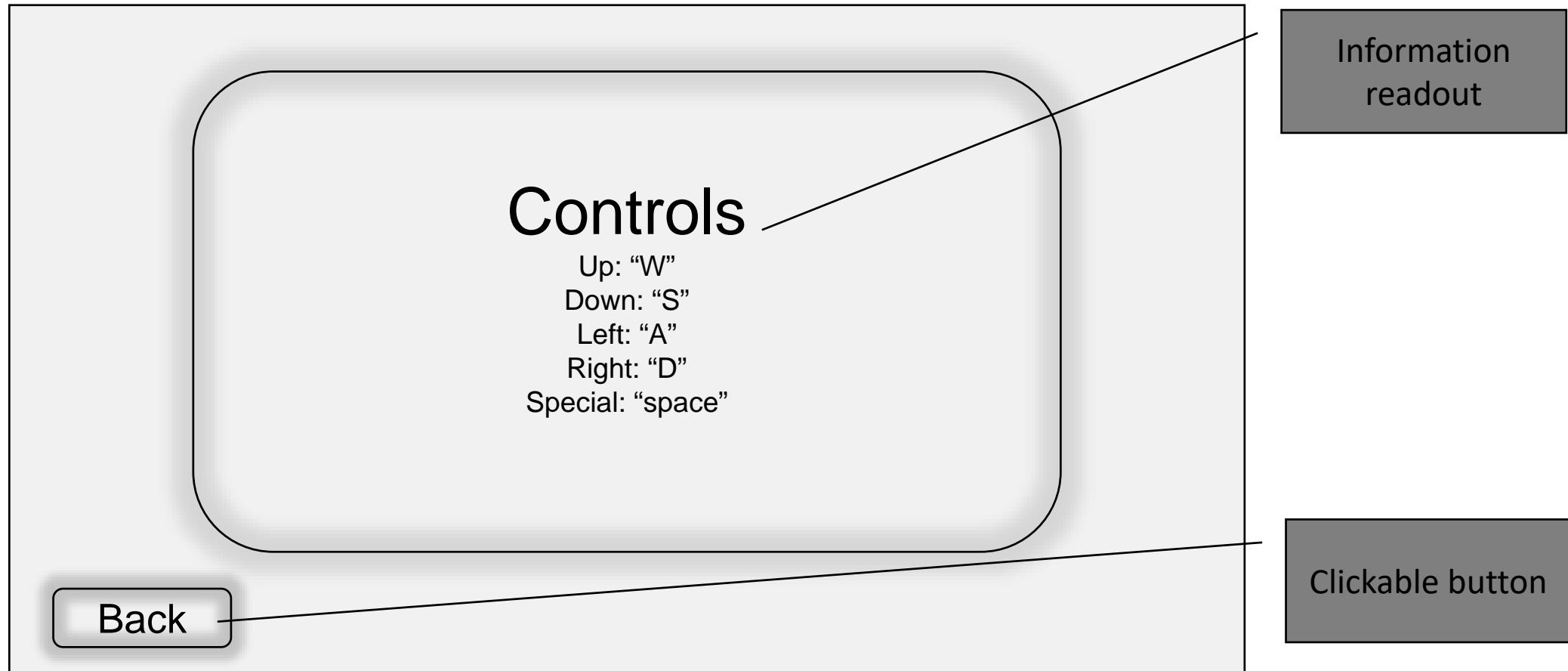
Button fill: clear (background unnecessary as button has a border)

Button drop-shadow: dark grey, gentle fading gradient, both inner and outer (emphasises that the element is not the background and makes it more distinctive)

Button size: approx. 1/5th screen height [bevelled rectangle] , 1/16th height [arrows], 1/8th screen width (fills the screen sufficiently without making it cluttered)

Widgets: left arrow button, right arrow button, play button, back button, title text, description text, level container

Initial program design plan – help screen UI



Initial program design plan – help screen UI

Font style: Arial (easy to read, common across all systems)

Font colour: Black (easy to read)

Font size: approximately 1/10th of the screen height [title], 1/16th screen height [button], 1/24th screen height [description] (fills buttons, easy to read, good padding and spacing in description)

Button border: Black solid thin line (clear distinction between background and button)

Button fill: clear (background unnecessary as button has a border)

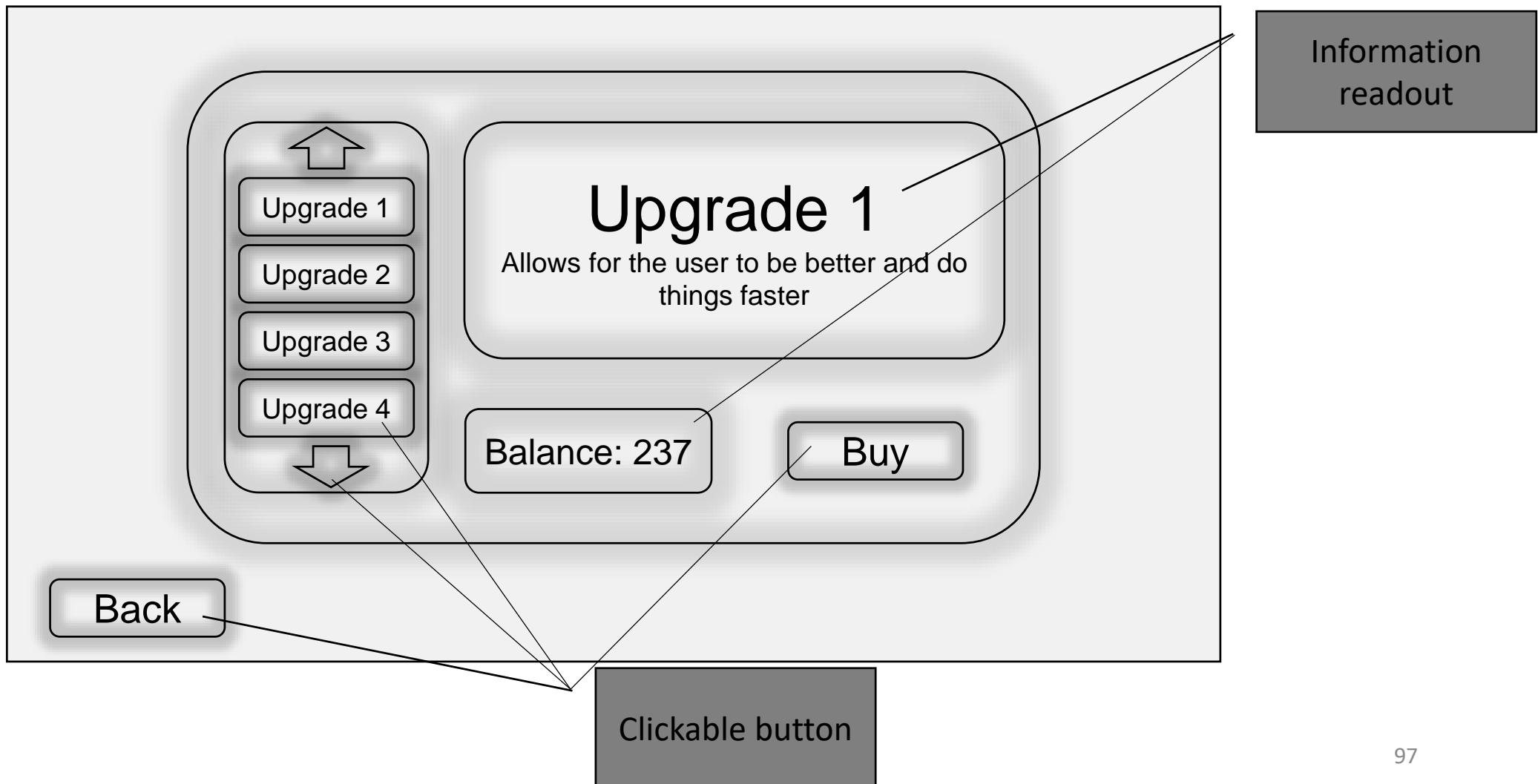
Button drop-shadow: dark grey, gentle fading gradient, both inner and outer (emphasises that the element is not the background and makes it more distinctive)

Button size: approx. 1/12th screen height, 1/12th screen width (fills the screen sufficiently without making it cluttered)

Text readout: ditto button, 80% screen height, 80% screen width (highly visible, follows same style choices as other elements)

Widgets: back button, title text, description text, help container

Initial program design plan – shop UI



Initial program design plan – shop UI

Font style: Arial (easy to read, common across all systems)

Font colour: Black (easy to read)

Font size: approximately 1/10th of the screen height [title], 1/16th screen height [back, buy button], 1/24th screen height [description, upgrades list], 1/20th screen height [balance] (fills buttons, easy to read, good padding and spacing in description)

Button border: Black solid thin line (clear distinction between background and button)

Button fill: clear (background unnecessary as button has a border)

Button drop-shadow: dark grey, gentle fading gradient, both inner and outer (emphasises that the element is not the background and makes it more distinctive)

Button size: approx. 1/12th screen height [buy, back, upgrades list], 1/18th screen height [arrows], 1/12th screen width [buy, back, upgrades list], 1/40th screen width [arrows] (fills the screen sufficiently without making it cluttered)

Text readout: ditto button, 50% screen height [details], 1/10th screen height [balance], 50% screen width [details], 1/8th screen width [balance] (highly visible, follows same style choices as other elements)

Widgets: back button, buy button, upgrade buttons, up arrow button, down arrow button, title text, description text, balance text, shop container, balance container, list container, description container

Initial program design – UI widgets

Widget type // Hungarian	Widget name	Widget description	Where used
Button // btn	btnLevel_select_goto	Used to show the level selection screen when clicked	Main menu
Button // btn	btnHelp_goto	When clicked, will show the help screen	Main menu
Button // btn	btnOptions_goto	When clicked, will show the options screen	Main menu
Button // btn	btnLogin_try	When clicked, hashes the user's password and attempts to log them in	Login screen
Text input // txt	txtLogin_username	Allows the user to enter their username	Login screen
Text input // txt	txtLogin_password	Allows the user to enter their password	Login screen

Initial program design – UI widgets

Widget type // Hungarian	Widget name	Widget description	Where used
Button // btn	btnLevel_left	A button shaped like an arrow that allows the user to change the selected level	Level selection screen
Button // btn	btnLevel_right	A button shaped like an arrow that allows the user to change the selected level	Level selection screen
Button // btn	btnPlay_level	Allows the user to play the selected level	Level selection screen
Button // btn	btnLevel_menu_goto	Goes back to the main menu when clicked	Level selection screen
Text label // lbl	lblLevel_title	A readout for the current level's title	Level selection screen

Initial program design – UI widgets

Widget type // Hungarian	Widget name	Widget description	Where used
Text label // lbl	lblLevel_description	A readout for the current level's description	Level selection screen
Container box // ctr	ctrLevel_container	The border surrounding the level title/description	Level selection screen
Button // btn	btnHelp_menu_goto	Goes back to the main menu when clicked	Help screen
Text label // lbl	lblHelp_title	Title text readout	Help screen
Text label // lbl	lblHelp_description	The information given to the user	Help screen
Container box // ctr	ctrHelp_container	Surrounding border for the text shown	Help screen
Button // btn	btnShop_goto	Shows the shop screen	Main menu
Button // btn	btnShop_menu_goto	Returns to main menu	Shop screen

Initial program design – UI widgets

Widget type // Hungarian	Widget name	Widget description	Where used
Button // btn	btnShop_menu_goto	Returns to main menu	Shop screen
Button // btn	btnShop_buy	Attempts to purchase the selected upgrade	Shop screen
Button // btn	btnShop_up	A button shaped like an arrow to let the user scroll through a list	Shop screen
Button // btn	btnShop_down	A button shaped like an arrow to let the user scroll through a list	Shop screen
Button // btn	btnShop_U1	The first upgrade option in the shop	Shop screen
Button // btn	btnShop_U2	The second upgrade option in the shop	Shop screen

Initial program design – UI widgets

Widget type // Hungarian	Widget name	Widget description	Where used
Button // btn	btnShop_U3	The third upgrade option in the shop	Shop screen
Button // btn	btnShop_U4	The fourth upgrade option in the shop	Shop screen
Container box // ctr	ctrShop_U_container	The border surrounding the upgrade buttons	Shop screen
Text label // lbl	lblShop_title	A readout for the selected upgrade's title	Shop screen
Text label // lbl	lblShop_description	A readout for the selected upgrade's description	Shop screen
Container box // ctr	ctrShop_US_container	The border surrounding the upgrade's information	Shop screen
Text label // lbl	lblShop_balance	A readout for the user's balance	Shop screen

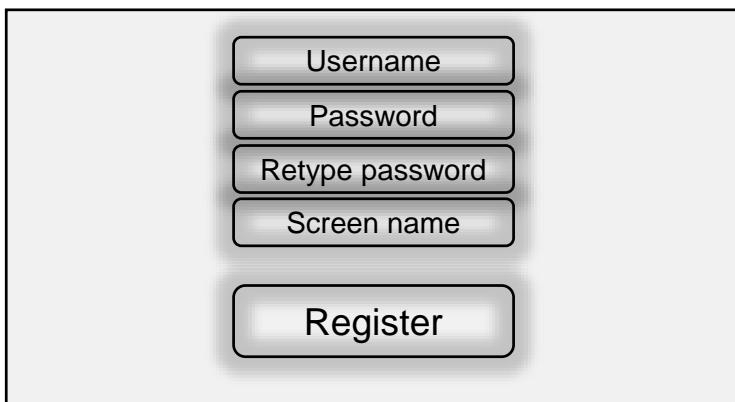
Initial program design – UI widgets

Widget type // Hungarian	Widget name	Widget description	Where used
Container box // ctr	ctrShop_balance_container	The border surrounding the user's balance	Shop screen
Container box // ctr	ctrShop_container	The border surrounding the shop screen	Shop screen

UI plan approval

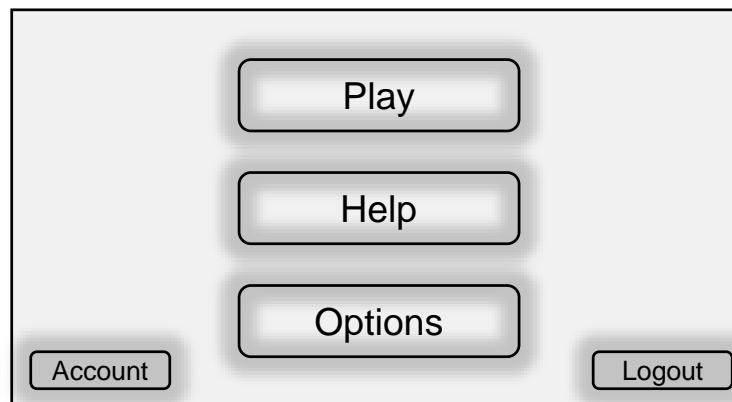
The GUI plan has been submitted to CDS and has been approved with additional comments that it should be expanded to show a registration screen, and should have a logout and account controls button in the main menu.

The new designs have been shown below, and I have added a new account control screen (right) that follows the previous designs



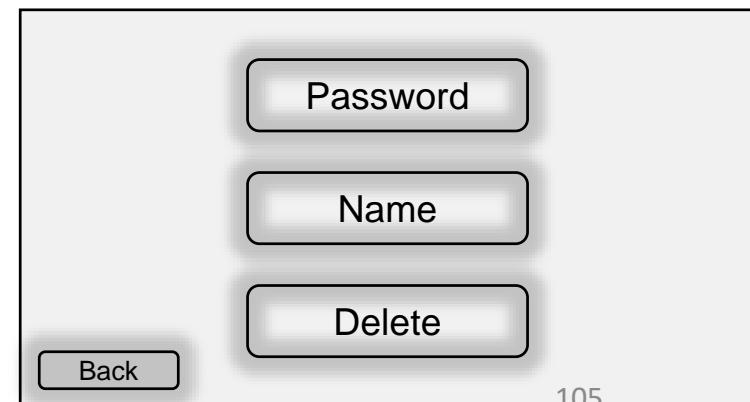
This image shows a registration form with four input fields: 'Username', 'Password', 'Retype password', and 'Screen name'. Below these is a large 'Register' button.

Username
Password
Retype password
Screen name
Register



This image shows a main menu with three primary options: 'Play', 'Help', and 'Options'. At the bottom left is an 'Account' button, and at the bottom right is a 'Logout' button.

Play
Help
Options
Account
Logout



This image shows an account control screen with four buttons: 'Password', 'Name', 'Delete', and 'Back'. The 'Back' button is located at the bottom left.

Password
Name
Delete
Back

SQL Tables

A requirement of CDS is that the program uses SQL tables as the main method of data persistence. Before I move on to development of the project, I need to decide what tables will be needed, and what columns will be in those tables.

The database needs to be in 3NF to ensure ACID rules are applied, as data loss, corruption and redundancy cannot occur.

SQL Tables

Table	Purpose	Columns
Levels	Stores each level and the meta data associated with it.	id, name, description, xspawn, yspawn, xexit, yexit
Rectangles	Contains each rectangle element that is generated per level	levelid, id, x, y, width, height, class
Circles	Contains each circle element generated per level	levelid, id, x, y, radius, class
Polygons	Contains each polygon element generated per level	levelid, id, points, class
Savefiles	Records and saves user's data	id, name, admin
Saveprogress	Saves a user's data for their best time in a level, and the collectibles found	playerid, level1time, level1collectibles {there will be a time and collectibles for each level}

SQL Tables

Table	Purpose	Columns
Settings	Saves a user's settings so that they are persistent for each user	playerid, up, down, left, right, special, volume {if additional settings are added, this will change}
Purchases	Saves which upgrades have been purchased from the in-game shop	playerid, upgrade1, upgrade2 {if additional upgrades are added, this will change}
Achievements	Saves which achievements each user has unlocked	playerid, achievement1, achievement2 {there will be a column per achievement}

PK indicates Primary Key
FK indicated Foreign Key

SQL Tables

Rectangles Table (including example data):

LEVEL (FK, text)	ID (PK, text)	X (num)	Y (num)	WIDTH (num)	HEIGHT (num)	CLASS (text)
I1	I1enclosure	0	0	15360	8640	obstacle_box
I1	I13	7053	2764	594	294	obstacle_box

Levels Table (including example data):

ID (PK, text)	NAME (text)	DESCRIPTION (text)	XSPAWN (num)	YSPAWN (num)	XEXIT (num)	YEXIT (num)
I1	Beginner	Teaching the basics of the game	0	0	15160	8440
I2	Maze	Hidden routes	15	10	9800	9800

PK indicates Primary Key
FK indicated Foreign Key

SQL Tables

Circles Table (including example data):

LEVEL (FK, text)	ID (PK, text)	X (num)	Y (num)	RADIUS (num)	CLASS (text)
I1	I18	-5	89	543	obstacle_box
I1	I19	646	64	12	obstacle_box

Polygons Table (including example data):

LEVEL (FK, text)	ID (PK, text)	POINTS (text)	CLASS (text)
I2	I24	[123,231],[321,232],[321,7534]	obstacle_box
I8	I82	[12,5.2],[1,642],[-6,23],[32,32],[0,2]	obstacle_box

SQL Tables

Savefiles Table (including example data):

PlayerID (PK, num)	playerName (text)	isAdmin (boolean)
4132795412	ADMIN	true
58320578491	Adam	false

Saveprogress Table (including example data):

pID (FK, num)	L0items (num)	L0time (num)	L1items (num)	L1time (num)
4132795412	12	17.2432	62	32.6430
58320578491	7	19.2141	67	31.6527

SQL Tables

Savesettings Table (including example data):

PlayerID (FK, num)	Key0 (num)	Key1 (num)	Key2 (num)	Key3 (num)
4132795412	87	83	65	68
58320578491	87	83	72	75

Saveupgrades Table (including example data):

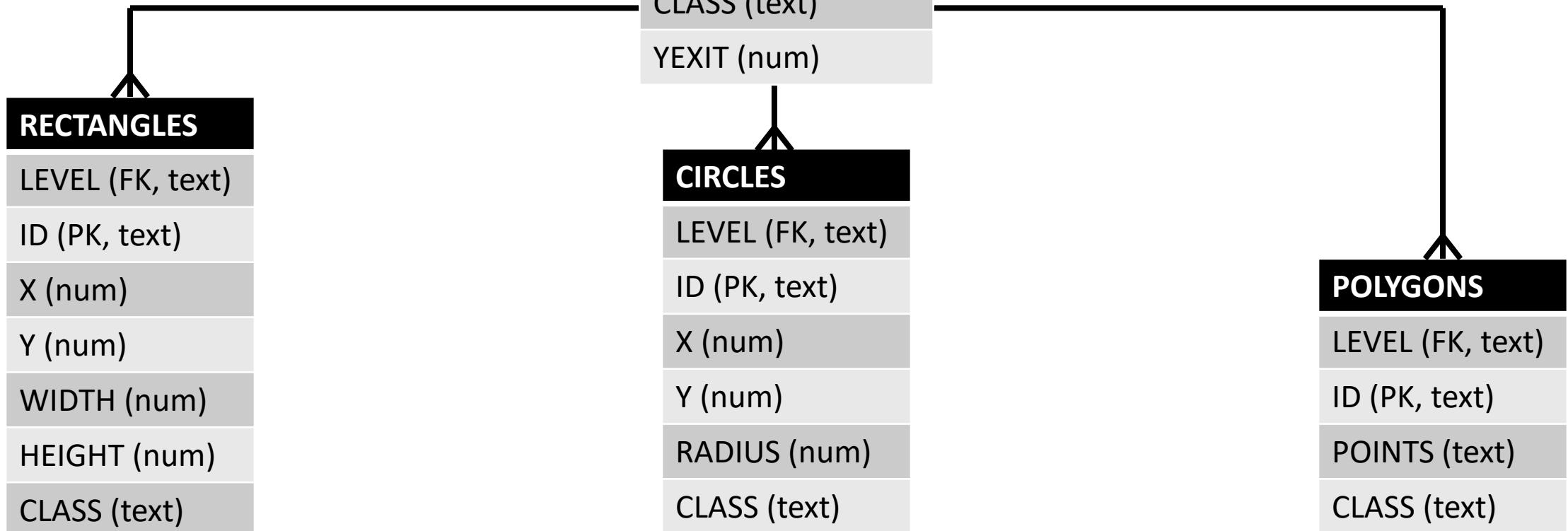
PlayerID (FK, num)	Item1 (boolean)	Item2 (boolean)	Item3 (boolean)	Item4 (boolean)
4132795412	True	False	False	True
58320578491	True	True	False	true

SQL Tables

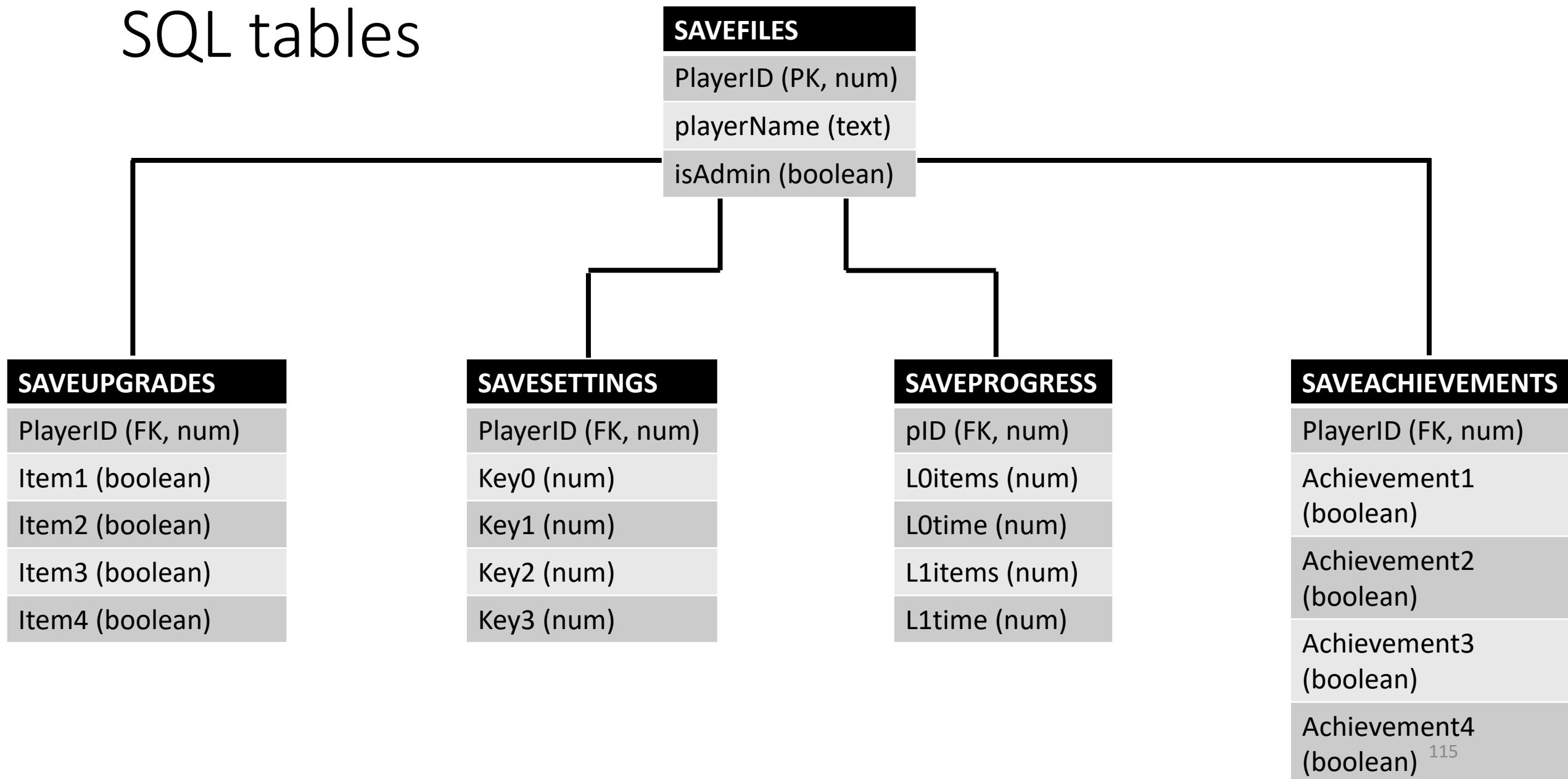
Saveachievements Table (including example data):

PlayerID (FK, num)	Achievement1 (boolean)	Achievement2 (boolean)	Achievement3 (boolean)	Achievement4 (boolean)
4132795412	True	False	False	False
58320578491	False	True	True	False

SQL Tables



SQL tables



Test plan

- The test plan will follow the same structure as my objectives, but will be split into the categories of expected data, extreme data and erroneous data.
- Each module will also be tested as it is implemented during development, and then retested at the end.
- The test plan can be seen in the “Test plan” tab/spreadsheet of the Gantt Chart. Example data is shown below.

Task ID	Main Task	Intermediate Task	Sub Task	Test data	Test type		
				Normal	Extreme	Erroneous	
1	Game Immersion		play game	get non-developers to play game	n/a		integration
1.a	Fullscreen	Detect page size Detect screen size Show message if page size is not screen size	enter/exit fullscreen mode	spam fullscreen toggle	n/a		unit
1.a.i			windowed browser	window crosses over 2 monitors	n/a		integration
1.a.ii			single monitor	2 monitors	n/a		integration
1.a.iii			windowed browser	2 monitors	n/a		integration
1.a.iv			enter fullscreen	enter fullscreen inside a windowed virtual machine	make window exactly correct size so that the page is the full size of the screen		integration
1.a.v			exit fullscreen	change tab/window focus	make window exactly correct size so that the page is the full size of the screen		regression
1.b	Sound effects		cause audio to play	repeated manual triggers	n/a		integration
1.b.i		Play audio files Event based triggers	manual audio play	repeatedly play same audio while still playing	n/a		integration
1.b.ii			trigger audio event	repeatedly trigger same audio event while previous audio is still playing	n/a		integration
1.c	Ambience effects	Play audio files Loop audio files Fade in audio Fade out audio Layer audio files Randomise audio selection	cause audio to play	repeated manual triggers	n/a		integration
1.c.i			manual audio play	repeatedly play same audio while still playing	n/a		integration
1.c.ii			wait for track to finish	force track to finish (seek/track)	n/a		integration
1.c.iii			manually trigger audio	multiple audios playing at once	play and then immediately mute		integration
1.c.iv			wait for track to finish	force track to finish (seek/track)	track to beyond finish time		integration
1.c.v			trigger multiple audios at once	n/a	n/a		integration
1.c.vi			run randomiser script	multithread randomiser script	n/a		integration

Development

Developmental testing

- All modules will be tested with normal data before being added to the main program. Some may have extreme and erroneous testing as well, but this will be made clear when used.
- GUI systems will typically be designed in a blank document and then cut and pasted into the main project, where additional functionality will be introduced.
- Scripts may be created in an external document and then imported, or created in the main project if they rely heavily on pre-existing features.
- The test results can be seen in the “Test results” tab/spreadsheet of the Gantt Chart. Example data is shown on the page below.

Developmental testing

Task ID	Main Task	Intermediate Task	Sub Task	Test result		
				Success	Evidence	Comment
1	Game Immersion	Fullscreen		full	23.3	
1.a			Detect page size	full	18	
1.a.i			Detect screen size	full	1	
1.a.ii			Show message if page size is not screen size	full	1	
1.a.iii			Hide message if page size is screen size	full	1	
1.a.iv		Sound effects	Pause game if page size is not screen size	full	18	
1.a.v				full	20.1	
1.b			Play audio files	full	20.1	
1.b.i			Event based triggers	full	20.1	
1.b.ii				full	20.1	
1.c		Ambience effects	Play audio files	full	20.2	
1.c.i			Loop audio files	full	20.2	
1.c.ii			Fade in audio	full	20.2	
1.c.iii			Fade out audio	full	20.2	
1.c.iv			Layer audio files	full	20.2	
1.c.v			Randomise audio selection	full	20.2	
10.f	Primary admin account	Primary admin account		full	9.2	
10.f.i			Generate a primary admin account with default username and password (most likely "administrationacc" and "epsilondeltaomega")	full	9.2	
10.f.ii			Ensure primary admin account has the admin record set to TRUE	full	9.2	
11	User login	User input		full	25	
11.a				none	9.1	the user could not type any characters, which was quickly rectified
11.a.i			Username field	full	9.1	
11.a.ii			Password field	full	9.1	
11.a.iii			Go to register screen button	full	9.1	
11.a.iv			Login button	full	9.1	
11.b		Password hashing		partial	8	the order of characters initially didn't have an effect on the hash, so it has been changed to influence the value
11.b.i			Split username into individual characters	full	8	
11.b.ii			Get ASCII value of each character	full	8	
11.b.iii			Check ASCII value is an accepted character (a-z, A-Z, 0-9)	full	8	
11.b.iv			If not accepted, report there is an error	full	8	
11.b.v			If accepted, increment a value by the ASCII value cubed	full	8	
11.b.vi			Repeat for next character	full	8	
11.b.vii			Repeat above for password, but square instead of cube	full	8	
11.b.viii			Perform a modulo-power operation of username breakdown to the power of password breakdown with modulo from an input parameter	full	8	

IDE choice

- I have decided to develop the project in an IDE called Brackets, developed by Adobe.
- Brackets is aimed for web design, which means that it is optimised for JavaScript, HTML and CSS which I will be using.
- Brackets has built in debugging tools for the console but I prefer to use Google Chrome's built in developer tools for adding breakpoints, stepping through code and pausing execution.
- The IDE also includes functionality such as autocomplete, auto-close-brackets, split screen viewing, “quick edit” (quickly change/make CSS rules and HTML elements, set colours etc.), live preview (displays the project in your browser without needing to save changes as you type more code) and other options too.

An example of using the IDE.

IDE choice



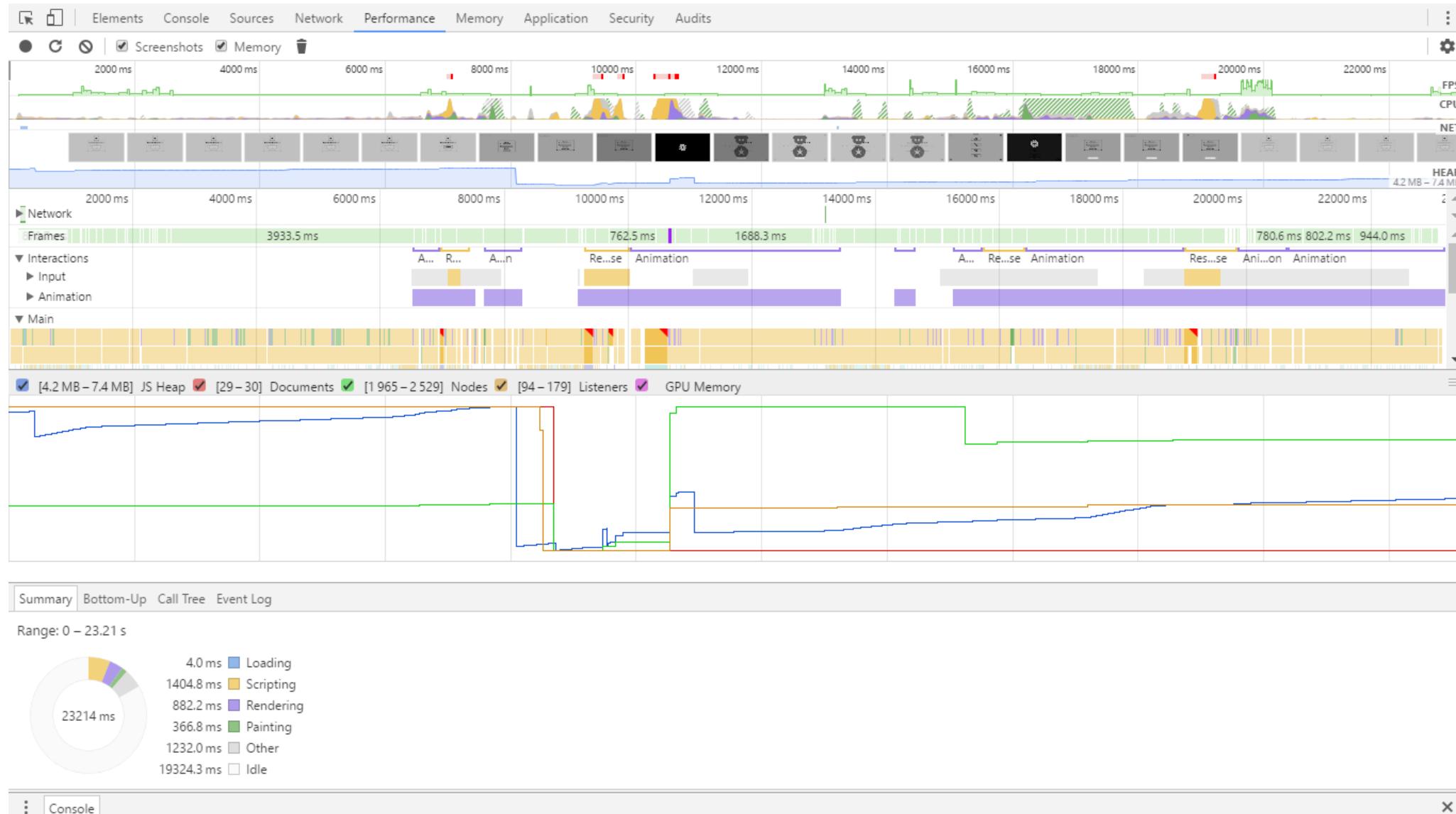
An example of debugging the program.

IDE choice



IDE choice

Using the debugger to optimise the program (finding lag spikes and recording listeners to determine what parts of the program are most CPU intensive).



Setting up the HTML

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="UTF-8">
    <title>
      Computing coursework
    </title>
    <style>
    </style>
  </head>
  <body>
    <svg id="canvas" viewBox="0 0 1920 1080">
      <defs id="filters">
      </defs>
    </svg>
    <script>
    </script>
  </body>
</html>
```

Allows for full UNICODE character set support

Tab title

Empty style sheet

The canvas element which the game will be displayed with dimensions 1920x1080 pixels, and the upper left view is (0,0)

Sub section of the canvas used for filters (blur etc.)

Empty script

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="UTF-8">
    <title>
      Computing coursework
    </title>
    <style>
      html, body {
        font-family: Calibri;
        margin: 0px;
        width: 100%;
        height: 100%;
        text-align: center;
      }
      svg, #canvas {
        height: 100%;
        width: 100%;
        padding: 0;
        background-color: #EEE;
        position: fixed;
        left: 0px;
        top: 0px;
        border: 0px;
      }
      #overscroll {
        overflow: hidden;
      }
    </style>
  </head>
  <body>
    <div id="overscroll">
    </div>
    <svg id="canvas" viewBox="0 0 1920 1080">
      <defs id="filters">
      </defs>
    </svg>
    <script>
    </script>
  </body>
</html>
```

Setting up the HTML

Sets up the whole document as filling the browser tab and sets the default font to Calibri, with text being centred around the middle of an element

Sets up the SVG canvas to fill the browser tab which cannot be scrolled and has a nearly white background colour

Sets up the div element with id “overscroll” to ensure that there is no scrollbar visible in the window

A div element with id “overscroll” that will be used to make sure that there are no scrollbars visible

Setting up the HTML

It is worth pointing out that CSS is, by definition causes inheritance of style rules. A parent HTML element will cause all of its children to have the same rules, unless the CSS states otherwise.

CSS also allows for multiple inheritance as a HTML element may inherit from a parent element, its classes and its ID. This means that I will need to ensure that each element is styled correctly, as it could be affected by other elements.

An example of inheritance is that the all elements will inherit the “font-family: Calibri” from the body.

Forcing the user to run in fullscreen

After some consideration, I decided that I would force the user to run the game in a fullscreen window. This is because the game will be more immersive as “distractions” from other programs will not be visible. It also means that I will not have to waste processing power on checking if the window for the browser has changed size and so have to adapt to that. This will ensure that my program will run at a more optimised rate, so this will further game immersion.

Forced fullscreen

```
</svg>
<div id="f11_popup_alert1" class="popup window">
  <div id="f11_popup_alert2">
    <p id="f11_popup_alert3">
      F11
    </p>
  </div>
</div>
<script>
```

Created a new div element which acts as a folder for other elements

Created a new div element which acts as a folder for other elements

Text element saying “F11” which is the hotkey to make the browser fullscreen

Forced fullscreen

```
.popup_window {  
    width: 100%;  
    height: 100%;  
    background-color: rgba(200,200,200,0.8);  
    top: 0px;  
    left: 0px;  
    display: block;  
    position: fixed;  
}  
#f11_popup_alert1 {  
    z-index: 100;  
}  
#f11_popup_alert2 {  
    width: 128px;  
    margin: 40px auto;  
    padding: 0px;  
    box-shadow: 0px 0px 8px 0px #555, 0px 0px 25px 0px #555 inset;  
    border: 3px solid;  
    border-radius: 20px;  
}  
#f11_popup_alert3 {  
    text-shadow: 0px 0px 10px #555;  
    font-size: 40px;  
}  
+rules
```

Sets up any element with the “popup_window” class as filling the browser tab, not scrolling and fills it a semi-transparent dark grey colour

Sets up any element with id “f11_popup_alert1” to be visibly in front of other layers (like the canvas)

Sets up any element with id “f11_popup_alert2” to be a square shape with rounded corners, and have an outer and inner shadow (which is a dark colour and blurred)

Sets up any element with id “f11_popup_alert3” to have a font size of 40px and an outer shadow (which is a dark colour and blurred)

Forced fullscreen

```
<script>
  function notfullscreen(){
    document.getElementById("f11_popup_alert1").style.display="block";
  }

  function fullscreen(){
    document.getElementById("f11_popup_alert1").style.display="none";
  }

  window.onresize=function(e){
    var maxHeight=window.screen.height, maxWidth=window.screen.width, curHeight>window.innerHeight, curWidth>window.innerWidth;
    if(maxWidth==curWidth&&maxHeight==curHeight){
      fullscreen();
    } else{
      notfullscreen();
    }
  };

```

When called, this function sets the display of the “f11_popup_alert1” to visible

When called, this function sets the display of the “f11_popup_alert1” to not visible

This function triggers when the browser window changes size

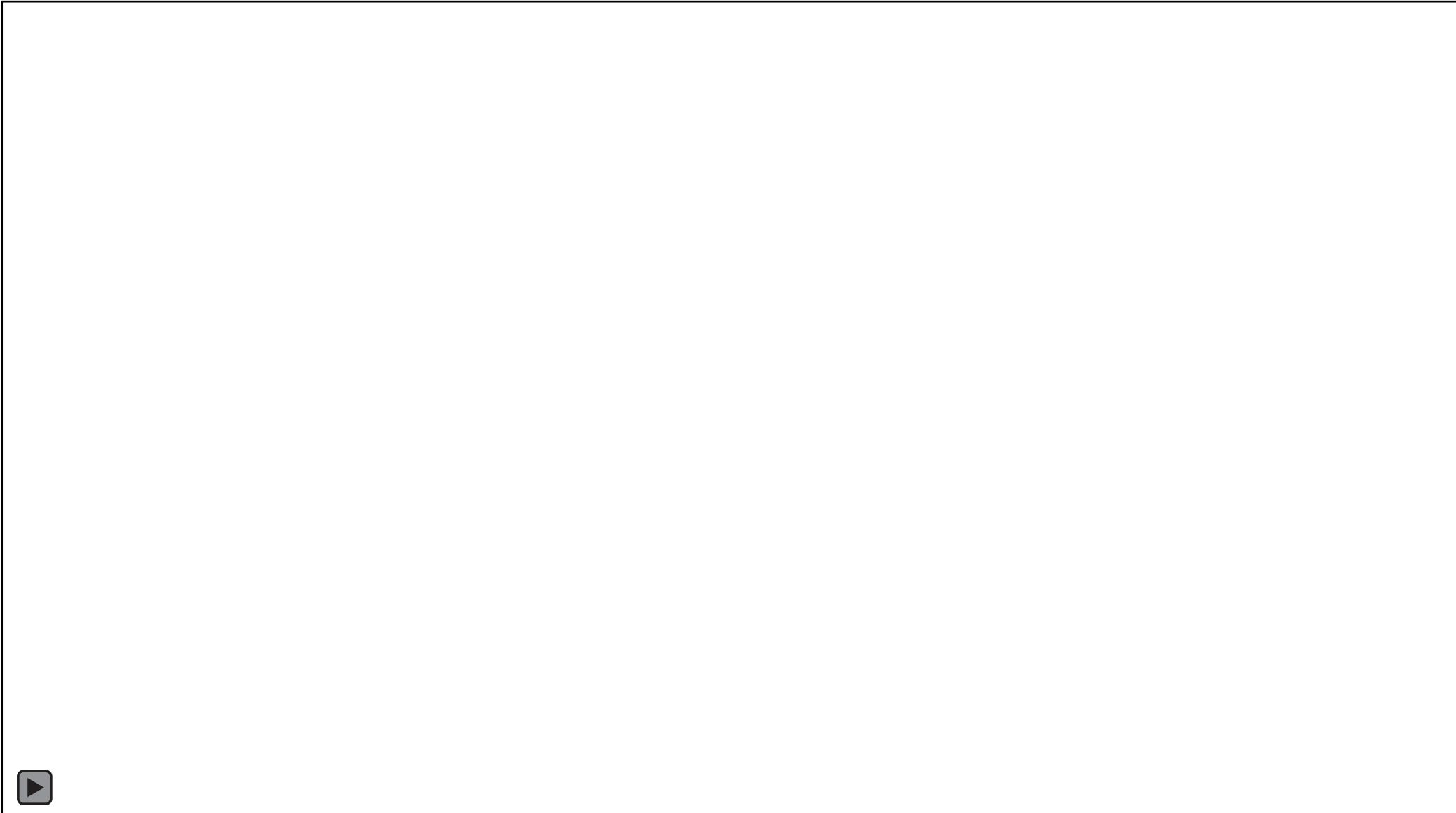
Creates 4 local variables and assigns the browser width and height, and window width and height to them

Checks if the browser width is equal to the window width and the same regarding height

If true then calls the fullscreen function

If not true then calls the notfullscreen function

Forced fullscreen – testing



Creating objects

As the game will have dynamic level loading, it is far more efficient to create the objects needed in each level on its load as opposed to having them all loaded all the time but having to decide which ones are visible and interactable.

As a result I wish to create a few functions that will create various SVG elements and append them to the SVG canvas. The functions will contain parameters for their position and dimensions.

Programming style

It is worth noting that I have decided to change from using a more object oriented programming approach to function/procedural programming. This is because I feel that JavaScript is easier to program in this manner. This does not change my initial pseudocode plan as I can simply transfer the classes to individual functions that use an object's attributes. Some parts of my code will use OOP as HTML itself is collections of classes with attributes and methods, which I will be making use of. However, the vast majority of my program will be functional based.

Creating objects – circle

SVG circles are composed of 3 key characteristics – their x position, y position and their radius. The x and y positions are for the centre of the circle (not the upper left corner like most other shapes).

Their structure appears like `<circle cx="" cy="" r="" />` however I also will need to include options for styling them (colour etc.) and their ID so it will more practically look like `<circle id="" cx="" cy="" r="" class="" />`.

Furthermore I wish to add the option to attach a filter to them. SVG filters can blur objects as well as other special effects. This may be useful for a more pleasing aesthetic later on.

Creating objects – circle

First of all it is necessary to create and define 2 variables as they will be repeatedly needed for these functions. The “svg” variable will be to reduce the amount of typing needed to find the SVG canvas element. The “svgns” will be used to tell the computer that the object that will be created is part of the SVG built in library.

```
var svgns="http://www.w3.org/2000/svg";  
var svg=document.getElementById("canvas");|
```

Creating objects – circle

When called, this function will create a circle based off user defined parameters for the x position, y position, radius, filter (if wanted), object id and style class.

```
function createcircle(posx, posy, radius, filter, id, classt){  
    var circle=document.createElementNS(svgns, "circle");  
    circle.setAttributeNS(null, "id", "c"+id);  
    circle.setAttributeNS(null, "cx", posx);  
    circle.setAttributeNS(null, "cy", posy);  
    circle.setAttributeNS(null, "r", radius);  
    if(filter!=""){  
        circle.setAttributeNS(null, "filter", filter);  
    }  
    circle.setAttributeNS(null, "class", classt);  
    svg.appendChild(circle);  
}
```

Creates a variable that is an SVG circle. This will later be appended to the canvas.

Sets the element ID to whatever the input “id” parameter specified with a “c” attached to the beginning

Sets the circle x position to the “posx” parameter

Sets the circle y position to the “posy” parameter

Sets the circle radius to the “radius” parameter

Checks if the “filter” parameter was left blank

If not then appends the “filter” parameter to the circle

Sets the style class of the circle to the “classt” parameter

Appends the circle element to the SVG canvas element in the document

Creating objects – rectangle

SVG rectangles are composed of 4 key characteristics – their x position, y position, width and height. The x and y positions are for the top left corner of the shape.

Their structure appears like `<rect x="" y="" width="" height="" />` however I also will need to include options for styling them (colour etc.) and their ID so it will more practically look like `<rect id="" x="" y="" width="" height="" class="" />`.

Again, I wish to add the option to attach a filter to them. SVG filters can blur objects as well as other special effects. This may be useful for a more pleasing aesthetic later on.

Creating objects – rectangle

Due to this being so similar to the circle generation script, it has not been annotated. It follows the exact same method, just with different names for the attributes.

```
function createrectangle(posx,posy,width,height,filter,id,classt){  
    var rectangle=document.createElementNS(svgns,"rect");  
    rectangle.setAttributeNS(null,"id","r"+id);  
    rectangle.setAttributeNS(null,"x",posx);  
    rectangle.setAttributeNS(null,"y",posy);  
    rectangle.setAttributeNS(null,"width",width);  
    rectangle.setAttributeNS(null,"height",height);  
    if(filter!=""){  
        rectangle.setAttributeNS(null,"filter",filter);  
    }  
    rectangle.setAttributeNS(null,"class",classt);  
    svg.appendChild(rectangle);  
}
```

Creating objects – polygon

SVG rectangles are composed of 1 key characteristic – their point positions.

Their structure appears like `<polygon points="" />` however I also will need to include options for styling them (colour etc.) and their ID so it will more practically look like `<polygon id="" points="" class="" />`.

Again, I wish to add the option to attach a filter to them. SVG filters can blur objects as well as other special effects. This may be useful for a more pleasing aesthetic later on.

Creating objects – polygon

Due to this being so similar to the circle generation script, it has not been fully annotated. It follows the exact same method, just with different names for the attributes.

```
function createpolygon(points,filter,id,classt){  
    var polygon=document.createElementNS(svgns,"polygon");  
    polygon.setAttributeNS(null,"id","p"+id);  
    var data_points="";  
    var max=points.length;  
    for(z=0;z<max;z++){  
        data_points+=points[z]+" ";  
    }  
    polygon.setAttributeNS(null,"points",data_points);  
    if(filter!=""){  
        polygon.setAttributeNS(null,"filter",filter);  
    }  
    polygon.setAttributeNS(null,"class",classt);  
    svg.appendChild(polygon);  
}
```

The points of a polygon element are stored as a string, but the parameter is an array

Stores the length of the string

Converts the array to a string separated by spaces

Creating objects – filters

SVG filters are composed of multiple elements – the filter and the modifier(s). The filter does not require any additional tags, but I wish to include ID, x, y, width and height tags as this will make it usable and will prevent cropping of effects outside of the shape's default bounding box.

The structures are <filter> <modifierName modifierData/> </filter>. An example for a blur effect: <filter ID="fblur" x="-100%" y="-100%" width="300%" height="300%> <feGaussianBlur stdDeviation="20"/> </filter>.

The filters need to be appended to the <defs> </defs> inside the SVG element – not the SVG itself.

Creating objects – filters

Due to this being so similar to the circle generation script, it has not been fully annotated. It follows the same method, just with different names for the attributes.

```
var fs=document.getElementById("filters");  
  
function createfilter(type,data,dimensions,id){  
    var filter=document.createElementNS(svgns,"filter");  
    filter.setAttribute("id","f"+id);  
    filter.setAttribute("x",dimensions[0]);  
    filter.setAttribute("y",dimensions[1]);  
    filter.setAttribute("width",dimensions[2]);  
    filter.setAttribute("height",dimensions[3]);  
    fs.appendChild(filter);  
    var max=data.length/2;  
    for(z=0;z<max;z++){  
        filter=document.createElementNS(svgns,type[z]);  
        filter.setAttributeNS(null,"in","SourceGraphic");  
        var placement=2*z;  
        filter.setAttributeNS(null,data[placement],data[placement+1]);\n        document.getElementById("f"+id).appendChild(filter);  
    }  
}
```

Creates a new variable defining the location of the element with the filters tag (the <defs> in the <svg>)

The dimensions parameter is a list containing x, y, width & height

This appends the filter element to the element defined in the fs variable

The data parameter is a list for the attribute names and values for the filter (as each one has different properties)

Creates the specific modifier element

Tells the modifier what graphics will change

Sets the modifier's name and value

Appends the modifier to the filter

Loop for multiple modifiers

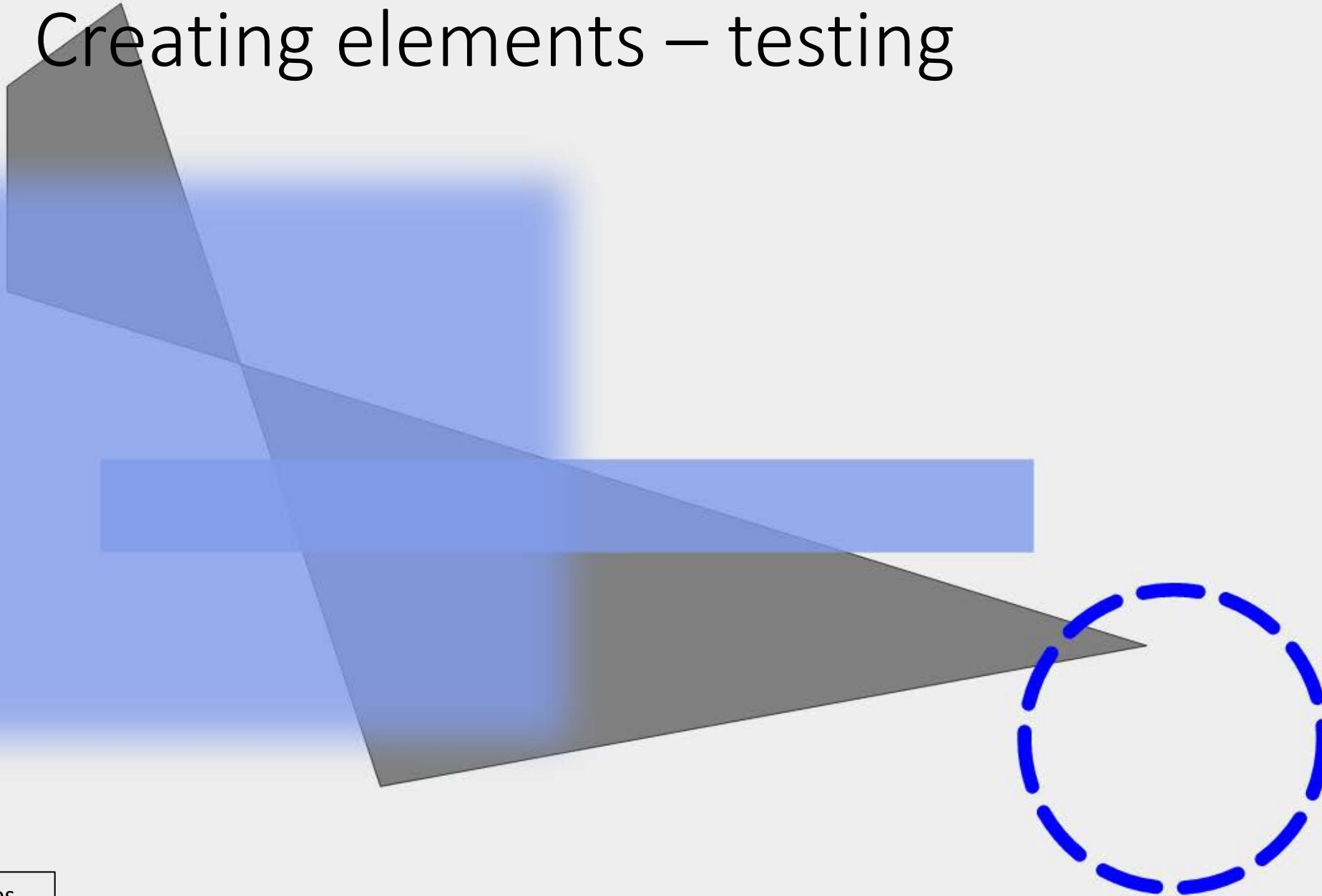
Creating elements – testing

```
.basic {  
    fill: grey;  
    stroke: black;  
    stroke-width: 1;  
    stroke-miterlimit: 10;  
}  
.dashed {  
    stroke: blue;  
    fill: none;  
    stroke-linecap: round;  
    stroke-dasharray: 60,30;  
    stroke-width: 15;  
}  
.filled {  
    stroke: none;  
    fill: rgba(128,155,235,0.8);  
}
```

This will generate a polygon, a circle, 2 rectangles and a filter. The filter will blur one of the 2 rectangles. The polygon will have a grey fill with a thin, black outline. The circle will have no fill, and a blue, dashed, rounded outline. Both rectangles will have no outline and be filled in a light blue colour that is semi-transparent.

```
createpolygon(["100,200","222,111","500,951","1321,800","100,420"], "", "polyshape", "basic");  
createfilter(["feGaussianBlur"], ["stdDeviation", 20], ["-50%", "-50%", "200%", "200%"], "blur");  
createcircle("1350", "900", "160", "", "circle", "dashed");  
createrectangle("-200", "300", "900", "600", "url(#fblur)", "rectangle", "filled");  
createrectangle("200", "600", "1000", "100", "", "rectangle", "filled");
```

Creating elements – testing



Creating elements – testing

As you can see, the topmost elements are the ones that are generated last. This will need to be taken into consideration when inserting the level elements and the player, because I cannot have the player hidden underneath another shape.

My code appears to have functioned fully as intended. For the most part, I will be using a black and white colour scheme with some grey in it, but I wish to allow customisability in a shop menu as stated in the objectives.

Drawing the levels

As each level will be premade, what I will need to do is create a method for drawing each level. For this I plan to use JavaScript objects to store the data that will be used in the HTML code. An example of this is would be a `<rect id="enclosure" x="0" y="0" width="15360" height="8640" class="obstacle_box"/>` being stored as `{id:"enclosure", type:"rect", x:"0", y:"0", class:"obstacle_box", width:"15360", height:"8640"}`. This will make the generation quite easy. An array of these objects will require basic iteration to create each object on the SVG canvas.

Drawing the levels

When called, this function will create a level with a parameter input of the level

```
function draw(lvl){  
    var objects=lvl.length;  
    xscroll=0, yscroll=0, xvelocity=0, yvelocity=0;  
    for(i=0;i<objects;i++){  
        var object=lvl[i];  
        if(object.type=="rect"){  
            creatrectangle(object.x,object.y,object.width,object.height,"",object.id,object.class);  
        }  
        else if(object.type=="circle"){  
            createcircle(object.x,object.y,object.r,"",object.id,object.class);  
        }  
        else if(object.type=="polygon"){  
            createpolygon(object.points,"",object.id,object.class);  
        }  
    }  
    createcircle("7680","4320","50","","player","st1");
```

A variable for the number of objects that need generating off the length of the input list

Creates variables that will be later used for the positioning of the canvas

Runs once per item in the input list

Selects the object to generate

Runs if the object is a rect

Generates a rectangle with no filter and the rest predetermined characteristics

Runs if the object is a circle

Generates a circle with no filter and the rest predetermined characteristics

Runs if the object is a polygon

Generates a polygon with no filter and the rest predetermined characteristics

Creates a player circle with radius 50 with a set style and ID for later use

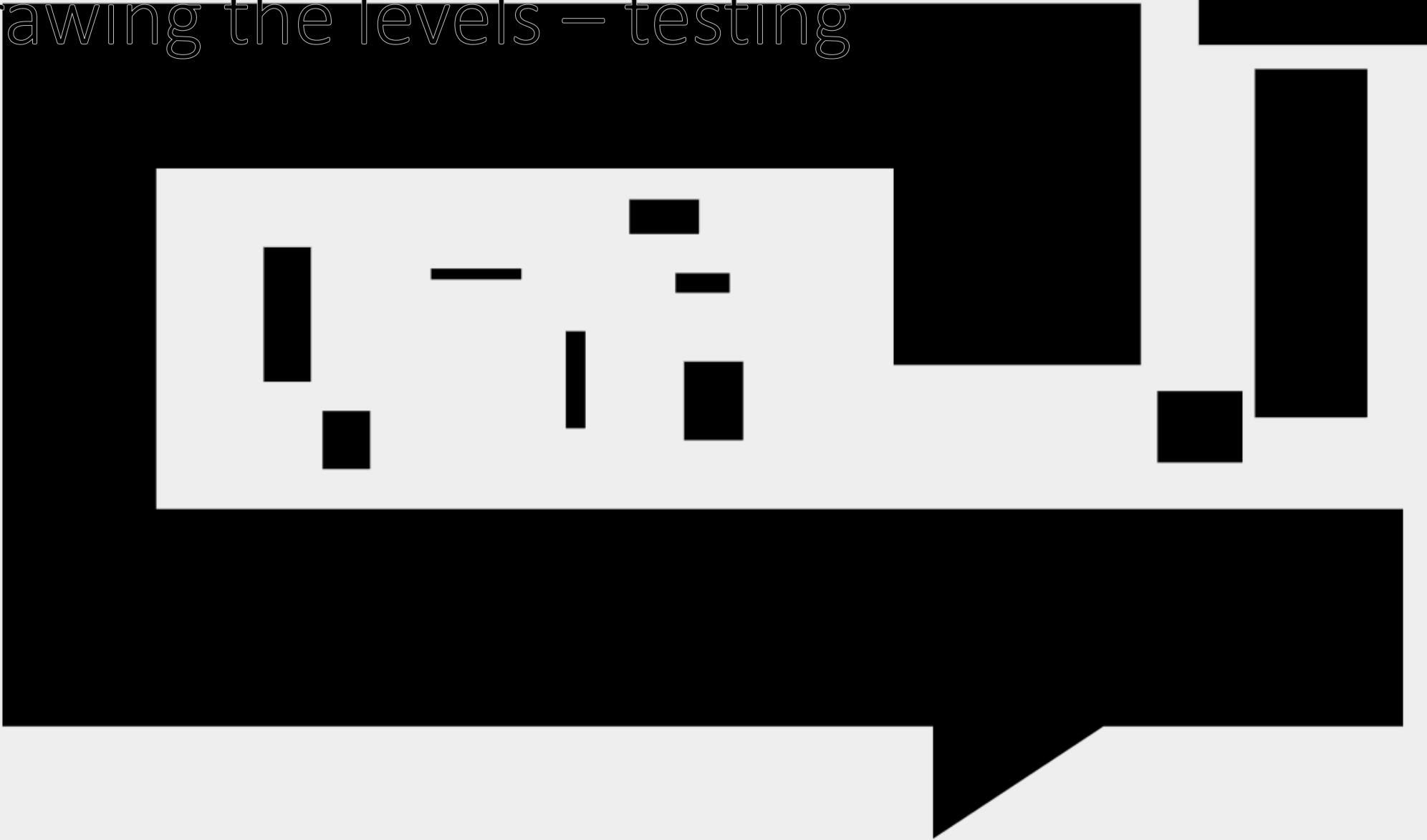
Drawing the levels – testing

```
draw([{id:"enclosure", type:"rect", x:"0", y:"0", class:"obstacle_box", width:"15360", height:"8640"}, {id:"2", type:"polygon", class:"obstacle_box", points:[ "1696,1089", "1696,7265", "9648,7265", "9648,8225", "11104.502,7265", "13664,7265", "13664,5409", "3008,5409", "3008,2497", "9312,2497", "9312,4177", "11424,4177", "11424,1089"]}], {id:"3", type:"rect", x:"7053.487", y:"2763.673", class:"obstacle_box", width:"593.881", height:"293.941"}, {id:"4", type:"rect", x:"3928.112", y:"3171.592", class:"obstacle_box", width:"401.92", height:"1148.409"}, {id:"5", type:"rect", x:"6510.149", y:"3889.612", class:"obstacle_box", width:"166.492", height:"827.959"}, {id:"6", type:"rect", x:"5358.207", y:"3354.139", class:"obstacle_box", width:"770.961", height:"91.495"}, {id:"7", type:"rect", x:"4431.253", y:"4570.578", class:"obstacle_box", width:"404.98", height:"494.975"}, {id:"8", type:"rect", x:"7521.099", y:"4149.099", class:"obstacle_box", width:"503.975", height:"670.466"}, {id:"9", type:"rect", x:"7447.602", y:"3393.137", class:"obstacle_box", width:"461.977", height:"166.492"}, {id:"10", type:"rect", x:"11566.396", y:"4401.086", class:"obstacle_box", width:"725.964", height:"608.97"}, {id:"11", type:"rect", x:"12400", y:"1649", class:"obstacle_box", width:"960", height:"2976"}, {id:"12", type:"rect", x:"11920", y:"657", class:"obstacle_box", width:"2080", height:"784"}]);
```

```
... ...
<svg id="canvas" viewBox="0 0 15360 8640">
```

This code will generate a basic idea for a level. While it may seem complex, it is only made of 12 different objects. The styles for the level have been defined also. I have also had to increase the viewBox dimensions to show the entire design.

Drawing the levels – testing



Creating the player

- The player will be composed of 2 elements – a circle for the appearance of the player, and a square for the player's hitbox.
- The hitbox will be used to determine whether the player has collided with any walls and if so then move the player backwards.
- Both elements will need to be in the same position, and the same sizes.

Creating the player

- I can use the premade functions for `createcircle` and `createrectangle` to speed up the process of generating the player elements. I shall call the circle element “player” as this is what the user will view as the player, and the hitbox as “hb” (short for hitbox).
- I think that a good size for the player would be 100x100 pixels (or a circle radius 50 pixels).
- The starting point for the player and hitbox are irrelevant, as they will be automatically moved each frame.

```
createcircle("7680", "4320", "50", "", "player", "st1");
createrectangle("0", "0", "100", "100", "", "hb", "st2");
```

Centring the player

- The player should be in the middle of the screen.
- This means that I not only need to change the screen's perspective, but also move the player to the middle.
- The best way to do this would be to move the visible player element, and then the hitbox element to the same place.

Centring the player

As this process will need to be performed every frame, I shall make it into a function

```
player = document.getElementById("cplayer");
```

A global variable called player is used to interact with the player circle

```
function centerplayer() {  
    player.setAttributeNS(null, "cx", (xscroll + 1920 / 2));  
    player.setAttributeNS(null, "cy", (yscroll + 1080 / 2));  
    var hitbox = document.getElementById("rhb");  
    hitbox.setAttributeNS(null, "width", player.r.animVal.value * 2);  
    hitbox.setAttributeNS(null, "height", player.r.animVal.value * 2);  
    hitbox.setAttributeNS(null, "transform", ("translate(" + (xscroll + 1920 / 2 - player.r.animVal.value) + ", " + (yscroll + 1080 / 2 - player.r.animVal.value) + ")"));  
}
```

Sets the centre of the player circle to the middle of the screen

Resizes the hitbox to perfectly fit the visible player

Sets the top left corner of the hitbox to align with the visible player.

Centring the screen

The `svg` element conveniently has a built in `viewBox` feature where you can specify the position of the top left corner of the screen, along with the width and height of the element which we have already used.

```
svg.setAttribute("viewBox", (xscroll + " " + yscroll + " " + 1920 + " " + 1080));  
centerplayer();
```

Sets the top left corner of the `svg` element and also sets the width and height to default values (1920x1080 pixels)

Creating a loop

- The player and screen centring will need to be updated every frame, so should be put inside a loop.
- As requested, I will set the loop to run 60 times a second to have a 60fps experience.
- This loop will run every 16.666666666667 milliseconds, which is $1000/60$ or 60 times per seconds.

```
window.setInterval(function () {
    svg.setAttribute("viewBox", (xscroll + " " + yscroll + " " + 1920 + " " + 1080));
    centerplayer();
}, 16.66666666666667);
```

Adding player movement

- For this function, we will need to know when the user is pressing a key.
- A good way to do this would be by having an array which would have a boolean value of “true” or “false” relating to whether a key is pressed at a given time.
- It will also be useful to allow the F1-F12 keys to still do their normal functions, but prevent other keys from doing anything while the browser is in focus.

Adding player movement

- To achieve this, I will use a built in method to detect if any key is pressed, and then set the value of an array to be true. When the key is released, another built in method will set the value of the array to false.
- However, having an array of all of the buttons on a keyboard is highly inefficient, so I will instead use an object which will add the key field when necessary.

Adding player movement

```
var keystate = {};
```

Creates the object that will have a readout for each key press

```
window.addEventListener("keydown", function (e) {
```

Triggers an event when any key is pushed down

```
    if (!((e.keyCode || e.which) > 111) && ((e.keyCode || e.which) < 124))) {
```

Passes a value which contains the key's id value

```
        keystate[e.keyCode || e.which] = true;
```

Checks that the key is not F1-F12

```
        e.preventDefault();
```

As different browsers have different methods of detecting which key is pressed,
this will return whichever value is generated and set the object key to true

```
    }, true);
```

Stops the key from doing anything

```
window.addEventListener("keyup", function (e) {
```

Triggers an event when any key is released

```
    keystate[e.keyCode || e.which] = false;
```

Sets the object key to false

```
}, true);
```

Key press – testing

This test involved me pressing individual keys at first, and then repeatedly hitting multiple keys at random. It also included the F1-F12 keys which resulted in the chrome help page opening up. As my function is meant to allow the F1-F12 keys to work as normal, this shows that it has been a success. The “state” of each key (true for pressed, false for

ers normal and extremeous data input for key

Adding player movement

```
window.setInterval(function () {  
    if (keystate[87]) { //w  
        yscroll -= 10; }  
    if (keystate[83]) { //s  
        yscroll += 10; }  
    if (keystate[65]) { //a  
        xscroll -= 10; }  
    if (keystate[68]) { //d  
        xscroll += 10; }  
    svg.setAttribute("viewBox", (xscroll + " " + yscroll +  
        " " + 1920 + " " + 1080));  
    centerplayer();  
}, 16.666666666667);
```

Checks if a key is pressed or not

Updates the yscroll variable

Ditto but for the xscroll variable

Runs 60 times per second

Non-linear player movement

- The current movement method works, but is linear in that the player moves at a constant rate rather than accelerating and decelerating.
- This makes moving the player less interesting.
- To make it more engaging, I wish to have non-linear player velocity.
- This can be achieved by having a velocity variable for each axis which is damped over time.
- For example, when a button is pressed then the velocity increases by 4, and is then divided by 2, then the velocity is added to the current position. If a button is not pressed then it does the same as before but does not increase the velocity.

Non-linear player velocity

```
window.setInterval(function () {
    if (keystate[87]) { //w
        yvelocity -= 2; -----> Updates the yvelocity variable
    }
    if (keystate[83]) { //s
        yvelocity += 2;
    }
    yvelocity *= 0.92; -----> Updates the yvelocity variable
    yscroll += yvelocity; -----> Updates the yscroll variable
    if (keystate[65]) { //a
        xvelocity -= 2;
    }
    if (keystate[68]) { //d
        xvelocity += 2;
    }
    xvelocity *= 0.92;
    xscroll += xvelocity;
    svg.setAttribute("viewBox", (xscroll + " " + yscroll +
    " " + 1920 + " " + 1080));
    centerplayer();
}, 16.666666666667);
```

Player collision

- The player needs to not be able to just travel through walls that compose the level, otherwise this makes the purpose of the maze redundant.
- The program will check if the player has collided with a wall after each position variable has updated. This means that if the player were to bump a wall, the program would know if it had been a result of a change in the x or y position, and so reverse the player's motion.
- The check will be done inside a function that returns a boolean value of true or false, depending on if it has collided or not.

Player collision

```
function checkcollision() {
    var r = player.r.animVal.value,
        frame = document.getElementById("renclosure").getBBox(),
        i;
    if ((player.cx.animVal.value < r) || (player.cy.animVal.value < r) || (player.cx.animVal.value > frame.width - r) ||
        (player.cy.animVal.value > frame.height - r)) {
        return true;
    }
    var collisions = [],
        len, r0 = document.getElementById("rhb").getBoundingClientRect(),
        r1 = svg.createSVGRect();
    r1.x = r0.left;
    r1.y = r0.top;
    r1.width = r0.width;
    r1.height = r0.height;
    var collisions_node = svg.getIntersectionList(r1, null);
    len = collisions_node.length;
    for (i = 0; i < len; i += 1) {
        collisions.push(collisions_node[i]);
    }
    for (i = len - 1; i >= 0; i -= 1) {
        if (collisions[i].id == "renclosure" || collisions[i].id == "cplayer" || collisions[i].id == "rhb") {
            collisions.splice(i, 1);
        }
    }
    if (collisions.length > 0) {
        return true;
    } else {
        return false;
    }
}
```

Gets the player circle's radius

Gets the enclosing rectangles dimensions

Checks the player is inside the bounding rectangle

Returns that a collision has occurred (true)

Gets the bounding rectangle for the hitbox element

Creates a temporary element to match the bounding rectangles dimensions

Creates a node list of each element that intersects the new temporary element created above

Generates a standard array from the node list (node lists are read only so cannot be edited)

As the hitbox is always: within the bounding rectangle, is touching the player and touching the hitbox, these items are removed from the array

Returns that a collision hasn't occurred (false)

Player collision

```
window.setInterval(function () {  
    if (keystate[87]) { //w  
        yvelocity -= 2;  
    }  
    if (keystate[83]) { //s  
        yvelocity += 2;  
    }  
    yvelocity *= 0.92;  
    yscroll += yvelocity;  
    centerplayer();  
    if (checkcollision()) {  
        yvelocity *= -1;  
        yscroll += yvelocity;  
    }  
    if (keystate[65]) { //a  
        xvelocity -= 2;  
    }  
    if (keystate[68]) { //d  
        xvelocity += 2;  
    }  
    xvelocity *= 0.92;  
    xscroll += xvelocity;  
    centerplayer();  
    if (checkcollision()) {  
        xvelocity *= -1;  
        xscroll += xvelocity;  
    }  
    svg.setAttribute("viewBox", (xscroll + " " + yscroll + " " + 1920 + " " + 1080));  
    centerplayer();  
}, 16.666666666667);
```

Centres the player

Checks if the player has collided with anything

If it has then the velocity is reversed, and the position updated

!!Player collision!!

- A major bug has occurred with this method, as the large polygon object's bounding rectangle includes all of it's contents, meaning that it is unfeasible to use polygons.
- As a result, scenery must NOT be made out of polygon objects, and instead only rectangles.
- This is not possible to show in a video as the player simply appears to collide with nothing, and I could only discover that this was the issue by using breakpoints in my code and checking variables to see which shape(s) the player had collided with, which turned out to be the polygon.

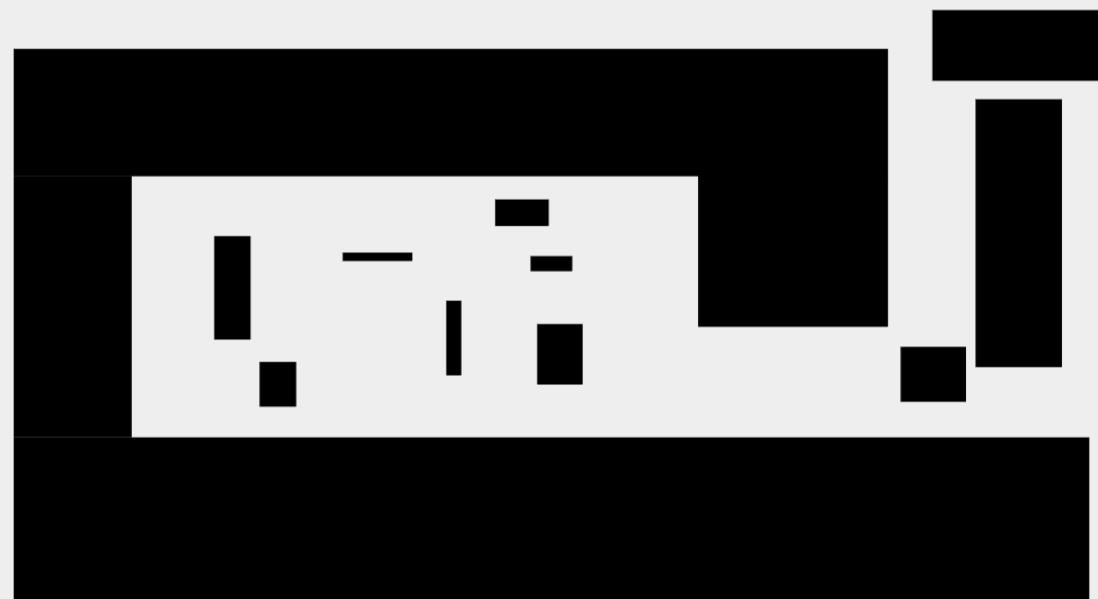
!!Player collision!!

- Another result of this unexpected problem, was that the polygon and circle tables will no longer be used, as the collision with those elements will cause problems.
- This means levels must be made only out of rectangle elements.

Changing the level

Because of the previous bug, the level has been changed to be made only out of rectangles. It is very similar to the previous level, but does not have any diagonals in it. I have also decided to make the level be half the size in each dimension as I felt it was far too large, so it is now $\frac{1}{4}$ of the original size.

The small quarter circle visible bottom right is the player sprite. It has temporarily been moved out of the main viewing window.



Play testing



Creating the shadows

- This procedure will be very complex as it must effectively ray-trace from the corners of each rectangle away from the player (so should dynamically update with the player).
- To reduce load on the user's computer, shadows should only be drawn if they are in view. To achieve this I will only draw them if they are within a giant + shape centred on the player (illustrated by image below).
- Each shadow will be a six sided polygon (irregular hexagon) where some of the points may end up in the same place, making it appear to have fewer points.

Drawing mock up

I will have shadows!

Draw zone (for
shadows)

I will not have shadows!!

No draw zone
(for shadows)

Obstacle box

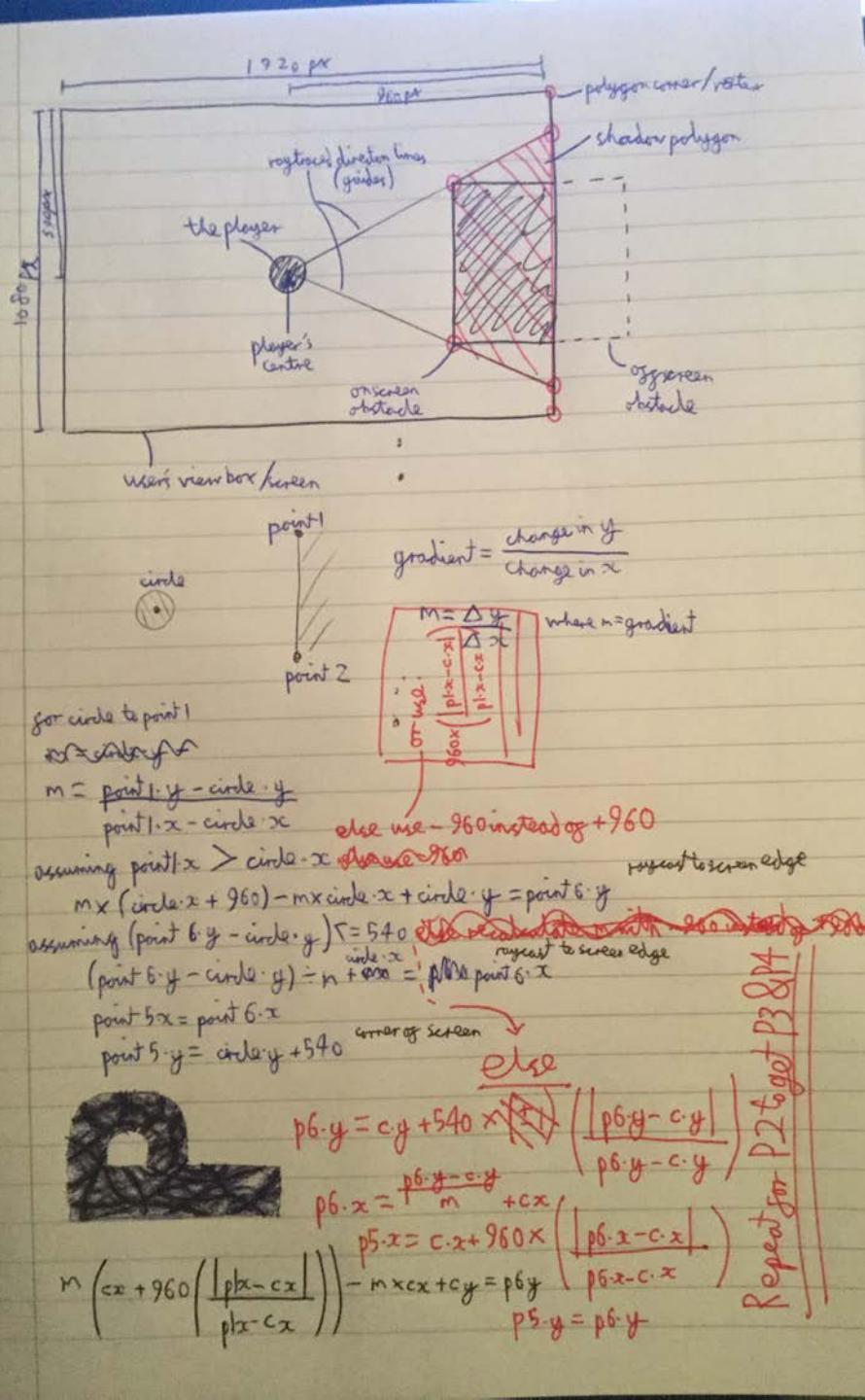
Visible area of
player's screen

The player sprite

The maths for calculating the shadow's shape

- Each shadow will inherit 2 points from the parent rectangle. These will be from the side that is casting the shadow.
- The centre of the ray casting will be from the centre of the screen, which is also the player's centre.
- The other 4 points will be calculated using the algorithm shown below, worked out as shown below.
- The previously found bug with the polygon collision has made it so that I need only calculate shadows based off rectangular elements rather than polygons.

The maths for calculating the shadow's shape



```

PROCEDURE createShadow(shape)
    points = [shape.x, shape.y, shape.x, shape.y + shape.height, shape.x + shape.width, shape.y + shape.height, shape.x + shape.width, shape.y]
    counter = 0
    WHILE counter < 4
        p1.x = points[(counter * 2) MOD 8]
            p1.y = points[(counter * 2 + 1) MOD 8]
            p2.x = points[(counter * 2 + 2) MOD 8]
            p2.y = points[(counter * 2 + 3) MOD 8]
            IF (((ABS(p1.x - c.x) <= 960) OR (ABS(p2.x - c.x) <= 960)) OR ((p1.x - c.x < 960) AND (p2.x - c.x > 960)) OR ((p1.x - c.x > 960) AND (p2.x - c.x < 960))) AND ((ABS(p1.y - c.y) <= 540) OR (ABS(p2.y - c.y) <= 540)) OR ((p1.y - c.y < 540) AND (p2.y - c.y > 540)) OR ((p1.y - c.y > 540) AND (p2.y - c.y < 540))) THEN
                m := (p1.y - c.y) / (p1.x - c.x) //BEGIN OF REPEAT
                p6.y := m * (c.x + 960 * (ABS(p1x - cx) / (p1x - cx))) - m * c.x + c.y
                IF (ABS(p6.y - c.y) <= 540) THEN
                    p6.x := (p6.y - c.y) / m + c.x
                    p5.x := p6.x
                    p5.y := c.y + 540 * (ABS(p6.y - c.y) / (p6.y - c.y))
                ELSE
                    p6.y := c.y + 540 * (ABS(p6.y - c.y) / (p6.y - c.y))
                    p6.x := (p6.y - c.y) / m + c.x
                    p5.x := c.x + 960 * (ABS(p6.x - c.x) / (p6.x - c.x))
                    p5.y := p6.y
                END IF
                //REPEAT SUBBING p1 for p2, p6 for p3, p5 for p4 from the "M = ..." line (with the comment)
            END IF
            counter += 1
        END WHILE
        shape := Shape.Polygon.create()
        shape.setVertices([p1,p2,p3,p4,p5,p6])
        shape.setId("s" + shape.id)
        ScreenCanvas.appendShape(shape)
    END PROCEDURE

```



Creating the shadows

- We also need to remove all of the shadows from the previous frame. This can easily be done by selecting them by their class (“shadowray”) and “killing” each element.
- This will most likely be more efficient than checking whether an element needs to exist or not in the next frame, as each vertex is checked anyway.

A function that redraws the shadows in the level

Creating the shadows

```
function shadowrays() {
    var kill = document.getElementsByClassName("shadowray");
    while (kill[0]) {
        kill[0].parentNode.removeChild(kill[0]);
    }
    var len = level.length,
        shape = {},
        points = [],
        poly = [],
        p1x, p2x, p1y, p2y, p3x, p3y, p4x, p4y, p5x, p5y, p6x, p6y, cx = player.cx.animVal.value,
        cy = player.cy.animVal.value,
        m, i, n;
    for (i = 0; i < len; i += 1) {
        shape = level[i];
        points = [shape.x, shape.y, shape.x, Number(shape.y) + Number(shape.height), Number(shape.x) + Number(shape.width),
        Number(shape.y) + Number(shape.height), Number(shape.x) + Number(shape.width), shape.y];
        for (n = 0; n < 4; n += 1) {
            p1x = points[(n * 2) % 8];
            p2x = points[(n * 2 + 2) % 8];
            p1y = points[(n * 2 + 1) % 8];
            p2y = points[(n * 2 + 3) % 8];
            if (((Math.abs(p1x - cx) <= 960) || (Math.abs(p2x - cx) <= 960) || ((p1x - cx < 960) && (p2x - cx > 960)) || ((p1x - cx > 960) && (p2x - cx < 960))) && ((Math.abs(p1y - cy) <= 540) || (Math.abs(p2y - cy) <= 540)) || ((p1y - cy < 540) && (p2y - cy > 540)) || ((p1y - cy > 540) && (p2y - cy < 540))) && shape.id != "enclosure") {
                m = (p1y - cy) / (p1x - cx);
                p6y = m * (cx + 960 * (Math.abs(p1x - cx) / (p1x - cx))) - m * cx + cy;
                if (Math.abs(p6y - cy) <= 540) {
                    p6x = (p6y - cy) / m + cx;
                    p5x = p6x;
                    p5y = cy + 540 * (Math.abs(p6y - cy) / (p6y - cy));
                } else {
                    p6y = cy + 540 * (Math.abs(p6y - cy) / (p6y - cy));
                    p6x = (p6y - cy) / m + cx;
                    p5x = cx + 960 * (Math.abs(p6x - cx) / (p6x - cx));
                    p5y = p6y;
                }
            }
        }
    }
}
```

Selects and removes all elements with the class “shadowray”

Iterates through each shape in the level

JavaScript version of the previous algorithm

Iterates through each side of the rectangle

Continued below

Creating the shadows

```
m = (p2y - cy) / (p2x - cx);
p3y = m * (cx + 960 * (Math.abs(p2x - cx) / (p2x - cx))) - m * cx + cy;
if (Math.abs(p3y - cy) <= 540) {
    p3x = (p3y - cy) / m + cx;
    p4x = p3x;
    p4y = cy + 540 * (Math.abs(p3y - cy) / (p3y - cy));
} else {
    p3y = cy + 540 * (Math.abs(p3y - cy) / (p3y - cy));
    p3x = (p3y - cy) / m + cx;
    p4x = Number(cx) + 960 * (Math.abs(p3x - cx) / (p3x - cx));
    p4y = p3y;
}
poly = [];
poly.push(p1x + ',' + p1y, p2x + ',' + p2y, p3x + ',' + p3y, p4x + ',' + p4y, p5x + ',' + p5y, p6x + ',' + p6y);
createpolygon(poly, "", ("s" + i + n), "shadowray");
}
```

Repeats for points 3 and 4

Creates an array of all the points
in the polygon

Creates the polygon shape with a
unique ID and class “shadowray”

Shadows – testing

The player and border have temporarily been restyled to make the shadows more obvious than what will actually be in the game. This change
is only for this test and is later reverted.



Stylising the webpage

In the future, the hitbox will be hidden, but for development purposes I will keep it shown around the player circle.

I am happy with how the shadows and obstacles appear, and the light grey background seems to work well. Initially I was unsure whether it should be changed to white, but I feel that there would be no benefit to this, and could in fact simply make the contrast too extreme which would be unattractive.

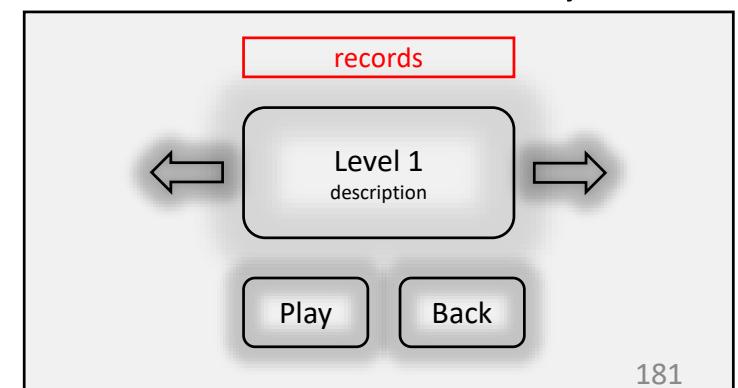
Collision changes

At this stage I have decided that the player bounces off obstacles too far, so I will add a “dampening” effect to the bounce, meaning that the velocity will be 3/5 of it’s speed before the collision. This will make the player easier to control.

```
centerplayer();
if (checkcollision()) {
    yscroll -= yvelocity;
    yvelocity *= -0.6;
}
```

Level selection

- As the game is intended to have multiple levels, it seems sensible to have a level selection screen so that the player can easily choose which level to participate in.
- This level screen will also show extra details such as record times, as well as the details of the actual level.
- The initial mock-up (right) does not show these extra details. I believe that they should be at the top of the screen, above the level name, as this is very noticeable.



Level selection

- Key points:
 - Each level will need a display title
 - Each level will need a display description
 - There should be a play button
 - There should be a back button (return to main menu – when made)
 - There should be navigation arrows (left/right) to change the level

Level selection

```
<div id="level_selector" class="level_selectors">
    <!--greyish background-->
    <div id="filler_bg"> </div>
    <!--the first main display box-->
    <div class="level">
        <!--the styling for the border-->
        <div id="lmain" class="display large_glow">
            <!--The records-->
            <p id="level_all_record" class="record all_record">All time record: </p>
            <p id="level_own_record" class="record own_record">Your record: </p>
            <!--level title-->
            <p class="name" id="level_name_text">setup</p>
            <!--level description-->
            <p class="description" id="level_description_text">first level I made</p>
        </div>
    </div>
    <!--a button that isn't actually a button but acts like a button. used to trigger playing a level-->
    <div class="button_large large_glow" id="play_level_button">
        <!--text-->
        <p>Play</p>
    </div>
    <!--ditto but for going back to the menu-->
    <div class="button_large large_glow" id="exit_level_selector_button">
        <p>Back</p>
    </div>
</div>
```

A grey background

The record readouts

The level name

The level description

The play button

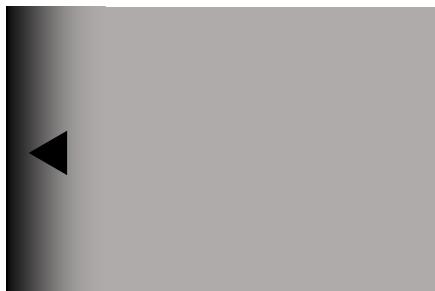
The back button

Level selection GUI



Level selection

- After some consideration, I have decided to change the style of arrows from my original plan. This is to make the user interface more attractive and immersive.
- I wish to make it so that there is a button that is the full height of the screen, with a small arrow overlaying the button pointing to the left/right. The full height button will not appear as a standard button, but instead a gradient region that darkens the edge of the screen (see below)



Level selection

```
</div>
<!--the arrow on the right of the screen to change the level--&gt;
&lt;div id="arrow_right" class="arrow_bg"&gt;
    <!--the pointer--&gt;
    &lt;div id="arrow_right_pointer" class="arrow_pointer"&gt; &lt;/div&gt;
&lt;/div&gt;
<!--ditto but left, not right--&gt;
&lt;div id="arrow_left" class="arrow_bg"&gt;
    &lt;div id="arrow_left_pointer" class="arrow_pointer"&gt; &lt;/div&gt;
&lt;/div&gt;
<!--a button that isn't actually a button but acts like a button.</pre>
```

The gradient button

The arrow pointer

Level selection final GUI



Level selection

- All buttons should be animated to change when the user moves the mouse over the button, to clearly indicate that if pressed then an event can happen.
- The play and back button will be made to appear lighter when the mouse is touching them.
- The left/right arrows will become darker when having a mouse over.
- They should also turn the mouse pointer into another image to make it even clearer.

Level selection – button testing



Level selection

- When the user changes the level, I wish to have an animated carousel effect for the transition. This means that the current selected level needs to “fly-off” to one side while the new level will need to “fly-in” from the opposite side.
- I will achieve this effect by using CSS animations which can be toggled by changing the class of the element.
- At any point, there will only be 2 level displays on the screen (the one flying off, and the other flying on). This means that I can optimise and reduce load and memory size by only using 2 level displays, which is more efficient than having 1 per level.

Level selection

```
var levelsdata = [{
    "id": "l0",
    "name": "level 1",
    "description": "description",
},
{
    "id": "l1",
    "name": "level 2",
    "description": "description 2",
},
{
    "id": "l2",
    "name": "level 3",
    "description": "let us see how long this can be before it overflows and causes the styling to fail. CAN IT SUPPORT CAPITALS? numbers 1234567890? symbols? !£$%^&*()_+{}@~:?:><",
    "locked": true,
}],
maindisplayname = document.getElementById("level_name_text"),
maindisplaydesc = document.getElementById("level_description_text"),
maindisplaybox = document.getElementById("lmain"),
maindisplayownrecord = document.getElementById("level_own_record"),
maindisplayotherrecord = document.getElementById("level_all_record"),
subdisplayname = document.getElementById("secondary_level_name_text"),
subdisplaydesc = document.getElementById("secondary_level_description_text"),
subdisplaybox = document.getElementById("lsecondary"),
subdisplayownrecord = document.getElementById("secondary_level_own_record"),
subdisplayotherrecord = document.getElementById("secondary_level_all_record"),
levelshown = 0,
current_times = ["12", "13", "73"];
```

An array of the level information

“shortcuts” to various elements
to save unnecessary typing

The currently displayed level

The current users times (milliseconds)

Level selection

Calls a procedure on the click of the left button that changes the displayed level

```
document.getElementById("arrow_left").addEventListener("click", function () {
    arrowl();
});

document.getElementById("arrow_right").addEventListener("click", function () {
    arrowr();
});

function mod(n, m) {
    return ((n % m) + m) % m;
}
```

A modulo function that returns positive values if the input is negative

Level selection

```
function arrowr() {
    levelshown += 1;
    maindisplaybox.classList.remove("slide_r2m");
    maindisplaybox.classList.remove("slide_l2m");
    void maindisplaybox.offsetWidth;
    maindisplaybox.classList.add("slide_r2m");
    maindisplayname.innerHTML = levelsdata[mod(levelshown, levelsdata.length)].name;
    maindisplaydesc.innerHTML = levelsdata[mod(levelshown, levelsdata.length)].description;
    var time = current_times[mod(levelshown, levelsdata.length)] / 1000;
    if (typeof time == "number" && time != 0) {
        maindisplayownrecord.innerHTML = "Your record: " + time;
    } else {
        maindisplayownrecord.innerHTML = "----";
    }
    maindisplayotherrecord.innerHTML = "All time record: ---- (user)";
    if (levelsdata[mod(levelshown, levelsdata.length)].locked === true) {
        maindisplayname.classList.add("locked");
        maindisplaydesc.classList.add("locked");
        maindisplayownrecord.classList.add("locked");
    } else {
        maindisplayname.classList.remove("locked");
        maindisplaydesc.classList.remove("locked");
        maindisplayownrecord.classList.remove("locked");
    }
    subdisplaybox.classList.remove("slide_m2l");
    subdisplaybox.classList.remove("slide_m2r");
    void subdisplaybox.offsetWidth;
    subdisplaybox.classList.add("slide_m2l");
    subdisplayname.innerHTML = levelsdata[mod(levelshown - 1, levelsdata.length)].name;
    subdisplaydesc.innerHTML = levelsdata[mod(levelshown - 1, levelsdata.length)].description;
    time = current_times[mod(levelshown - 1, levelsdata.length)] / 1000;
    if (typeof time == "number" && time != 0) {
        subdisplayownrecord.innerHTML = "Your record: " + time;
    } else {
        subdisplayownrecord.innerHTML = "----";
    }
    subdisplayotherrecord.innerHTML = "All time record: ---- (user)";
    if (levelsdata[mod(levelshown - 1, levelsdata.length)].locked === true) {
        subdisplaydesc.classList.add("locked");
        subdisplayname.classList.add("locked");
        subdisplayownrecord.classList.add("locked");
    } else {
        subdisplaydesc.classList.remove("locked");
        subdisplayname.classList.remove("locked");
        subdisplayownrecord.classList.remove("locked");
    }
}
```

Updates the level shown

Adds the slide-in animation

Sets the level name and description

Sets the level user time (if applicable, and displays "----" if not)

Blurs the view if the level is still locked
except for the highscore

Sets the secondary display (for slide-off) as above with the level
that was shown until the user clicked the button

There is a very similar script for the
function arrowl, which just has
different animation names and
changes levelshown by -1. It is not
shown as it is so similar to this one.

Level selection – functionality testing



Creating levels

- As levels are in the form of SVG elements, I can easily produce them using an SVG editor.
- I have chosen to use Adobe Illustrator ©, but there are many other editors out there which will work just as well.
- I will need to convert the raw SVG file code into the format used by my scripts.
- To do this I will create a small utility program that converts SVG into my format, so it can be read in the main game file.

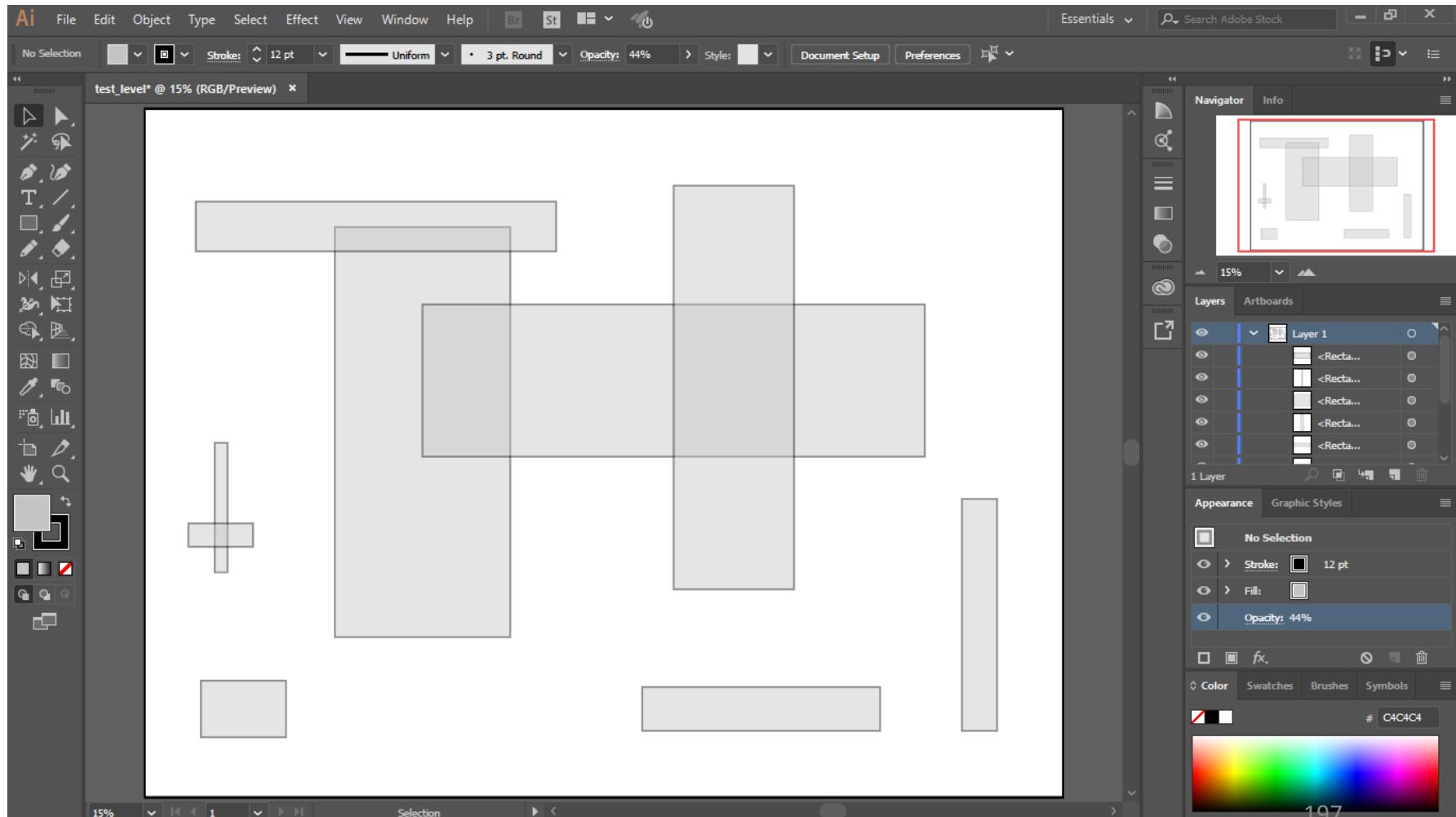
Creating levels

I have decided that I will include a meta item about the level inside the large array that will be produced. This will always be the first item in each level array, and will include information on the level's name, description, ID, player spawn point, level exit, and bits (the number of collectible points that will be made to spawn in the level at a later date).

A sub-program will be made for this purpose, which will be included in the final project upload inside a folder (named “utility” or similar).

Creating levels

An example of a level design in Adobe Illustrator ©



Creating levels

The above example,
exported as SVG code
(source)



A screenshot of a Windows Notepad window titled "ai14892329273.txt - Notepad". The window contains an XML document representing an SVG file. The XML code defines a style for a class "st0" with a black stroke and no fill, and a large rectangle with dimensions 5760x4320 pixels centered at (-0.5, -0.5). It also defines several smaller nested rectangles with various widths and heights, all sharing the same "st0" class. The Notepad window has standard Windows-style minimize, maximize, and close buttons in the top right corner.

```
<?xml version="1.0" encoding="utf-8"?>
<!-- Generator: Adobe Illustrator 21.0.2, SVG Export Plug-In . SVG Version: 6.00 Build 0) -->
<svg version="1.1" id="Layer_1" xmlns="http://www.w3.org/2000/svg" xmlns:xlink="http://www.w3.org/1999/xlink" x="0px" y="0px"
      viewBox="0 0 5760 4320" style="enable-background:new 0 0 5760 4320;" xml:space="preserve">
<style type="text/css">
    .st0{fill:none;stroke:#000000;stroke-width:12;stroke-miterlimit:10;}</style>
<rect x="-0.5" y="-0.5" class="st0" width="5760" height="4320"/>
<rect x="1193.5" y="743.5" class="st0" width="1100" height="2573"/>
<rect x="1743.5" y="1229.5" class="st0" width="3150" height="954"/>
<rect x="3318.5" y="483.5" class="st0" width="755" height="2533"/>
<rect x="320.5" y="583.5" class="st0" width="2260" height="313"/>
<rect x="3120.5" y="3629.5" class="st0" width="1493" height="274"/>
<rect x="5126.5" y="2449.5" class="st0" width="220" height="1454"/>
<rect x="353.5" y="3589.5" class="st0" width="533" height="354"/>
<rect x="440" y="2097" class="st0" width="80" height="813"/>
<rect x="274" y="2603" class="st0" width="406" height="147"/>
</svg>
```

Creating levels

Configures the webpage and style

```
<!DOCTYPE html>
<html>

<head>
  <meta charset="UTF-8">
  <title> Computing coursework </title>
  <style>
    html,
    body {
      font-family: Calibri;
      margin: 0px;
      width: 100%;
      height: 100%;
      text-align: center;
      font-size: 20px;
      padding: 0px;
    }
  </style>
</head>
```

This is where the result will be shown

```
<p id="text_out">output</p>
<script>
```

Asks the user for a text string input of SVG code

```
var svginput = window.prompt("svg code").split("/> <");
svginput.splice(0, 1, svginput[0].replace("<", ""));
var len = svginput.length;
```

Turns the string into an array of each element

```
svginput.splice((len - 1), 1, svginput[len - 1].replace("/>", ""));
var svgoutput = [{}];
```

Removes extra characters

```
"id": "",
"name": "",
"description": "",
"xspawn": 0,
"yspawn": 0,
"xexit": 1920,
"yexit": 1080,
"bits": 0
```

Defines the output with default values

```
], object, working, leni, workingi, n, i;
```

Loops through each item in the array

```
for (n = 0; n < len; n++) {
```

The first item is always the enclosing rectangle

```
object = {};
```

Otherwise it is just another random item

```
working = svginput[n].split(" ");
```

```
if (n == 0) {
```

```
object.id = "enclosure";
```

```
} else {
```

```
object.id = String(n);
```

Creating levels

Sets the shape type (e.g. "rect")

```
object.type = working[0];
leni = working.length;
for (i = 1; i < leni; i++) {
```

Loops through the current SVG item

```
workingi = working[i].split("=")
```

Sets the x position if item is x

```
if (workingi[0] == "x") {
```

Sets the y position if item is y

```
object.x = Math.round(Number(workingi[1].replace(/\//gi, "")));
```

Sets the class of the item

```
} else {
```

Sets the width and height

```
if (object.x == null) {
```

SVG removes the x/y if they are 0,
so I re-add it

```
object.x = 0;
```

```
if (workingi[0] == "y") {
```

```
object.y = Math.round(Number(workingi[1].replace(/\//gi, "")));
```

```
} else {
```

```
if (object.y == null) {
```

```
object.y = 0;
```

```
}
```

```
if (workingi[0] == "class") {
```

```
object.class = workingi[1].replace(/\//gi, "");
```

```
if (object.class == "st0") {
```

```
object.class = "obstacle_box"
```

```
} else if (object.class == "st1") {
```

```
object.class = "obstacle_box lethal"
```

```
}
```

```
} else if (workingi[0] == "width") {
```

```
object.width = Math.round(Number(workingi[1].replace(/\//gi, "")));
```

```
} else if (workingi[0] == "height") {
```

```
object.height = Math.round(Number(workingi[1].replace(/\//gi, "")));
```

```
}
```

```
}
```

```
svgoutput.push(object);
```

Adds the object to the output array

```
}
```

```
document.getElementById("text_out").innerHTML = JSON.stringify(svgoutput);
```

Displays the output array

```
</script>
```

```
</body>
```

```
</html>
```

Creating levels – testing/example



Creating levels

Levels created in this way can simply be copied and pasted into an array of levels. Because this will obviously become quite a large script as more levels are created, I believe it is worthwhile putting the levels into a separate script so that I can keep the main HTML document clean from large arrays.

Storing levels

```
var levels = [[{ //huge array of arrays of objects for the format of each level. the basic structure is that the first object is the level meta data, the 2nd is the surrounding wall, and beyond that it's just each shape that makes up the level
  "id": "l0", //level id
  "name": "pretty", //level name
  "description": "oooooo shadows", //description
  "xspawn": 0, //where the player starts
  "yspawn": 0,
  "xexit": 2200, //where the exit is
  "yexit": 2200,
  "bits": 50 //umber of qubits in the level
}, {
  id: "enclosure", //shape id
  type: "rect", //shape type
  x: 0, //top left x
  y: 0, //top left y
  class: "obstacle_box", //class for style
  width: 5760, //width
  height: 2160 //height
}, {
  id: "1",
  type: "rect",
  x: 150,
  y: 200,
  class: "obstacle_box",
  width: 100,
  height: 100
}, {
  id: "2",
  type: "rect",
  x: 350,
  y: 200,
  class: "obstacle_box",
  width: 100,
  height: 100
}, {
  id: "3",
  type: "rect",
  x: 550,
  y: 200,
  class: "obstacle_box",
  width: 100,
  height: 100
}, {
  id: "4",
  type: "rect",
  x: 750,
  y: 200,
  class: "obstacle_box",
  width: 100,
  height: 100
}], [{"x": 0, "y": 0, "order": 1}, {"x": 5760, "y": 0, "order": 2}, {"x": 5760, "y": 2160, "order": 3}, {"x": 0, "y": 2160, "order": 4}, {"x": 0, "y": 0, "order": 5}], [{"x": 0, "y": 0, "order": 1}, {"x": 150, "y": 200, "order": 2}, {"x": 350, "y": 200, "order": 3}, {"x": 550, "y": 200, "order": 4}, {"x": 750, "y": 200, "order": 5}, {"x": 0, "y": 0, "order": 6}]]
```

The separate document with example data

```
<script src="assets/map_data.js" id="map_data"></script>
```

Links to the separate document

Due to the size of the document, not all of the code has been shown. It is highly repetitive, as all follows the same format.

Loading levels

- Currently the level selector and playable game components are in 2 separate files, which need to be combined together.
- The playable component should only be triggered when the play button has been pressed.

Loading levels

```
<div id="overscroll"> </div>
<svg id="canvas" viewBox="0 0 7680 4320" class="hide">
  <defs id="filters"> </defs>
</svg>
```

The SVG element is hidden by default

```
document.getElementById("play_level_button").addEventListener("click", function () {
  level = levels[mod(levelshown, levelsdata.length)]
  draw(level);
  document.getElementById("level_selector").classList.add("hide");
  svg.classList.remove("hide");
});
```

When the play button is pressed, the level is selected and drawn, and the selection menu is hidden and the SVG element shown

```
    cy = player.cy.animVal.value
    m, i, n;
for (i = 1; i < len; i += 1) {
  shape = level[i];
  points = [shape.x, shape.y,
```

The shadowray and draw functions now ignore the first metadata item of each level (the first item)

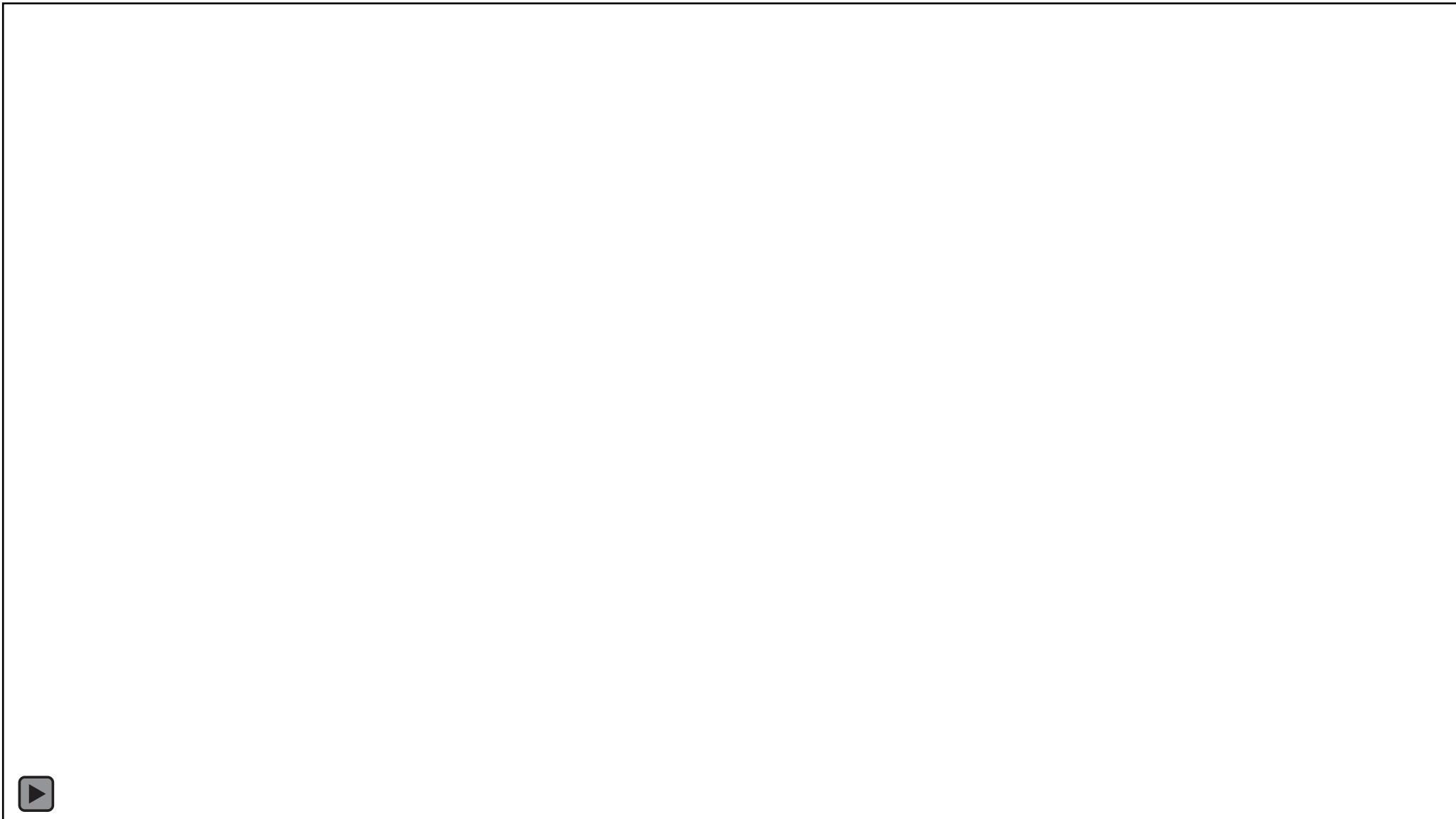
```
function draw(lvl) {
  var objects = lvl.length;
  xscroll = -100, yscroll = -100, xvelocity = 0, yvelocity = 0;
  for (i = 1; i < objects; i++) {
    var object = lvl[i];
    if (object.type == "rect") {
```

```
} window.setInterval(function () {
  if (!svg.classList.contains("hide")) {
    if (keystate[87]) { //w
```

The main game loop now only runs if the SVG element is visible (so when a level has been selected) 205

Loading levels – testing

The level selector does not immediately show the correct first level. This will be fixed by making the program call the arrowl or arrowr function on start, forcing it to update.



Saving the levels

- Each level should have data stored in SQL tables. There will be 2 tables – Levels and Rectangles.
- The Levels table will be used to store the meta data about each level (id, name, description, xspawn, yspawn, xexit, yexit, bits).
- The rectangles table will contain all of the rectangle elements used in the levels (associated level, id, x, y, width, height, class).
- Because JavaScript is not meant to be used to generate any user files, I will have to use the WebSQL feature to create the tables the first time the game is launched.

Saving the levels

```
var db = openDatabase('Overshadowed database', '1.0', 'all data', 4 * 1024 * 1024);
```

Creates the SQL file with maximum size 4MB

```
var commands = ['DROP TABLE IF EXISTS LEVELS', 'DROP TABLE IF EXISTS RECTANGLES', 'CREATE TABLE RECTANGLES (level, id, x, y, width, height, class)', 'CREATE TABLE LEVELS (id, name, description, xspawn, yspawn, xexit, yexit, bits)'],
    initcommands = ['CREATE TABLE IF NOT EXISTS RECTANGLES (level, id, x, y, width, height, class)', 'CREATE TABLE IF NOT EXISTS LEVELS (id, name, description, xspawn, yspawn, xexit, yexit, bits)'];
```

```
function initialise() {
    var n;
    for (n = 0; n < initcommands.length; n += 1) {
        (function (n) {
            db.transaction(function (tx) {
                tx.executeSql(initcommands[n]);
            });
        })(n);
    }
    setup();
}
```

Arrays of command SQL statements

Loops through and runs the statements in the initcommands array

Calls the setup procedure

```
function setup() {
    var n;
    for (n = 0; n < 1; n += 1) {
        var len;
        (function (n) {
            db.transaction(function (tx) {
                tx.executeSql('SELECT * FROM RECTANGLES', [], function (tx, results) {
                    len = results.rows.length;
                    regen(len);
                }, null);
            });
        })(n);
    }
}
```

Forces asynchronous callback (prevents SQL statements being processed simultaneously and failing)

Gets the contents of the RECTANGLES table

Gets the length of the output

Passes the value into the regen procedure

Saving the levels

```
function regen(length) {  
    var levels_inner = levels_inner_length();  
    if (length != levels_inner) {  
        var n;  
        for (n = 0; n < commands.length; n += 1) {  
            (function (n) {  
                db.transaction(function (tx) {  
                    tx.executeSql(commands[n]);  
                });  
            })(n);  
        }  
        create_levels();  
        create_rects();  
    } else {  
        getlevelsdata();  
    }  
}
```

Gets the number of objects in the levels array

If the variables do not match, the table needs
regenerating

Loops through and runs the statements in the
commands array

Generates the levels and rectangles tables

Reads the level data

```
"use strict"; //enforces certain special keyword uses, helps with a minor bug that could happen  
var levels_inner = 0;  
for (var i = 0; i < levels.length; i += 1) { //a loop for each array  
    levels_inner += levels[i].length - 1; //increments but removes  
}  
return levels_inner;
```

Iterates through the levels array for each
shape object

Saving the levels

```
function create_levels() {
    var i;
    for (i = 0; i < levels.length; i += 1) {
        (function (i) {
            var str = 'INSERT INTO LEVELS (id, name, description, xspawn, yspawn, xexit, yexit, bits) VALUES ("' + levels[i][0].id + '", "' + levels[i][0].name + '", "' + levels[i][0].description + '", "' + levels[i][0].xspawn + '", "' + levels[i][0].yspawn + '", "' + levels[i][0].xexit + '", "' + levels[i][0].yexit + '", "' + levels[i][0].bits + '")';
            db.transaction(function (tx) {
                tx.executeSql(str);
            });
        })(i);
        if (i == levels.length - 1) {
            getlevelsdata();
        }
    }
}

function create_rects() {
    var n, levelt, i;
    for (n = 0; n < levels.length; n += 1) {
        levelt = levels[n];
        for (i = 1; i < levelt.length; i += 1) {
            (function (i) {
                var str = 'INSERT INTO RECTANGLES (level, id, x, y, width, height, class) VALUES ("' + levelt[0].id + '", "' + levelt[i].id + '", "' + levelt[i].x + '", "' + levelt[i].y + '", "' + levelt[i].width + '", "' + levelt[i].height + '", "' + levelt[i].class + '")';
                db.transaction(function (tx) {
                    tx.executeSql(str);
                });
            })(i);
        }
    }
}
```

Iterates through each level

Appends the meta data to the SQL table from the levels array

Reads the level data

Iterates through each level

Iterates through each object per level

Appends the shape data to the SQL table from the levels array

Reading the levels

```
function getlevelsdata() {  
    levelsdata = [];  
    var len, i;  
    db.transaction(function (tx) {  
        tx.executeSql('SELECT * FROM LEVELS', [], function (tx, results) {  
            len = results.rows.length;  
            for (i = 0; i < len; i += 1) {  
                var obj = {};  
                obj.id = results.rows.item(i).id;  
                obj.name = results.rows.item(i).name;  
                obj.description = results.rows.item(i).description;  
                obj.locked = false;  
                obj.bits = results.rows.item(i).bits;  
                levelsdata.push(obj);  
            }  
        }, null);  
    });  
}
```

Reads the LEVELS array

Iterates through the SQL query output

Creates the level meta data object

Appends the meta data to the levelsdata array

Reading the levels

```
function getlevel(level_id) {  
    var len, i, objects = [];  
    db.transaction(function (tx) {  
        tx.executeSql('SELECT * FROM LEVELS', [], function (tx, results) {  
            len = results.rows.length;  
            for (i = 0; i < len; i += 1) {  
                if (level_id == JSON.stringify(results.rows.item(i).id)) {  
                    var obj = {};  
                    obj.id = results.rows.item(i).id;  
                    obj.name = results.rows.item(i).name;  
                    obj.description = results.rows.item(i).description;  
                    obj.xspawn = Number(results.rows.item(i).xspawn);  
                    obj.yspawn = Number(results.rows.item(i).yspawn);  
                    obj.xexit = Number(results.rows.item(i).xexit);  
                    obj.yexit = Number(results.rows.item(i).yexit);  
                    obj.bits = Number(results.rows.item(i).bits);  
                    objects.push(obj);  
                    leveltoobj(level_id, objects);  
                }  
            }  
        }, null);  
    });  
}
```

Reads the LEVELS array

Iterates through the SQL query output

Creates the level meta data object

Appends the meta data to the levelsdata array

Calls a procedure to generate an array of all
the rectangles in the level

Reading the levels

```
function leveltoobj(level_id, objects) {  
    db.transaction(function (tx) {  
        tx.executeSql('SELECT * FROM RECTANGLES WHERE level=' + level_id, [], function (tx, results) {  
            var len = results.rows.length,  
                object, i;  
            for (i = 0; i < len; i += 1) {  
                object = {};  
                object.type = "rect";  
                object.id = results.rows.item(i).id;  
                object.class = results.rows.item(i).class;  
                object.x = Number(results.rows.item(i).x);  
                object.y = Number(results.rows.item(i).y);  
                object.width = Number(results.rows.item(i).width);  
                object.height = Number(results.rows.item(i).height);  
                objects.push(object);  
            }  
            level = objects;  
        }, null);  
    });  
}
```

Reads the RECTANGLES array with matching ID

Iterates through the SQL query output

Generates an object with rectangle data

Appends the object to an array

Sets the level to the objects array

Reading from SQL levels

```
initialise();  
try {  
    arrowl();  
    window.setTimeout(function () {  
        arrowr();  
    }, 150);  
} catch (err) {}
```

If a computer is running slowly, prevents a critical crash as does not run if the SQL has not completed

```
document.getElementById("play_level_button").addEventListener("click", function () {  
    getlevel('' + mod(levelshown, levelsdata.length) + ''');  
    window.setTimeout(function () {  
        draw(level);  
        window.setTimeout(function () {  
            shadowrays();  
            document.getElementById("level_selector").classList.add("hide");  
            svg.classList.remove("hide");  
        }, 50);  
    }, 250);  
});
```

Get the level from the SQL tables

Delay so that the SQL can complete

Delay so that the level is drawn before shadows

Loading animation

- Currently, the transition between the level selection screen and the playable level is rather jarring.
- Additionally, there is a slight delay between the SQL query and the level drawing, which breaks the in-game immersion.
- As a result, creating a loading animation seems to be the logical way of making this transition and combatting the loading time issue.
- The animation will not be a progress bar, but simply an indicator that the program is still running and processing, and hasn't simply frozen.

Loading animation

```
<!--loading bar screen-->
<div class="blackbg" id="bgblack">
    <!--the loading bar positioning-->
    <div class="loading" id="loadingbar">
        <!--the loading bar that is animated-->
        <div id="progress"> </div>
    </div>
</div>
```

The HTML elements for the loading bar

```
function initialiseloading() {
    document.getElementById("loadingbar").classList.remove("fly_up");
    document.getElementById("bgblack").classList.remove("fade_bg");
}

function clearloading() {
    document.getElementById("loadingbar").classList.add("fly_up");
    document.getElementById("bgblack").classList.add("fade_bg");
}
```

Triggers the loading animation to appear

Triggers the loading animation to vanish

Loading animation

```
document.getElementById("play_level_button").addEventListener("click", function () {
    getlevel("l" + mod(levelshown, levelsdata.length) + "");
    initialiseloading();
    window.setTimeout(function () {
        draw(level);
        shadowrays();
        document.getElementById("level_selector").classList.add("hide");
        svg.classList.remove("hide");
        svg.setAttribute("viewBox", (xscroll + " " + yscroll + " 1920 1080"));
        clearloading();
    }, 1000);
});
```

Triggers the loading animation to appear

Waits 1 second

Updates the viewBox

Hides the loading animation

```
initialise();
initialiseloading();
window.setTimeout(clearloading, 2000);
try {
    arrowl();
    window.setTimeout(function () {
        arrowr();
    }, 150);
} catch (err) {}
```

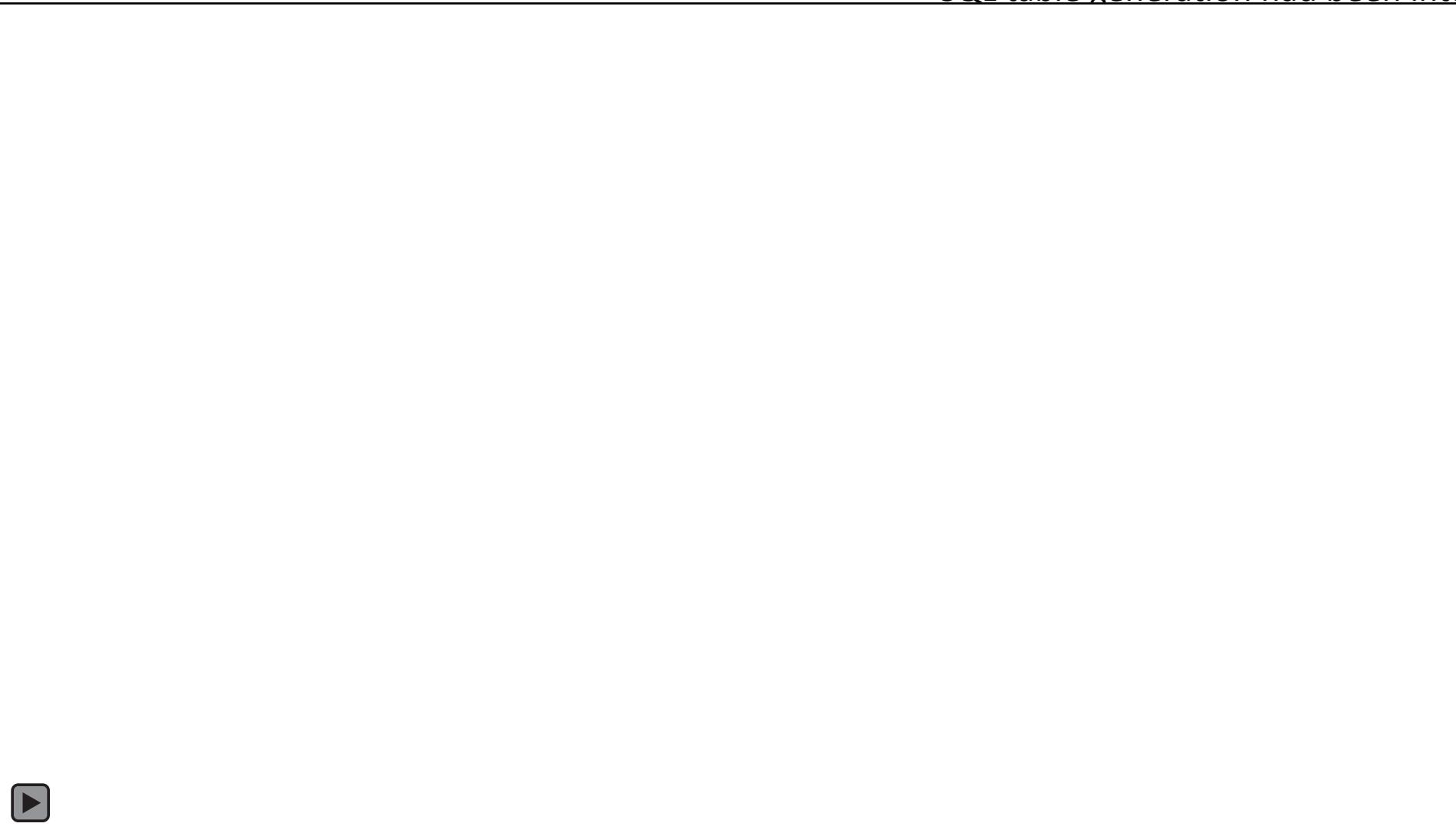
Triggers the loading animation to appear

Waits 2 seconds then hides the animation

Loading animation – testing

As can be seen, the level fails to load. After having seen this, I checked the SQL table and it appeared to have failed to generate the LEVELS table correctly (missing off the first and last levels). Further analysis shows that this was because the SQL table generation had been interrupted by a

making the
interaction.



The previously seen bug has not been fixed, but has been avoided for this test, meaning that the level now loads correctly.

Loading animation – testing



Level exit

- The point of the game is to try to beat the record times of other players on the same network.
- To do this, there needs to be a point when the timer terminates (the end of each level).
- When a level has been completed, the user should be automatically returned to the level selection screen and made aware of whether they have got a new record or not.
- Seeing as there is not currently a user save system, the first step towards this should be a level exit.

Level exit

Creates the SVG rectangle in the draw function

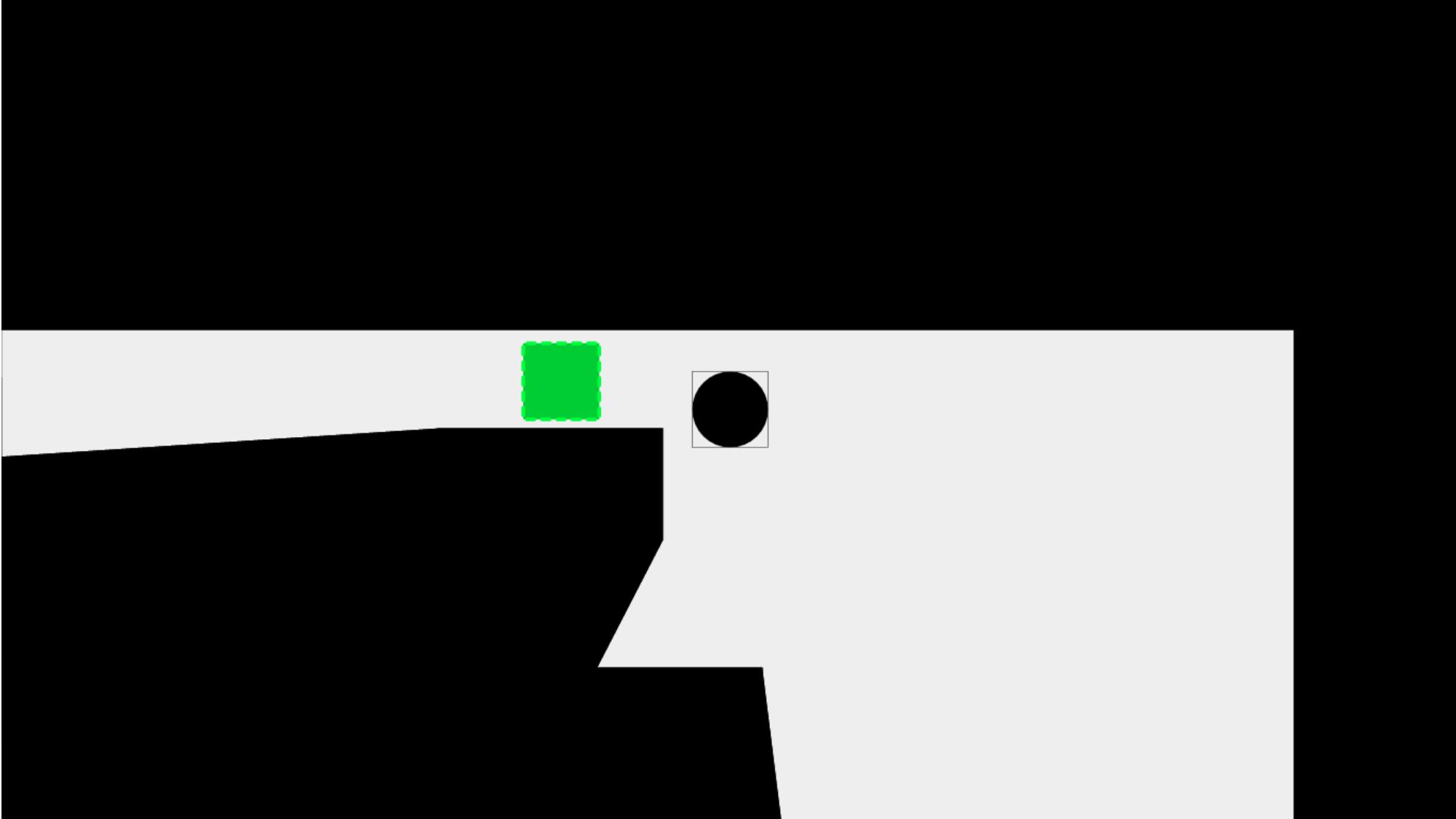
```
createrectangle("0", "0", "100", "100", "", "no", "st2");
createrectangle(lvl[0].xexit - 50, lvl[0].yexit - 50, "100", "100", "", "exit", "exit");
player = document.getElementById("player");
```

```
collisions.splice(1, 1);
} else if (collisions[i].id == "rexit") {
    initialiseloading();
    window.setTimeout(function () {
        document.getElementById("level_selector").classList.remove("hide");
        svg.classList.add("hide");
        clearloading();
    }, 1000);
}
```

Checks if the player has touched the exit rectangle

Triggers the loading animation, hides the level and shows the level selection screen

Level exit GUI



Level clearing

- The level should be cleared when completed so that a new level isn't loaded on top of an old one.
- This will use a technique similar to that in the shadowray function.

Level clearing

```
function clear() {
    var kill, i;
    for (i = 1; i < level.length; i += 1) {
        if (level[i].type == "rect") {
            kill = document.getElementById("r" + level[i].id);
            kill.parentNode.removeChild(kill);
        }
    }
    var killlist = ["rexit", "rhb", "cplayer"];
    for (i = killlist.length - 1; i >= 0; i -= 1) {
        kill = document.getElementById(killlist[i]);
        kill.parentNode.removeChild(kill);
    }
    kill = document.getElementsByClassName("shadowray");
    while (kill[0]) {
        kill[0].parentNode.removeChild(kill[0]);
    }
}
```

Iterates through the levels list to remove the rectangle elements

Removes all rect elements

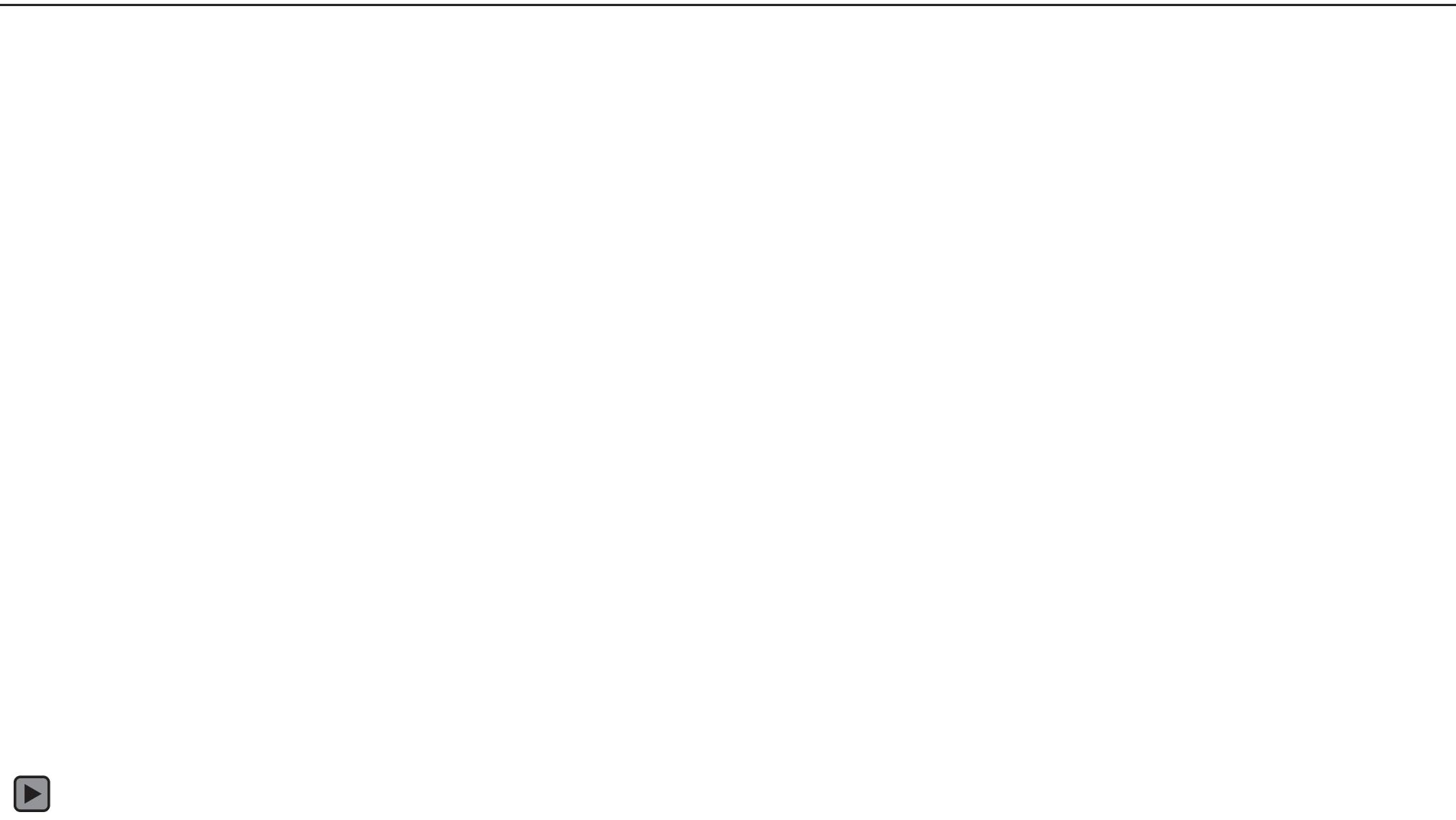
Removes the player, hitbox and exit

Iterates through the premade elements to remove them

Removes the shadows

If a shadow element exist, it is removed

Level clearing – testing



User accounts

- A vital part of the game is user accounts.
- Each user will have the ability to play and unlock levels, as well as have their high score shown on the leader board.
- Each user has the ability to be an admin.

User accounts – tables

```
var commands = ['DROP TABLE IF EXISTS LEVELS', 'DROP TABLE IF EXISTS RECTANGLES', 'DROP TABLE IF EXISTS SAVEFILES', 'DROP TABLE IF EXISTS  
SAVEPROGRESS', 'CREATE TABLE RECTANGLES (level, id, x, y, width, height, class)', 'CREATE TABLE LEVELS (id, name, description, xspawn, yspawn,  
xexit, yexit, bits)', 'CREATE TABLE SAVEFILES ("id" NOT NULL UNIQUE, name, admin, qbits, reset, PRIMARY KEY("id"))',  
initcommands = ['CREATE TABLE IF NOT EXISTS RECTANGLES (level, id, x, y, width, height, class)', 'CREATE TABLE IF NOT EXISTS LEVELS (id,  
name, description, xspawn, yspawn, xexit, yexit, bits)', 'CREATE TABLE IF NOT EXISTS SAVEFILES ("id" NOT NULL UNIQUE, name, admin, qbits,  
reset, PRIMARY KEY("id"))'];
```

Updated commands and initcommands arrays

```
function setupsaves() {  
    var str = "(pid",  
        n;  
    for (n = 0; n < levels.length; n += 1) {  
        str += ",l" + levels[n][0].id + "items,'l" + levels[n][0].id + "time' NUMERIC";  
    }  
    str += ")";  
    db.transaction(function (tx) {  
        tx.executeSql('CREATE TABLE SAVEPROGRESS ' + str);  
    });  
    window.setTimeout(function () {  
        createsavefile("0", "ADMIN", true, false);  
        window.setTimeout(function () {  
            opensavesdatabase(true);  
        }, 500);  
    }, 500);  
}
```

Iterates for each level in the array

Generates a string for a time and item per level

Creates the table with the fields generated above

Creates a save file and opens the saves database

User accounts – tables

```
function opensavesdatabase(reset) {
    if (reset) {
        logindetails = [];
    }
    db.transaction(function (tx) {
        tx.executeSql('SELECT * FROM SAVEFILES', [], function (tx, results) {
            var i, obj;
            for (i = 0; i < results.rows.length; i += 1) {
                obj = {};
                obj.id = results.rows.item(i).id;
                obj.name = results.rows.item(i).name;
                obj.admin = results.rows.item(i).admin;
                obj.qbits = results.rows.item(i).qbits;
                obj.reset = results.rows.item(i).reset;
                logindetails.push(obj);
            }
        }, null);
    });
}
```

Clears the table based off parameters

Selects all the save files

Iterates through the SQL query output

Appends all of the attributes per user to an array

User accounts – tables

```
function createsavefile(id, name, admin, update) {
    var str = "(pid",
    str2 = '(' + id + '",',
    obj, n;
    for (n = 0; n < levels.length; n += 1) {
        str += ",l" + levels[n][0].id + "items,l" + levels[n][0].id + "time";
        if (admin) {
            str2 += ',0",0';
        } else {
            str2 += '","",""';
        }
    }
    str += ")";
    str2 += ")";
    db.transaction(function (tx) {
        tx.executeSql('INSERT INTO SAVEFILES (id,name,admin,qbits,reset) VALUES (' + id + '","' + name + '","' + admin + '",0,"false")');
        tx.executeSql('INSERT INTO SAVEPROGRESS ' + str + ' VALUES ' + str2);
    });
    if (update === true) {
        obj = {};
        obj.id = id;
        obj.name = name;
        obj.admin = admin;
        obj.qbits = 0;
        obj.reset = false;
        logindetails.push(obj);
    }
}
```

Iterates for each item in the levels array

Fills in the parameters of the table with 0s or blanks

Appends to the savefiles and saveprogress tables

Adds the new user to the user array based off parameters

User accounts – tables

Tables (5)

- > LEVELS
- > RECTANGLES
- > SAVEFILES
- > SAVEPROGRESS
- > _WebKitDatabaseInfoTable_

The generated tables (including one made by the browser)

Table: **SAVEPROGRESS**

pid	l0items	l0time	l1items	l1time	l2items	l2time	l3items	l3time	l4items	l4time	l5items	l5time	l6items	l6time
Filter	Fil...	...												
1 3718274321299354	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table: **SAVEFILES**

id	name	admin	qbits
Filter	Filter	Filter	Filter
1 37182743212...	ADMIN	true	0

Table: **LEVELS**

	id	name	description	xspawn	yspawn	xexit	yexit	bits
1	l10	NEW	NEW broken	-400	20	540	3050	80
2	l9	Nuance	please fix this	0	0	4810	4560	80
3	l8	Islands	avoid the isla...	0	0	4334	2520	80
4	l7	mini maze	tricky	-170	-400	2184	2864	80
5	l6	boring	ugh	0	0	3440	1290	80
6	l5	lethal maze	hard	-760	-340	9020	160	80
7	l4	lethal	teaching abou...	0	0	3695	460	80
8	l3	quiz	okay	0	0	250	5140	80
9	l2	maze	good luck	0	0	3120	1840	80
10	l1	Beginner	Classic	-400	-200	3690	1320	100

All figures pretty

Table: **RECTANGLES**

	level	id	x	y	width	height	class
1	l0	enclosure	0	0	5760	2160	obstacle_box
2	l0	1	150	200	100	100	obstacle_box
3	l0	2	350	200	100	100	obstacle_box
4	l0	3	550	200	100	100	obstacle_box
5	l0	4	750	200	100	100	obstacle_box
6	l0	5	950	200	100	100	obstacle_box
7	l0	6	1150	200	100	100	obstacle_box
8	l0	7	1350	200	100	100	obstacle_box
9	l0	8	1550	200	100	100	obstacle_box
10	l0	9	1750	200	100	100	obstacle_box
11	l0	10	1950	200	100	100	obstacle_box
12	l0	11	2150	200	100	100	obstacle_box
13	l0	12	2350	200	100	100	obstacle_box
14	l0	13	2550	200	100	100	obstacle_box
15	l0	14	2750	200	100	100	obstacle_box
16	l1	enclosure	0	0	7680	4320	obstacle_box
17	l1	1	3527	1382	297	147	obstacle_box

Figure 8.1

User accounts – hashing

- According to the Data Protection Act, passwords should be kept secure.
- As a result, the user's password should be hashed.
- I will make my own hashing algorithm to achieve this.
- However, because JavaScript can only work with numbers up $2^{54} - 1$, I will import a library that can deal with larger numbers.
- The library is from <https://www.npmjs.com/package/big-integer> and is available for use under creative commons license. I will be giving credit to the author in the credits screen later on.

User accounts – hashing

```
<!--link to a library of JavaScript functions to deal with numbers too big for normal JavaScript-->  
<script src="assets/external/bigInt.js" id="bigInt_external"></script>
```

Imports a library to deal with large numbers

```
function hash(a, b, modulo) { //hashes number a with effector b and returns the modulo  
    var usern = 0,  
        passw = 0,  
        hash, i;  
    for (i = 0; i < a.length; i += 1) {  
        if ((a[i].charCodeAt() > 47 && a[i].charCodeAt() < 58) || (a[i].charCodeAt() > 64 && a[i].charCodeAt() < 91) ||  
            (a[i].charCodeAt() > 96 && a[i].charCodeAt() < 123)) { //checks if the character is a-z, A-Z, 0-9  
            usern += Math.pow(a[i].charCodeAt(), 3); //processes the username 1 character at a time  
        } else {  
            alertmsg("Username can only contain upper/lower case letters and numbers");  
            return null; //prevents further script execution  
        }  
    }  
    for (i = b.length - 1; i > 0; i -= 1) {  
        if ((b[i].charCodeAt() > 47 && b[i].charCodeAt() < 58) || (b[i].charCodeAt() > 64 && b[i].charCodeAt() < 91) ||  
            (b[i].charCodeAt() > 96 && b[i].charCodeAt() < 123)) {  
            passw += Math.pow(b[i].charCodeAt(), 2);  
        } else {  
            alertmsg("Password can only contain upper/lower case letters and numbers");  
            return null;  
        }  
    }  
    hash = bigInt(parseInt(usern)).modPow(parseInt(passw), parseInt(modulo)).value; //uses a library to do usern to the power of passw  
    and then applies the modulo  
    return hash;
```

Runs once per character in username

Runs once per character in password

Applies a modulo function to the power

The bigint library requires either string or integer input, but returns arrays, so I have had to use polymorphism to make results usable (see green highlighting)

Outputs the result

User accounts – hashing testing

```
hash("administrationacc", "epsilondeltaomega", "9007199254740991");
3718274321299354
hash("username", "password", "9007199254740991");
6911995805890383
hash("value1", "value2", "9007199254740991");
4231493888424613
hash("value2", "value1", "9007199254740991");
5572803985416926
hash("value", "value", "9007199254740991");
5563504410029235
hash("value", "value", "9007199254740991");
5563504410029235
hash("value1", "value", "9007199254740991");
1253346850542243
hash("value", "value1", "9007199254740991");
8317648745696379
```

These are the results of testing that the hashing function works, and is consistent. As can be seen, swapping the username and password results in a different output to what they would normally be. I will use the first hash for the default admin password that is generated at the program's start up.

```
setTimeout(function () {
  createsavefile("3718274321299354", "ADMIN", true, false);
  window.setTimeout(function () {
```

User accounts – login form

```
document.getElementById("newacc_button").addEventListener("click", function () {
    document.getElementById("login_box").classList.add("login_cl2");
    document.getElementById("registration_box").classList.add("registration_cl2");
});
document.getElementById("oldacc_button").addEventListener("click", function () {
    document.getElementById("registration_box").disabled = "disabled";
    document.getElementById("login_box").classList.remove("login_cl2");
    document.getElementById("registration_box").classList.remove("registration_cl2");
});
```

Transitions between windows on button click

As can be seen in the code, a disabled property is set for the login form when in the registration form, however this has not worked so a different method will be needed. This will be added at a later time as it is not a priority.

User accounts – login form

```
document.getElementById("login_button").addEventListener("click", function () {
    var user = document.getElementById("username_login"),
        pass = document.getElementById("password_login"),
        hashed = hash(user.value, pass.value, "9007199254740991"),
        i, n;
    for (i = 0; i < logindetails.length; i += 1) {
        if (logindetails[i].id == hashed) {
            loggedin_user = logindetails[i];
            loggedin();
            loggedin_user.username = user.value;
            document.getElementById("login_form").reset();
            levelshown = -1;
            window.setTimeout(function () {
                arrowr();
            }, 200);
            break;
        } else if (i == logindetails.length - 1) {
            alert("fail");
        }
    }
});

function loggedin() {
    initialiseloading();
    window.setTimeout(function () {
        document.getElementById("login_registration_window").classList.add("hide");
        document.getElementById("level_selector").classList.remove("hide");
        clearloading();
    }, 800);
}
```

Hashes the user's input for username and password with a modulo

Iterates for each available user

Linear search to try and find the user in the logindetails array

Clears the input form

Redraws the level selection screen

Alerts the user that the login failed

Updates the display

User accounts – login form

```
document.getElementById("registration_button").addEventListener("click", function () {
    var user = document.getElementById("username_registration").value,
        pass1 = document.getElementById("password_registration").value,
        pass2 = document.getElementById("repassword_registration").value,
        uname = document.getElementById("name_registration").value;
    if (pass1 == pass2 && user.length > 5 && pass1.length > 5) {
        var hashed = hash(user, pass1, "9007199254740991"),
            n = 0,
            i;
        if (hashed != null) {
            for (i = 0; i < logindetails.length; i += 1) {
                if (logindetails[i].id == hashed) {
                    alert("Username already taken");
                } else {
                    n += 1;
                }
            }
            if (n == logindetails.length) {
                createsavefile(hashed, uname, false, true);
                document.getElementById("registration_form").reset();
            }
        } else {
            if (pass1 != pass2) {
                alert("Passwords not matching");
            } else if (pass1.length <= 5) {
                alert("Password must be at least 6 characters long");
            } else if (user.length <= 5) {
                alert("Username must be at least 6 characters long");
            }
        }
    });
});
```

Gets the user input fields

Checks the passwords match and the username ad password are both 6 or more characters long

Hashes the user fields

Checks that the hash hasn't failed

Iteratively performs a linear search to check account doesn't already exist

Alerts the user that the creation failed

Creates the save and clears the form

Alerts the user that the creation failed and gives a reason

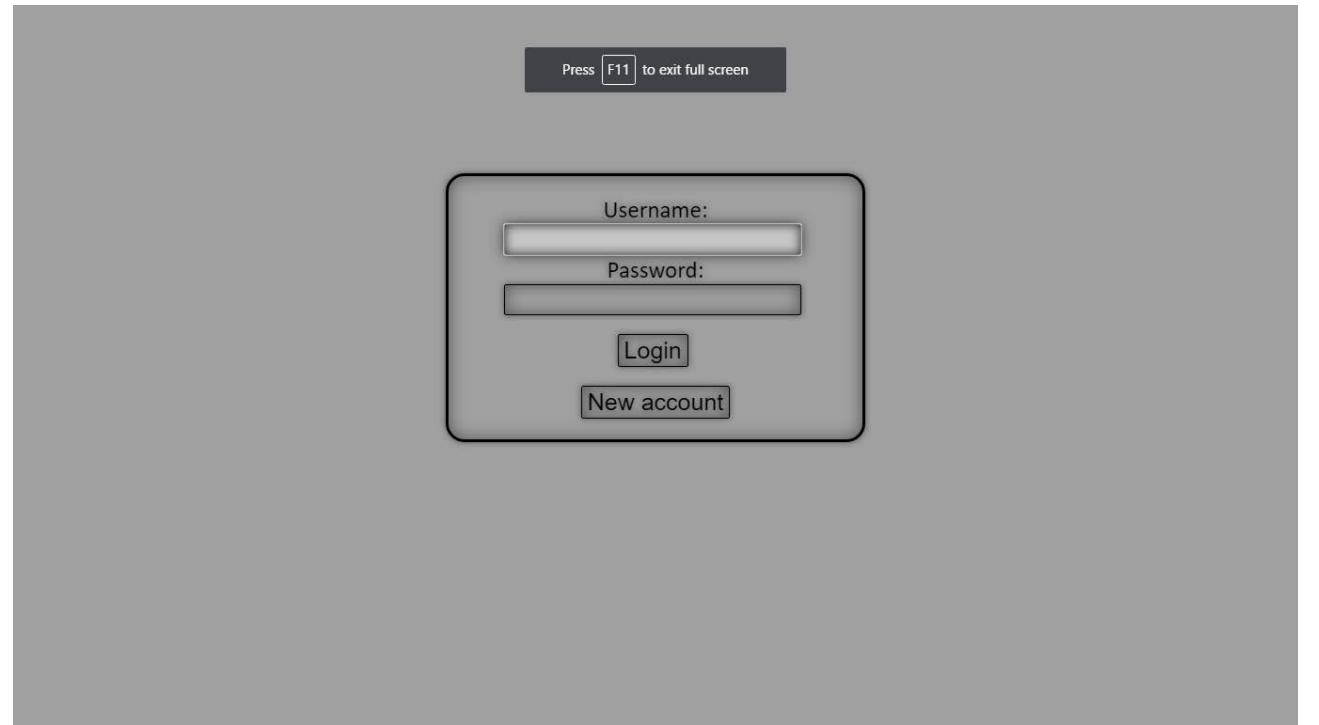
User accounts – login form GUI



User accounts

A major bug has come up in that it is not possible to type inside the text fields. This is most likely because of the keys being set up to not run their action, as they are logged for the game. To fix this, I am introducing a new variable called “play” which will be set to true when playing, and the key bindings will only be stopped then rather than at all times.

```
window.addEventListener("keydown", function (e) {
  if (!((e.keyCode || e.which) > 111) && ((e.keyCode || e.which) < 124)) {
    keystate[e.keyCode || e.which] = true;
    if (play) {
      e.preventDefault();
    }
  }
}, true);
```



User accounts – login form (GUI)

To ensure that passwords cannot be seen, text in the password fields are displayed as dots, meaning that a user can see the length of their password but not the text itself. This is more secure than having it displayed as plain text like the username and player name fields, as no one else can see the password either.

Username:
sometexthere

Password:
.....

Username:
sometexthere

Password:
.....

Retype password:
.....

Player name:
sometexthere

User accounts – testing



User accounts – testing

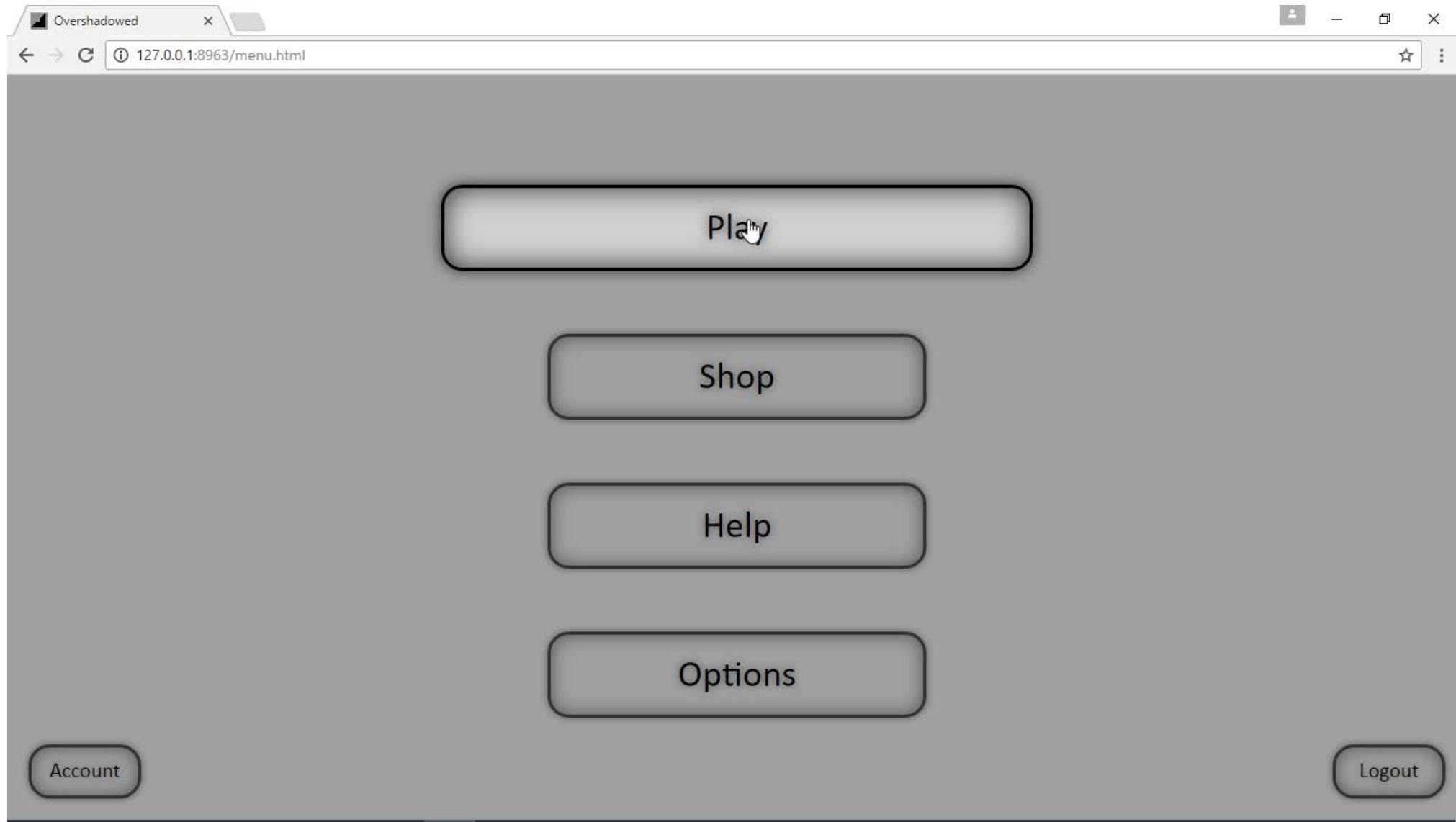


Main menu

- Almost all games have a main menu where the user can select various options (such as to logout, play the game or exit).
- To increase game immersion, I feel that this should be an intermediate step between the login screen and the level selection screen.
- The main menu should have a help button to inform new users on how to play the game.

Main menu GUI

The login button now redirects to this screen (the main menu). Currently, the only other functioning button is the play button which goes to the level selection screen.



Main menu

```
document.getElementById("goto_level_selector").addEventListener("click", function () {  
    initialiseloading();  
    window.setTimeout(function () {  
        document.getElementById("menu_selector").classList.add("hide");  
        document.getElementById("level_selector").classList.remove("hide");  
        clearloading();  
    }, 800);  
});
```

Updates the display to show the level selector
There is a similar process for the back button, so the script has not been shown
Iterates through the SQL query output

```
document.getElementById("logout_button").addEventListener("click", function () {  
    initialiseloading();  
    for (var i = 0; i < logindetails.length; i += 1) {  
        if (logindetails[i].id == loggedin_user.id) {  
            delete logindetails[i].username; //removes the username key for the previously logged in user  
            loggedin_user = {}; //clears the logged in user details  
            break; //stops the loop  
        }  
    }  
    window.setTimeout(function () {  
        document.getElementById("login_registration_window").classList.remove("hide");  
        document.getElementById("menu_selector").classList.add("hide");  
        clearloading();  
    }, 800);  
});
```

Iterates for each user in the array
Updates the display to the login screen

Main menu – testing

From this point onwards, I will mainly be testing with the user “ADMIN” which has username “administrationacc” and password “epsilondeltaomega”. This is the main admin account that is generated on game load.



User times

- As the main aim of the game is to compete for record times to complete a level, it is necessary to save and load times.
- The table itself has already been made and functions correctly, but is not read or written to.
- There needs to be an in-game timer to display how the user is doing.

User times

```
timer = document.getElementById("timer_text"),
time_begin = 0,
time_end = 0;

play = true;
time_begin = Date.now();

low.setInterval(function () {
if (play) {
    time_end = Date.now();
    var time_final = time_end - time_begin;
    timer.innerHTML = time_final / 1000;
    if (keystate[87]) { //w

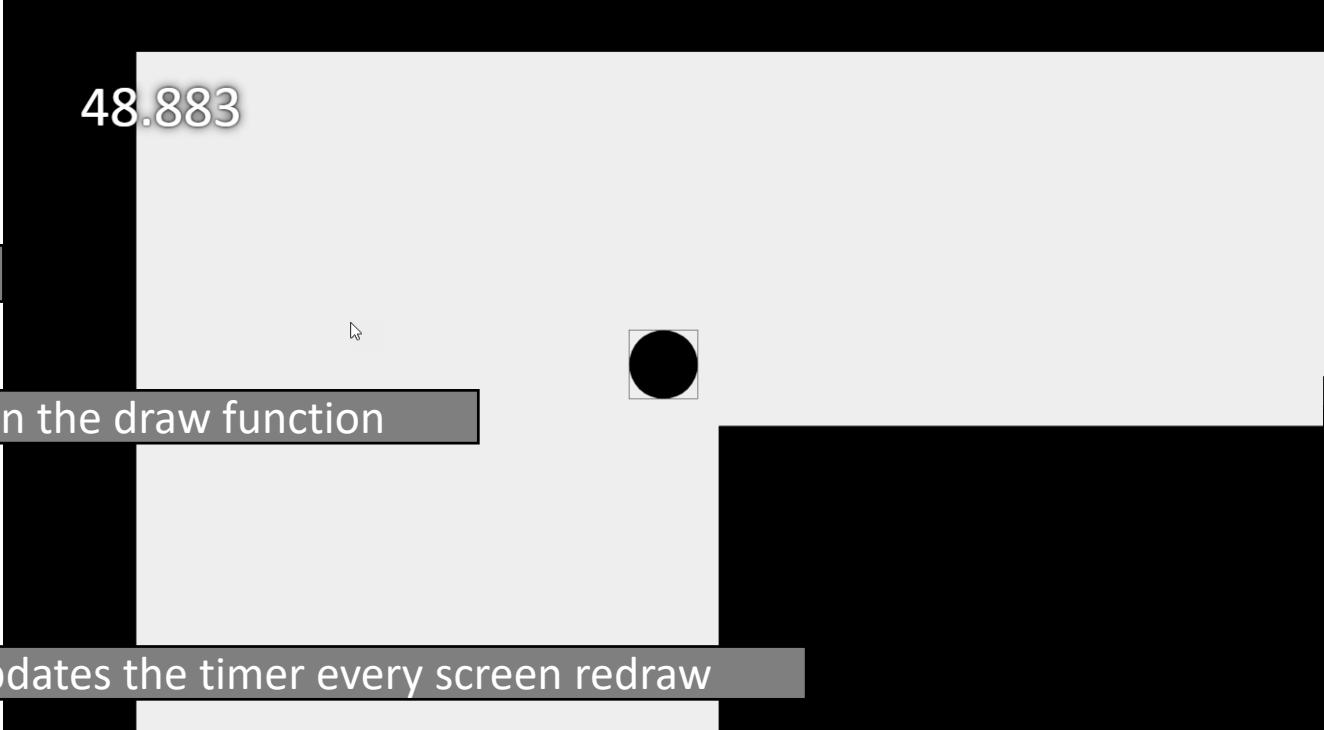
} else if (collisions[i].id == "rexit") { initialiseloading();
    play = false;
    time_end = Date.now();
    var time_final = time_end - time_begin;
    timer.innerHTML = time_final / 1000;
    window.setTimeout(function () {
```

Variable setup

Sets the start time in the draw function

Updates the timer every screen redraw

Stops the timer when the level is completed



48.883

User times

```
function getuserstimes(id) {  
    var output = [];  
    db.transaction(function (tx) {  
        var str = "pid",  
            str2 = [],  
            n;  
        for (n = 0; n < levels.length; n += 1) {  
            str += ", l" + levels[n][0].id + "time";  
            str2.push("l" + levels[n][0].id + "time");  
        }  
        tx.executeSql('SELECT ' + str + ' FROM SAVEPROGRESS WHERE pid=' + id + "", [], function (tx, results) {  
            var o;  
            for (o in results.rows[0]) {  
                output.push(results.rows[0][o]);  
            }  
            output.splice(0, 1);  
        }, null);  
    });  
    return output;  
}
```

Iterates for each level in the array

Strings for each level time

Selects the user's times

Returns the times as an ordered array

User times

```
function getrecordtimes() {
    var output = [],
        output_new = [];
    db.transaction(function (tx) {
        var str = "pid, name",
            n;
        for (n = 0; n < levels.length; n += 1) {
            str += ", l" + levels[n][0].id + "time";
        }
        tx.executeSql('SELECT ' + str + ' FROM SAVEPROGRESS INNER JOIN SAVEFILES WHERE pid=id', [], function (tx, results) {
            var i, z;
            for (i = 0; i < results.rows.length; i += 1) {
                output.push([]);
                var o;
                for (o in results.rows[i]) {
                    output[i].push(results.rows[i][o]);
                }
                //           output[i].splice(0, 1);
            }
            for (z = 2; z < output[0].length; z += 1) {
                var y = 0,
                    max = {};
                max.num = output[y][z];
                if (Number(max.num) == 0 || Number(max.num) == undefined || max.num == "") {
                    max.num = Infinity;
                }
                max.user = output[0][0];
                for (y = 1; y < output.length; y += 1) {
                    if (Number(output[y][z]) < Number(max.num) && (Number(output[y][z]) == 0 || Number(output[y][z]) == undefined || output[y][z] == "") === false) {
                        max.num = output[y][z];
                        max.user = output[y][0];
                        max.name = output[y][1];
                    }
                }
                output_new.push(max);
            }
        }, null);
    });
    return output_new;
}
```

Iterates for each level in the array

String for each level time

Selects all save file's times

Iterates through the SQL query output

Iterates through the output array

Sorts through the values to get the lowest

Sets the value to infinity if none exists

Iterates through the output array

Overwrites time if new one is smaller

Returns the array

User times

A bug can be observed where the record time has “(undefined)” in it. This is because my SQL query does not specify that the name value needs to be taken from the correct table, so does not exist.



User times

```
D.transaction(function (tx) {
  var str = "pid, SAVEFILES.name",
    n:
```

The bug was not fixed by updating the SQL statement. This is because the statement was working fine, but I had forgotten to set the name initially. If there were 2 users then this would not have mattered, which will always be the case when the game is distributed as their will be the main ADMIN account, as well as a non-admin user generated account. All admin accounts will not have times saved to them, so won't be able to get records, which would fix this bug. However, I simply added in the line below to fix the program for 1 user testing purposes.

```
max.name = output[0][1];
```

User times

Checks if the new time is a record

```
timer.innerHTML = time_final / 1000;
if ((time_final < current_times[mod(levelshown, levelsdata.length)]) || (current_times[mod(levelshown, levelsdata.length)] == ""))
{
    updateusertime(loggedin_user.id, String("l" + mod(levelshown, levelsdata.length)), time_final); —————— Updates the user's times
    current_times.splice(mod(levelshown, levelsdata.length), 1, time_final);
    alert("New personal record set in " + (time_final / 1000) + " seconds"); —————— Tells the user they set a new personal best
    maindisplayownrecord.innerHTML = "Your record: " + time_final / 1000; —————— Updates the level selection record
    window.setTimeout(function () {
        record_times = getrecordtimes(); —————— Gets the new record times
        window.setTimeout(function () {
            var time = Number(record_times[mod(levelshown, levelsdata.length)].num);
            if (typeof time == "number" && time != 0 && time != Infinity) {
                maindisplayotherrecord.innerHTML = "All time record: " + (time / 1000) + " (" +
                record_times[mod(levelshown, levelsdata.length)].name + ")";
            }
            else {
                maindisplayotherrecord.innerHTML = "----";
            }
        }, 200);
    }, 200);
}
window.setTimeout(function () {
```

Updates the user's times

Tells the user they set a new personal best

Updates the level selection record

Gets the new record times

Sets the overall record time display

```
function updateusertime(id, level, time) {
    db.transaction(function (tx) {
        tx.executeSql('UPDATE SAVEPROGRESS SET l' + level + 'time=' + time + '' WHERE pid=' + id + '');
    });
}
```

Updates the SQL table

User times – testing



Snackbar alerts

- Snackbars are small notifications that appear at the bottom of the screen for a short period of time before vanishing.
- Currently the popup method user (alert) is not very attractive or intuitive, and breaks game immersion.
- Adding snackbar notifications will prevent this immersion break.
- They will be used in places like failed login attempts and new record's set.

Snackbar alerts

```
function alertmsg(message) {
    var popup = document.createElement("div");
    popup.classList.add("popup_window");
    popup.classList.add("slideup_alert");
    var popup_inner = document.createElement("div");
    popup_inner.classList.add("popup_inner_window");
    popup_inner.classList.add("slideup_alert_inner");
    var alert_box = document.createElement("div");
    alert_box.classList.add("slideup_alert_box_show");
    alert_box.classList.add("large_glow");
    alert_box.classList.add("slideup_alert_box_hidden");
    var alert_text = document.createElement("p");
    alert_text.classList.add("slideup_alert_message");
    alert_text.innerHTML = message;
    alert_box.appendChild(alert_text);
    void alert_box.offsetWidth;
    alert_box.classList.remove("slideup_alert_box_hidden");
    popup_inner.appendChild(alert_box);
    popup.appendChild(popup_inner);
    document.getElementById("snackbars").appendChild(popup);
    setTimeout(function () {
        void popup.offsetWidth;
        var kill = document.getElementsByClassName("slideup_alert")[0];
        kill.parentNode.removeChild(kill);
    }, 4000);
}
```

Creates various styled elements

Sets the text based off the “message” parameter

Inserts into webpage

Removes after 4 seconds

1

255

Snackbar alerts – testing



Snackbar alerts – testing

A minor bug is apparent as the record time is updated from 0 to 1.804. This is because JavaScript considers 0 to be equal to "" so I have enforced an absolutely equal to check which means that the type (number or string) must be the same as the other.



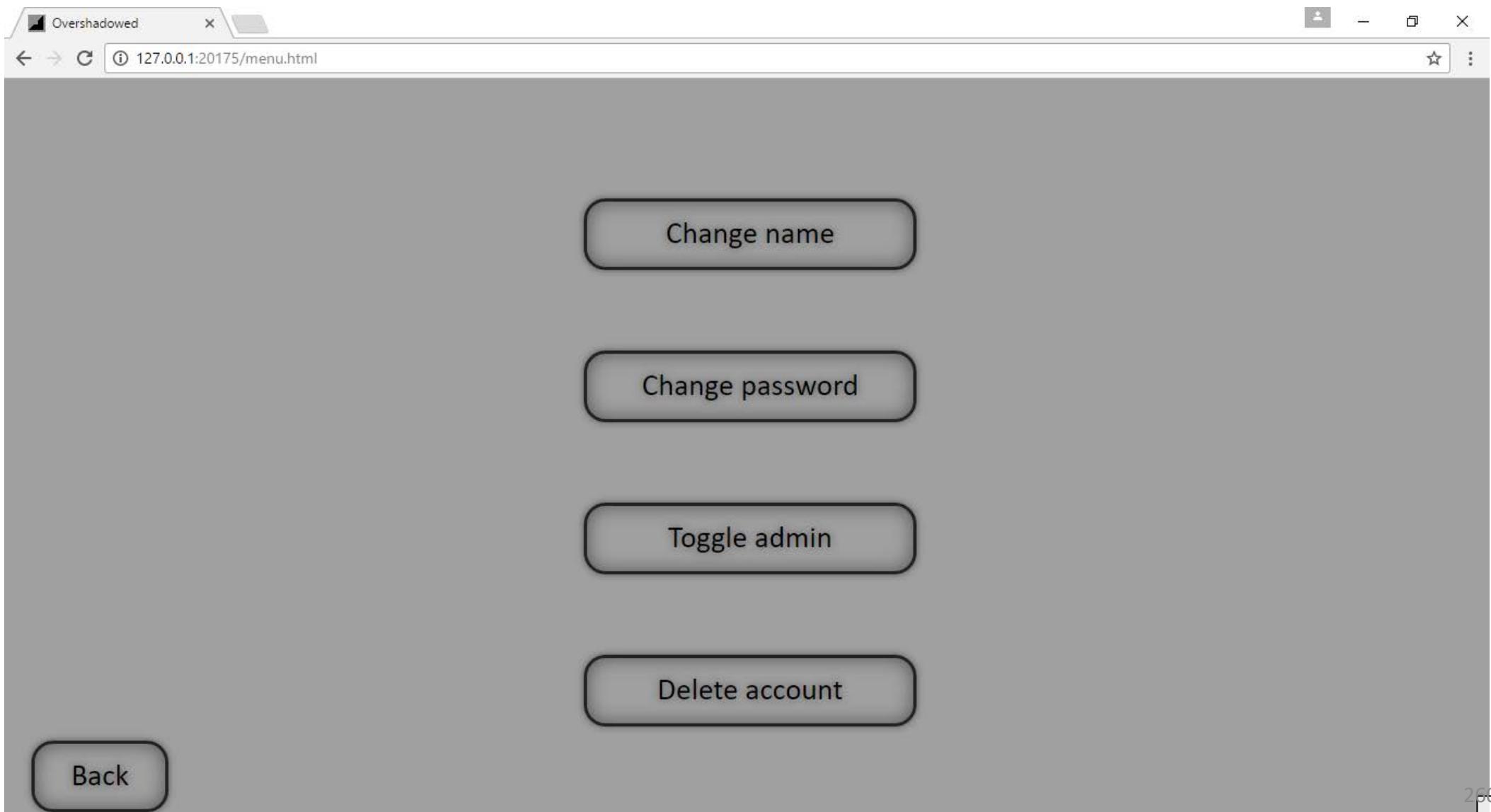
User registration with snackbars – testing



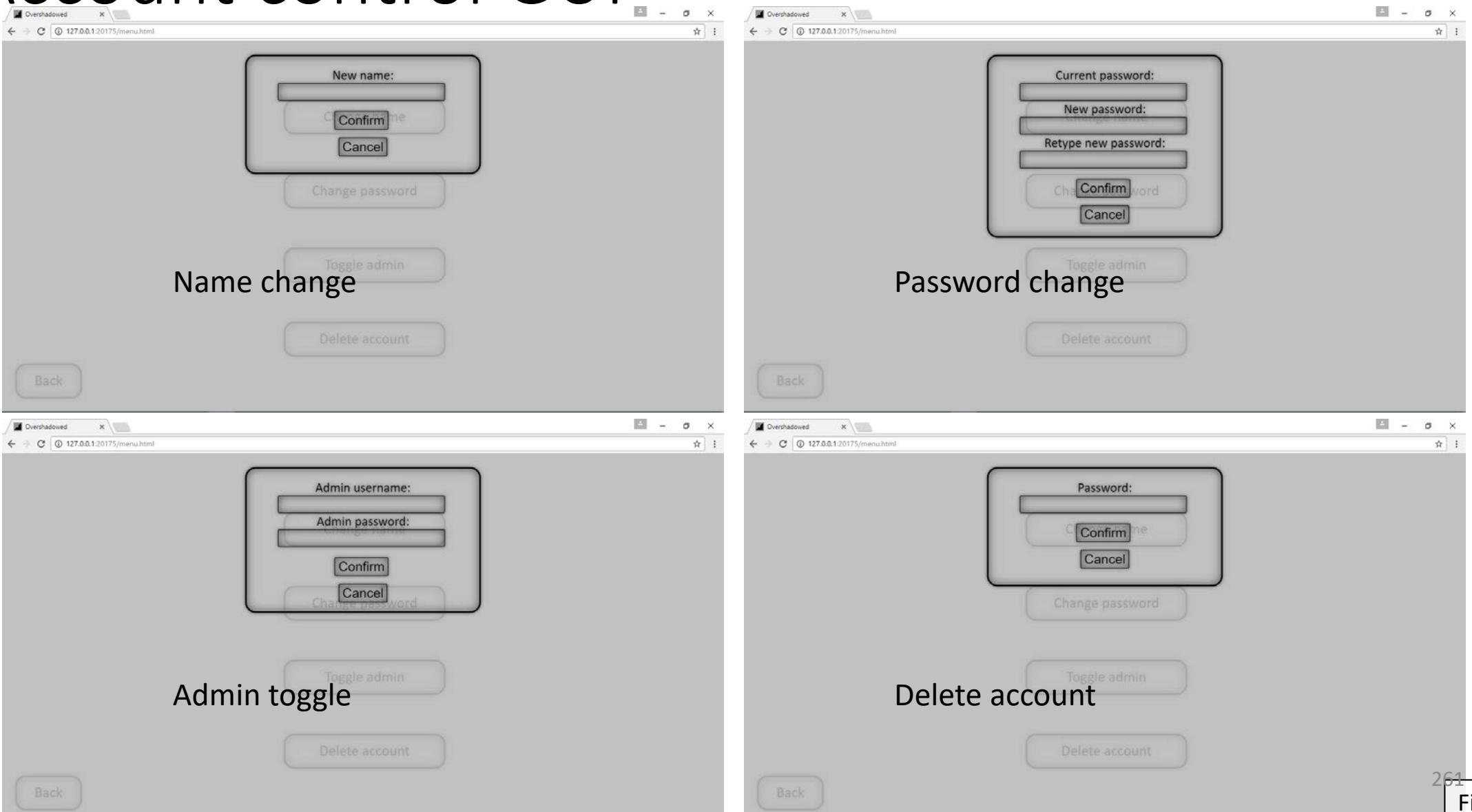
Account control

- On almost all systems with users, there is an account control screen.
- This includes options such as changing the password, making the user an admin, deleting an account and so on.
- I wish to add these features to my project, and for administrators to have the ability to change other user's settings.

Account control GUI



Account control GUI



Account control

```
document.getElementById("change_button_name").addEventListener("click", function () {  
    var user =loggedin_user.id,  
        name = document.getElementById("new_name").value;  
    if (user != 3718274321299354) {  
        changesavefilename(user, name, true);  
        loggedin_user.name = name;  
        alertmsg("New name set");  
        document.getElementById("name_change").reset();  
    } else {  
        alertmsg("Cannot edit the main ADMIN account in this way")  
    }  
});
```

Gets the user ID and name

Checks the user is not the main admin account

Updates the SQL database

Clears the form and displays a message

Warns that the admin account can't be edited

Account control

```
document.getElementById("login_button_password_change").addEventListener("click", function () {
    var user =loggedin_user.username,
        pass = document.getElementById("login_oldpassword"),
        pass1 = document.getElementById("password_change1"),
        pass2 = document.getElementById("password_change2");
    var hashed = hash(user, pass.value, "9007199254740991");
    if (pass1.value != pass2.value) {
        alertmsg("Passwords not matching");
    } else if (pass1.value.length <= 5) {
        alertmsg("Password must be at least 6 characters long");
    } else {
        for (i = 0; i < logindetails.length; i++) {
            if (logindetails[i].id == hashed) {
                hashed_new = hash(user, pass1.value, "9007199254740991");
                changesavefilepassword(hashed, hashed_new, true);
               loggedin_user.id = hashed_new;
                alertmsg("Password changed");
                document.getElementById("password_change").reset();
                break;
            } else if (i == logindetails.length - 1) {
                alertmsg("Current password failed");
            }
        }
    }
});
```

Gets the username, and original and updated passwords

Checks the passwords match

Checks the password is long enough

Iterates through the user array to find the current user

Updates the new hash to the SQL database

Alerts the user that the operation succeeded

Resets the form

Alerts the user that the operation failed

Account control

```
document.getElementById("login_button_admin").addEventListener("click", function () {  
    var user = document.getElementById("username_login_admin"),  
        pass = document.getElementById("password_login_admin");  
    var hashed = hash(user.value, pass.value, "9007199254740991");  
    if (loggedin_user.id != "3718274321299354") {  
        for (i = 0; i < logindetails.length; i++) {  
            if (logindetails[i].id == hashed) {  
                if (String(logindetails[i].admin) == "true") {  
                    var admin_name = logindetails[i].name,  
                        user_name = loggedin_user.name;  
                    if (String(loggedin_user.admin) == "false") {  
                        changesavefileadmin(loggedin_user.id, true, true);  
                        alertmsg(admin_name + " has authorised " + user_name + " to be an admin");  
                        changesavefileadmin(hashed, true, true)  
                        loggedin_user.admin = "true";  
                    } else {  
                        changesavefileadmin(loggedin_user.id, false, true);  
                        alertmsg(admin_name + " has removed " + user_name + "'s admin powers");  
                        loggedin_user.admin = "false";  
                    }  
                    document.getElementById("admin_confirm").reset();  
                } else {  
                    alertmsg("Attempted login is not an admin");  
                }  
                break;  
            } else if (i == logindetails.length - 1) {  
                alertmsg("Login failed");  
            }  
        }  
    } else {  
        alertmsg("Cannot edit the main ADMIN account in this way");  
    }  
});
```

Gets the username and password

Checks the user isn't the main admin

Iterates through the user array to find the current user

Makes the user an admin if they aren't already

Otherwise removes admin rights

Alerts user that login was not an admin

Alerts user that login was incorrect

Warns that the admin account can't be edited

Account control

```
document.getElementById("change_button_delete_account").addEventListener("click", function () {
    var user =loggedin_user.username,
        pass = document.getElementById("delete_password");
    var hashed = hash(user, pass.value, "9007199254740991");
    if (hashed != 3718274321299354) {
        for (i = 0; i < logindetails.length; i++) {
            if (logindetails[i].id == hashed) {
                var temp_mod = document.getElementById("confirm_delete_account");
                temp_mod.classList.remove("fadeoutblur");
                void temp_mod.offsetWidth;
                temp_mod.classList.add("fadeinblur");
                temp_mod.classList.remove("hide");
                break;
            } else if (i == logindetails.length - 1) {
                alertmsg("Password incorrect");
            }
        }
    } else {
        alertmsg("Cannot edit the main ADMIN account in this way");
    }
});
```

Gets the username and password

Checks the user isn't the main admin

Iterates through the user array to find the current user

Displays the confirm delete account screen

Warns that the admin account can't be edited

Warns that the admin account can't be deleted

Account control

```
document.getElementById("confirm_kill_account").addEventListener("click", function () {
    removesave(loggedin_user.id, true);
    initialiseloading();
    alertmsg("User deleted");
    document.getElementById("delete_account").reset();
    var temp_mod1 = document.getElementById("confirm_delete_account"),
        temp_mod2 = document.getElementById("login_window_delete_account"),
        temp_mod3 = document.getElementById("account_settings");
    temp_mod1.classList.remove("fadeinblur");
    void temp_mod1.offsetWidth;
    temp_mod1.classList.add("fadeoutblur");
    temp_mod2.classList.remove("fadeinblur");
    void temp_mod2.offsetWidth;
    temp_mod2.classList.add("fadeoutblur");
    temp_mod3.classList.remove("fadeinblur");
    void temp_mod3.offsetWidth;
    temp_mod3.classList.add("fadeoutblur");
    window.setTimeout(function () {
        temp_mod1.classList.add("hide");
        temp_mod2.classList.add("hide");
        temp_mod3.classList.add("hide");
    }, 500);
    window.setTimeout(function () {
        document.getElementById("login_registration_window").classList.remove("hide");
        document.getElementById("menu_selector").classList.add("hide");
        clearloading();
    }, 800);
});
```

Deletes the SQL entries for the current user

Clears the input boxes, and “logs out” the deleted user

Displays the login screen

Account control

```
function changesavefileadmin(id, admin, update) {  
    db.transaction(function (tx) {  
        tx.executeSql('UPDATE SAVEFILES SET admin=' + admin + ' WHERE id=' + id + '');  
    });  
    if (update === true) {  
        var obj = {},  
            i;  
        obj.id = loggedin_user.id;  
        obj.name = loggedin_user.name;  
        obj.admin = admin;  
        obj.bits = bits;  
        for (i = 0; i < logindetails.length; i += 1) {  
            if (logindetails[i].id == id) {  
                obj.qbits = logindetails[i].qbits;  
                obj.reset = logindetails[i].reset;  
                logindetails.splice(i, 1, obj);  
            }  
        }  
    }  
}
```

SQL query to set the admin property to the “admin” parameter of the “id” user

Updates the logindetails array (if the “update” parameter is true)

Iterates for each user in the array

Account control

```
function changesavefilepassword(olddid, newid, update) {
  db.transaction(function (tx) {
    tx.executeSql('UPDATE SAVEFILES SET id=' + newid + '' WHERE id=' + oldid + '');
  });
  if (update === true) {
    var obj = {};
    i;
    obj.id = newid;
    obj.name = loggedin_user.name;
    obj.admin = loggedin_user.admin;
    for (i = 0; i < logindetails.length; i += 1) {
      if (logindetails[i].id == newid) {
        obj.qbits = logindetails[i].qbits;
        obj.reset = logindetails[i].reset;
        logindetails.splice(i, 1, obj);
      }
    }
  }
}
```

Replaces the “old” user id with the new one

Updates the logindetails array (if the “update” parameter is true)

Iterates for each user in the array

Account control

```
function changesavefilename(id, name, update) {
  db.transaction(function (tx) {
    tx.executeSql('UPDATE SAVEFILES SET name=' + name + '' WHERE id=' + id + '');
  });
  if (update === true) {
    var obj = {},
      i;
    obj.id = id;
    obj.name = name;
    obj.admin = loggedin_user.admin;
    for (i = 0; i < logindetails.length; i += 1) {
      if (logindetails[i].id == id) {
        obj.qbits = logindetails[i].qbits;
        obj.reset = logindetails[i].reset;
        logindetails.splice(i, 1, obj);
      }
    }
  }
}
```

Replaces the name field with the new chosen one

Updates the logindetails array (if the “update” parameter is true)

Iterates for each user in the array

Account control

```
function removesave(id, update) {
  db.transaction(function (tx) {
    tx.executeSql('DELETE FROM SAVEFILES WHERE id="' + id + '"');
    tx.executeSql('DELETE FROM SAVEPROGRESS WHERE pid="' + id + '"');
  });
  if (update === true) {
    var i;
    for (i = 0; i < logindetails.length; i += 1) {
      if (logindetails[i].id == id) {
        logindetails.splice(i, 1);
      }
    }
  }
}
```

Removes the row from each table

Updates the logindetails array (if the “update” parameter is true)

Iterates for each user in the array

Account control – testing



271

Figure 13.3

Countdown timer

- Currently the level timer begins when the level loads rather than when the loading screen has vanished.
- To rectify this, I am adding a 3 second count down which gives the player time to adjust to the level having only just appeared when they start.
- During this time, the player will be unable to move, and will simply have to wait for the countdown to finish.

Countdown timer

```
function countdown(reset) {
    document.getElementById("countdown").classList.remove("fadeout"); ————— Makes the countdown visible
    void document.getElementById("countdown").offsetWidth;
    document.getElementById("countdown").classList.remove("hide");
    (function myLoop(i) {
        document.getElementById("countowntext" + i).classList.remove("countdown_text_in");
        document.getElementById("countowntext" + i).classList.remove("countdown_text_out");
        void document.getElementById("countowntext" + i).offsetWidth;
        document.getElementById("countowntext" + i).classList.add("countdown_text_in");
        window.setTimeout(function () {
            document.getElementById("countowntext" + i).classList.add("countdown_text_out");
            if (--i) myLoop(i);
            else {
                timer.classList.remove("hide"); ————— Makes the timer visible
                timer.classList.add("fadein");
                document.getElementById("countdown").classList.add("fadeout");
                if (reset === true) {
                    time_begin = Date.now(); ————— Resets the timer based on a parameter
                }
                play = true;
                window.setTimeout(function () {
                    document.getElementById("countdown").classList.add("hide"); ————— Makes the countdown not visible
                }, 500);
            }
        }, 1000, [i])
    })(3);
}
```

Makes the countdown visible

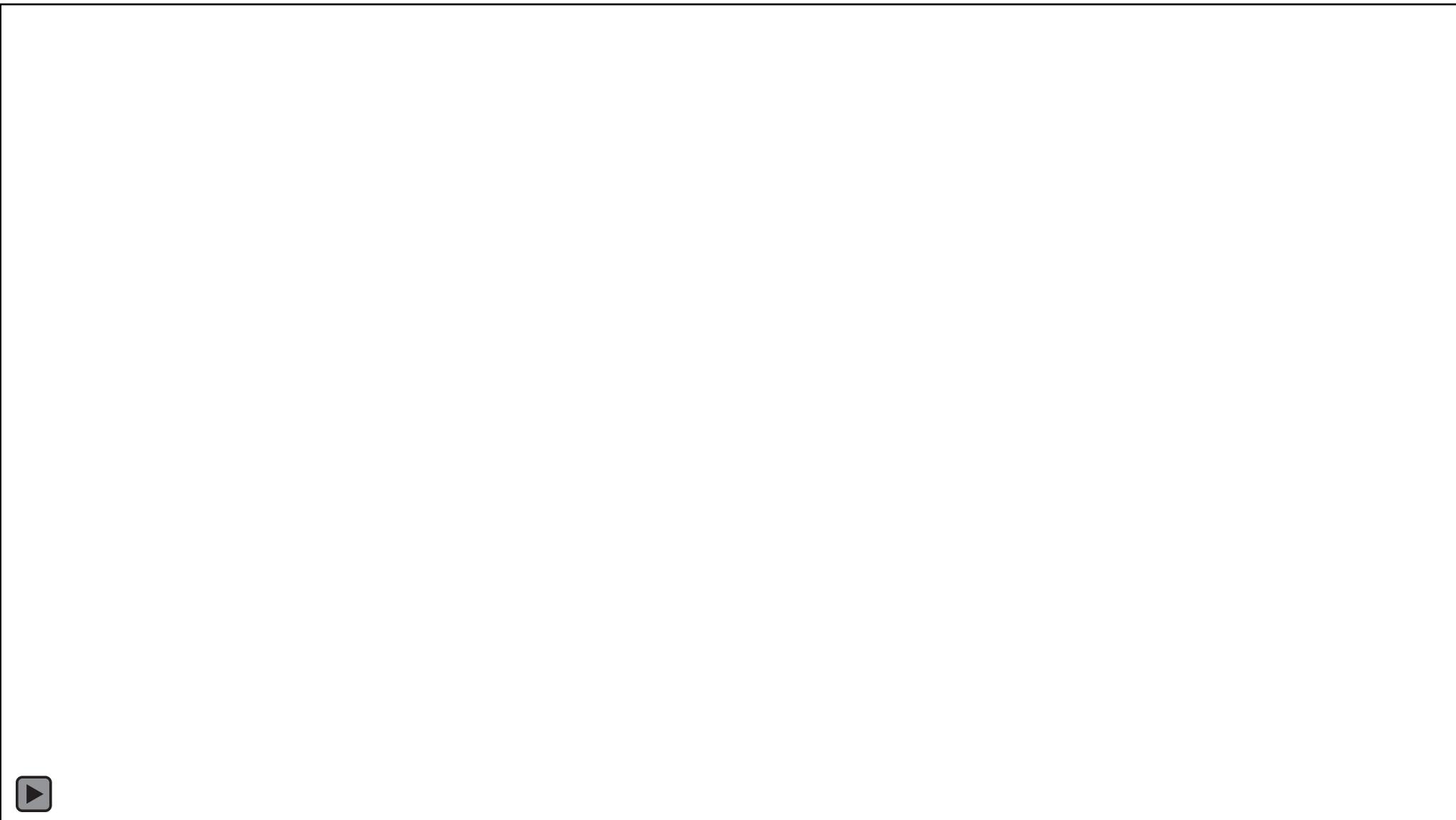
A loop with a 1 second delay between each iteration, that shows each number for the countdown at a time

Makes the timer visible

Resets the timer based on a parameter

Makes the countdown not visible

Countdown timer – testing



Lethal obstacles

- To add a greater challenge to the maze, I am adding in walls that (when collided with) damage the player.
- If the player takes too much damage then they will fail the level and have to restart it.
- There will be a visual indicator on how much damage the player has taken.
- Over time, the amount of damage taken will decrease.
- On the start of each level, the player will have taken no damage.

Lethal obstacles

- The lethal obstacles will have the exact same appearance as non-lethal obstacles.
- The player will be able to use a method of detecting the obstacles that will have a cooldown delay between uses.
- This detection will cause the lethal obstacles to flash a different colour, making it clear that they should be avoided.

Lethal obstacles

```
<div id="painbox"> </div>
```

The visual health/damage indicator

```
    collisions.splice(1, 1);
} else if (collisions[i].classList.contains("lethal")) {
    health -= 50;
} else if (collisions[i].id == "rexit") {
```

Inside the checkcollision function, removes 50 health if the player touches a lethal obstacle

```
player = document.getElementById("player");
health = 100;
countdown(true);
```

Inside the draw function, sets health to 100 (max)

```
}
```

```
shadowrays();
if (health < 100) {
    health += 0.1;
}
```

Inside the main game loop

Increases the health if less than full

```
document.getElementById("painbox").style.boxShadow = "inset 0 0 " + (100 - health) * 2.5 + "px " + (100 - health) / 5 + "px red";
```

Renders the visual health/damage indicator

```
if (health < 0) {
```

```
    time_end = 0;
```

```
    play = false;
```

```
    initialiseloading();
```

```
    window.setTimeout(function () {
```

```
        clear();
```

```
        document.getElementById("level_selector").classList.remove("hide");
```

```
        timer.classList.add("hide");
```

```
        clearloading();
```

```
        health = 100;
```

```
        document.getElementById("painbox").style.boxShadow = "inset 0 0 " + (100 - health) * 2.5 + "px " + (100 - health) / 5 +
```

```
        "px red";
```

```
}, 1000);
```

If health is less than 0, then quits the level without saving the time

Shows the level selector screen

Resets the health/damage visualiser

Lethal obstacles

```
    evg.appendChild(rectangle);
    if (rectangle.classList.contains("lethal")) {
        if (document.getElementById("flethal") === null) {
            createfilter(["feGaussianBlur"], ["stdDeviation", 20], ["-600%", "-600%", "1300%", "1300%"], "lethal");
        }
        createrectangle(posx, posy, width, height, "url(#flethal)", ("l" + id), "lethal_animation_hidden");
        lethal.push("rl" + id);
    }

if(keystate[32]) { //space
    for (i = 0; i < lethal.length; i += 1) {
        document.getElementById(lethal[i]).classList.add("lethal_animation_shown");
        cooldown = 250;
        try {
            document.getElementById(lethal[0]).addEventListener("transitionend", clear_lethal, true);
        }
        catch (err) {}
    }
}

if (collisions[i].id == "renclosure" || collisions[i].id == "cplayer" || collisions[i].id == "rhb" ||
collisions[i].classList.contains("shadowray") || collisions[i].classList.contains("lethal_animation_hidden")) {
    collisions.splice(i, 1);
} else if (collisions[i].classList.contains("lethal")) {
    health -= 50;
}
```

Currently redundant

Inside the createrectangle function, now creates a second rectangle for visual effect

Creates a filter if none exists that blurs items with a large clipping box

Creates the rectangle and appends the id to an array

Inside the main loop, checks if the space key is pressed

Iterates through the lethal obstacles array

Adds an animated effect

Tries to remove the effect when complete

Ignores collisions with the visual element

Lethal obstacles

```
function clear_lethal() {  
    for (i = 0; i < lethal.length; i += 1) {  
        document.getElementById(lethal[i]).classList.remove("lethal_animation_shown");  
    }  
}
```

Iterates for each lethal obstacle in the array

Removes all of the pulse effects from the lethal elements

At this time I will not be adding in a timer for the cooldown, as I am yet to decide a few other game mechanics which could cause this to change. The timer effect will, however, be added at a later time.

Help screen

- The main menu has several buttons in it that don't have currently have any function.
- The help button should be made to carry out its intended use of letting the player know how to use the program as something of a priority, as it explains the project to a general user.
- The help button will cause a submenu to open with options to explain controls, objectives, the shop and accounts.
- Each submenu item will open a screen of text that the user can read to understand the game.

Help screen – testing

The show/hide scripts are the same as those used in the accounts screen so are not shown for these buttons.



Level locking

- I wish to make it so that non-admin users have to unlock levels to play them.
- This will be done by completing the previous level, resulting in the next level being unlocked.
- By default only the first level should be unlocked.
- This will allow for progressively harder levels to be unlocked over time which scale with the player's ability.

Level locking

```
function dolevellocking() {
    if (String(loggedin_user.admin) == "true") {
        var i;
        for (i = 0; i < levelsdata.length; i += 1) {
            levelsdata[i].locked = false;
        }
    } else {
        db.transaction(function (tx) {
            var str = "pid",
                n;
            for (n = 0; n < levels.length; n += 1) {
                str += ", l" + levels[n][0].id + "time";
            }
            tx.executeSql('SELECT ' + str + ' FROM SAVEPROGRESS WHERE pid=' + loggedin_user.id + "", [], function (tx, results) {
                var o, times = [],
                    i, sta;
                for (o in results.rows[0]) {
                    times.push(results.rows[0][o]);
                }
                times.splice(0, 1);
                for (i = 0; i < times.length; i += 1) {
                    if (Number(times[i]) != 0) {
                        sta = false;
                        levelsdata[i].locked = false;
                    } else {
                        if (sta) {
                            levelsdata[i].locked = true;
                        } else {
                            levelsdata[i].locked = false;
                            sta = true;
                        }
                    }
                }
            }, null);
        });
    }
}
```

Unlocks all levels if the user is an admin

Iterates for each level in the array

Generates a string for each level time

Iterates for each level in the array

SQL query to get the save times for the logged in user

Iterates through the SQL query output

Turns the results into a structured array

Iterates through the users times array

Locks the levels based off whether the time is 0/"" or a non 0 value and carries the previous forwards

Level locking – testing



User settings

- In all games, there are options to change the control scheme amongst other settings (for example using the arrow keys rather than wasd).
- I wish to add these option which will be saved per user, and loaded every time they log in.
- These options will also include a method of changing the contrast and brightness, as well as sound levels (which are yet to be implemented).
- The options will be saved in a new table called savesettings, which has the player's id and each setting for the columns.

User settings

Adds the generate commands for the new table

```
var commands = ['DROP TABLE IF EXISTS LEVELS', 'DROP TABLE IF EXISTS RECTANGLES', 'DROP TABLE IF EXISTS SAVEFILES', 'DROP TABLE IF EXISTS SAVEPROGRESS', 'DROP TABLE IF EXISTS SETTINGS', 'CREATE TABLE RECTANGLES (level, id, x, y, width, height, class)', 'CREATE TABLE LEVELS (id, name, description, xspawn, yspawn, xexit, yexit, bits)', '|CREATE TABLE SAVEFILES (id, name, admin, qbits, reset)', 'CREATE TABLE SETTINGS (pid, key0, key1, key2, key3, key4, spacebar')'],
    initcommands = ['CREATE TABLE IF NOT EXISTS RECTANGLES (level, id, x, y, width, height, class)', 'CREATE TABLE IF NOT EXISTS LEVELS (id, name, description, xspawn, yspawn, xexit, yexit, bits)', 'CREATE TABLE IF NOT EXISTS SAVEFILES (id, name, admin, qbits, reset)', 'CREATE TABLE IF NOT EXISTS SETTINGS (pid, key0, key1, key2, key3, key4, spacebar')'];
```

```
tx.executeSql('INSERT INTO SAVEPROGRESS ' + SQL + ' VALUES ' + SQL2);
tx.executeSql('INSERT INTO SETTINGS (pid, key0, key1, key2, key3, key4, spacebar) VALUES ("' + id +
'"', "87", "83", "65", "68", "32", "false")');
});
```

Adds to the table when a save is generated

```
function getusersettings(id) {
    var output = [];
    db.transaction(function (tx) {
        tx.executeSql('SELECT * FROM SETTINGS WHERE pid=' + id + '', [], function (tx, results) {
            var o;
            for (o in results.rows[0]) {
                output.push(results.rows[0][o]);
            }
            output.splice(0, 1);
        }, null);
    });
    return output;
}
```

A function to return a user's saved settings

A query to get the data from the SQL table

Iterates through the SQL query output

```
function updateusersettings(id, setting, change) {
    db.transaction(function (tx) {
        tx.executeSql('UPDATE SETTINGS SET ' + setting + '="' + change + '" WHERE pid=' + id + '');
    });
}
```

A procedure to change the SQL tables

A query to get change the data in the SQL table

User settings

```
brightness.addEventListener("input", function () {
    document.body.style.filter = "brightness(" + brightness.value + "%) contrast(" + contrast.value + "%)";
}, false);
contrast.addEventListener("input", function () {
    document.body.style.filter = "brightness(" + brightness.value + "%) contrast(" + contrast.value + "%)";
}, false);
```

Updates the brightness and contrast when the sliders change

```
document.getElementById("appearance_reset_options").addEventListener("click", function () {
    brightness.value = 100;
    contrast.value = 100;
    document.body.style.filter = "brightness(" + brightness.value + "%) contrast(" + contrast.value + "%)";
});
```

Sets the brightness and contrast to their normal values (100%)

```
document.getElementById("sound_reset_options").addEventListener("click", function () {
    music_volume.value = 0.8;
    sfx_volume.value = 0.8;
});
document.getElementById("sound_mute").addEventListener("click", function () {
    music_volume.value = 0;
    sfx_volume.value = 0;
});
```

Sets the sound options to their default (has no effect as there are no sounds yet)

Sets the sound options to silent (has no effect as there are no sounds yet)

User settings

For each control key (movement), when clicked opens a key setting dialogue box

```
document.getElementById("options_0").addEventListener("click", function () {  
    set_key_popup(controls[0], "Move up", 0);  
});  
document.getElementById("options_1").addEventListener("click", function () {  
    set_key_popup(controls[1], "Move down", 1);  
});  
document.getElementById("options_2").addEventListener("click", function () {  
    set_key_popup(controls[2], "Move left", 2);  
});  
document.getElementById("options_3").addEventListener("click", function () {  
    set_key_popup(controls[3], "Move right", 3);  
});  
document.getElementById("options_4").addEventListener("click", function () {  
    set_key_popup(controls[4], "Flash", 4);  
});
```

```
function set_key_popup(key, text, option) {  
    key_changing = option;  
    var temp_mod = document.getElementById("controls_options_window_key");  
    temp_mod.classList.remove("fadeoutblur");  
    void temp_mod.offsetWidth;  
    temp_mod.classList.add("fadeinblur");  
    temp_mod.classList.remove("hide");  
    set_key_key.innerHTML = check_character_code(key);  
    set_key_text.innerHTML = text + ":";  
    alertmsg("Press any key to set a new binding");  
    var i, temp_arr = Object.keys(keystate);  
    for (i = 0; i < temp_arr.length; i++) {  
        if (keystate[temp_arr[i]]) {  
            keystate[temp_arr[i]] = false;  
        }  
    }  
    key_setting = true;
```

Displays the key setting screen

Sets the view to the current key

Iterates for each key in the array

Sets none of the keys to pressed on dialogue launch

Sets the key_setting variable to true

User settings

```
else if (key_setting) {
    var i, temp_arr = Object.keys(keystate);
    for (i = 0; i < temp_arr.length; i++) {
        if (keystate[temp_arr[i]]) {
            if (controls.includes(parseInt(temp_arr[i]))) {
                alertmsg("Key already in use");
                keystate[temp_arr[i]] = false;
            }
        } else {
            var temp_key = check_character_code(temp_arr[i]);
            alertmsg("Key set to " + temp_key);
            updateusersettings(loggedin_user.id, "key" + key_changing, temp_arr[i]);
            close_key_set_window();
            keystate[temp_arr[i]] = false;
            controls.splice(parseInt(key_changing), 1, temp_arr[i]);
            var temp_mod = document.getElementById("options_key_" + key_changing);
            temp_mod.classList.remove("fadeinblur");
            void temp_mod.offsetWidth;
            temp_mod.classList.add("fadeoutblur");
            key_setting = false;
            window.setTimeout(function () {
                temp_mod.classList.remove("fadeoutblur");
                void temp_mod.offsetWidth;
                temp_mod.classList.add("fadeinblur");
                document.getElementById("options_key_" + key_changing).innerHTML = temp_key;
                document.getElementById("help_" + key_changing).innerHTML = temp_key;
            }, 500);
        }
    }
}
```

Inside the main loop, checks if the key_setting variable is true when play is false

Iterates for each key in the array

Checks if any key is pressed

If the key is already bound to something, then it is ignored

Gets the keyboard value for the key

Tells the user the key is allowed

Updates the SQL table

Hides the key setting window

Replaces the key in controls

Updates the key in the options screen

Updates the key in the help screen

User settings

```
function close_key_set_window() {  
    var temp_mod = document.getElementById("controls_options_window_key");  
    temp_mod.classList.remove("fadeinblur");  
    void temp_mod.offsetWidth;  
    temp_mod.classList.add("fadeoutblur");  
    key_setting = false; —————— Sets the key_setting variable to false  
    window.setTimeout(function () {  
        temp_mod.classList.add("hide");  
    }, 500);  
    document.getElementById("controls_key_return_options").addEventListener("click", function () {  
        close_key_set_window();  
        alertmsg("Key set cancelled"); —————— Closes the window on a "back" button click  
    });  
}
```

Hides the key setting window

Sets the key_setting variable to false

Closes the window on a "back" button click

This should be outside of the function, as currently it will only run when the function has run, making it useless as it is not possible to cancel the key setting at the moment. The fix is shown below.

```
window.setTimeout(function () {  
    temp_mod.classList.add("hide");  
}, 500);  
document.getElementById("controls_key_return_options").addEventListener("click", function () {  
    close_key_set_window();  
    alertmsg("Key set cancelled");  
});
```

User settings

Keyboard character values are different to standard UNICODE characters, so I have had to make a function that returns the correct representation for them. Not all characters are shown as it is a long list of them.

Returns the correct symbol for each key

Checks if the character matches

Returns the character

The spacebar checks if it should be space or _

If no symbol matches then the UNICODE character is returned

```
function check_character_code(trial) {  
    if (trial == 9) { //tab  
        return ("\t");  
    } else if (trial == 8) { //backspace  
        return ("\b");  
    } else if (trial == 38) { //arrow up  
        return ("\u2191");  
    } else if (trial == 40) { //arrow down  
        return ("\u2193");  
    } else if (trial == 37) { //arrow left  
        return ("\u2190");  
    } else if (trial == 39) { //arrow right  
        return ("\u2192");  
    } else if (trial == 17) { //control  
        return ("\u0003");  
    } else if (trial == 18) { //alt  
        return ("\u0002");  
    } else if (trial == 91) { //mac command  
        return ("\u2299");  
  
    } else if (trial == 34) { //page down  
        return ("\u2193");  
    } else if (trial == 32) { //spacebar  
        if (document.getElementById("spacebar_checkbox").checked) {  
            return ("_");  
        } else {  
            return ("space");  
        }  
    } else if (trial == 220) { //backslash  
        return ("\u005c");  
    } else if (trial == 223) { //weird apostrophe  
        return ("\u0080");  
    } else if (trial == 190) { //full stop  
        return (".");  
    } else if (trial == 191) { //forward slash  
        return ("/");  
    } else {  
        return (String.fromCharCode(trial));  
    }  
}
```

User settings

Updates the options and help keys when the button is toggled

```
document.getElementById("spacebar_checkbox").addEventListener("click", function () {  
    var i;  
    for (i = 0; i < controls.length; i++) {  
        document.getElementById("options_key_" + i).innerHTML = check_character_code(controls[i]);  
        document.getElementById("help_" + i).innerHTML = check_character_code(controls[i]);  
    }  
    updateusersettings(loggedin_user.id, "spacebar", String(document.getElementById("spacebar_checkbox").checked));  
});
```

Iterates for each control in the array

Updates the SQL table so that the setting is preserved
for each save file

User settings – testing

The menu is set up the same as for the help menu, with submenus and then text/element screens.

As can be seen, changing the space/_ option does not update the controls yet, so this needs to be fixed.



User settings – testing

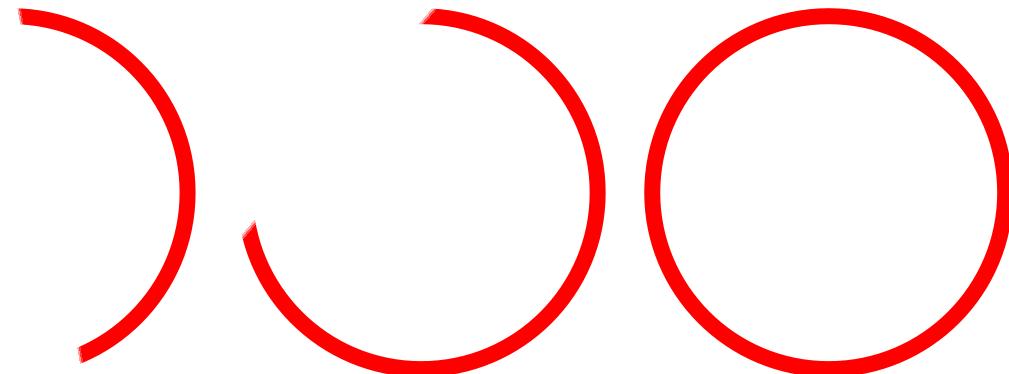
Here is a functioning version of the spacebar toggle, the bug was a mistype in the code which has now been corrected, so now works fully.

Due to not being able to see key strokes, I have not included me testing with the updated controls but they do however work as



Lethal detection

- Currently, the user can press the lethal display key (default spacebar) as frequently as they wish. This is not how I intend it to be used.
- Instead, I wish for the user to only be able to use it every 10 seconds or so, and for it to have a cooldown period in-between uses when it cannot be used.
- The cooldown will be a visual effect, which I have decided to make a circle that goes from having no outline to having a full circle outline (see below for idea).



Lethal detection

```
<svg id="svg_circle_loader" width="200" height="200" class="hide">
  <defs>
    <filter id="flash_blur" x="-50" y="-50" width="200" height="200">
      <feGaussianBlur stdDeviation="5"></feGaussianBlur>
    </filter>
  </defs>
  <circle cx="100" cy="100" r="50" id="circle_flash_glow" class="can_pause" filter="url(#flash_blur)"></circle>
  <circle cx="100" cy="100" r="50" id="circle_flash_loader" class="can_pause"></circle>
</svg>

function use_flash() {
  flash_cooldown = true;
  css_getclass(".circle_loader_load").style.animationDuration = "15s";
  var temp_mod = document.getElementById("circle_flash_glow"),
      temp_mod2 = document.getElementById("circle_flash_loader");
  temp_mod.classList.add("circle_flash_use");
  temp_mod2.classList.add("circle_loader_load");
  flash_timer = new Timer(function () {
    temp_mod.classList.remove("circle_flash_use");
    temp_mod2.classList.remove("circle_loader_load");
    flash_cooldown = false;
  }, parseInt(15000));
}

function Timer(callback, delay) {
  var timerId, start, remaining = delay;
  this.pause = function () {
    window.clearTimeout(timerId);
    remaining -= new Date() - start;
  };
  this.resume = function () {
    start = new Date();
    window.clearTimeout(timerId);
    timerId = window.setTimeout(callback, remaining);
  };
  this.resume();
}
```

A new SVG canvas that will be displayed above the main game canvas

Sets the cooldown animation to 15 seconds long

Animates the use effects

Creates a special timer for the cooldown effect

The constructor for a timer that can be paused and resumed

Pauses the timer when called

Resumes the timer when called

Starts the timer when constructed

Lethal detection

```
if (keystate[controls[4]] && !flash_cooldown) {  
    use_flash();  
    for (i = 0; i < lethal.length; i += 1) {  
        document.getElementById(lethal[i]).classList.add("lethal_animation_shown");  
        try {  
            document.getElementById(lethal[0]).addEventListener("transitionend", clear_lethal, true);  
        }  
        catch (err) {}  
    }  
  
function cssrules() {  
    var rules = {};  
    var ds = document.styleSheets,  
        dsl = ds.length;  
    for (var i = 0; i < dsl; ++i) {  
        var dsdi = ds[i].cssRules,  
            dsil = dsdi.length;  
        for (var j = 0; j < dsil; ++j) rules[dsdi[j].selectorText] = dsdi[j];  
    }  
    return rules;  
};  
  
function css_getclass(name, createifnotfound) {  
    var rules = cssrules();  
    if (!rules.hasOwnProperty(name)) throw 'todo:deal_with_notfound_case';  
    return rules[name];  
};
```

The flash use effect now checks if the cooldown has expired or not before being in action

Iterates for each lethal obstacle in the array

Gets the CSS stylesheets for the document and allows for editing of class rules (as well as element rules)

Iterates for each style sheet in the array

Iterates for each rule in the array

Returns a specific class for editing, saving time by not iterating through all elements to change their individual style

Lethal detection

```
var n;
use_flash();
for (i = 0; i < pausable.length; i += 1) {
    pausable[i].classList.add("paused");
}
try {
    flash_timer.pause();
}
catch (err) {}
window.setTimeout(function () {

document.getElementById("svg_circle_loader").classList.add("hide");|
```

When a level is played, the flash effect starts at 0

Iterates for each pausable element in the array

Attempts to pause the timer at the start

The timer is hidden when a level is finished (won/lost)

Iterates for each pausable element in the array

Attempts to resume the timer at the start after the countdown

Lethal detection – testing



Lethal detection – testing



Game pausing

- Another feature that is currently lacking is the ability to pause the game.
- Currently, if the user leaves the fullscreen mode (F11) then an overlay appears, but this does not stop the game from running.
- I believe that it should make the game pause, and then resume it upon re-entering fullscreen mode, and also trigger the countdown again.

Game pausing

```
    timer.classList.add("running");
    if (reset === true) {
        time_begin = Date.now();
        time_pause = Date.now();
    }
    else {
        time_pause -= Date.now();
        time_begin += Math.abs(time_pause);
    }
    play = true;
```

Inside the countdown function, if the countdown is not for the first time then will update the time taken to exclude the time the game was paused

```
play = false;
time_pause = Date.now();
var i;
for (i = 0; i < pausable.length; i += 1) {
    pausable[i].classList.add("paused");
}
try {
    flash_timer.pause();
}
catch (err) {}
```

Inside the not fullscreen function, pauses the timer and graphic effects, and sets the pause time

Iterates for each pausable element in the array

At some point, I will also add a pause key so that the user can choose to pause the game more easily, but this is an idea that I will come back to as it is not critical to gameplay.

Game pausing – testing



In-game currency

- To allow the player to get better times, I wish to introduce upgrades into the game.
- To get upgrades, the user must spend in-game currency.
- In-game currency will be gained through 2 methods – completing achievements and colliding with in-game collectibles.
- Each collectible can only be acquired once, and there are a set number per level.
- In-game currency will be called Qbits.

In-game currency – collectibles

```
function getRandomInt(min, max) {  
    min = Math.ceil(min);  
    max = Math.floor(max);  
    return Math.floor(Math.random() * (max - min + 1)) + min;  
}
```

This function returns a random integer in the range passed in to the function

```
createrectangle(lvl[0].xexist - 50, lvl[0].yexist - 50, "15", "15", "", "qbit" + i, "qbit");  
}  
}  
try {  
    var xpost, ypost;  
    for (i = 0; i < lvl[0].bits; i += 1) { //WORKINGON  
        xpost = getRandomInt(25, lvl[1].width-50);  
        ypost = getRandomInt(25, lvl[1].height-50);  
        createrectangle(xpost, ypost, "15", "15", "", "qbit" + i, "qbit");  
    }  
}  
catch (err) {  
    console.log("invalid");  
}
```

Generates the correct number of qbits per level inside the draw procedure (which are shaped as small squares)

Iterates for each qbit

While developing the random function, I sometimes had errors from invalid results. This prevents the program from crashing if an error occurs.

In-game currency – collectibles

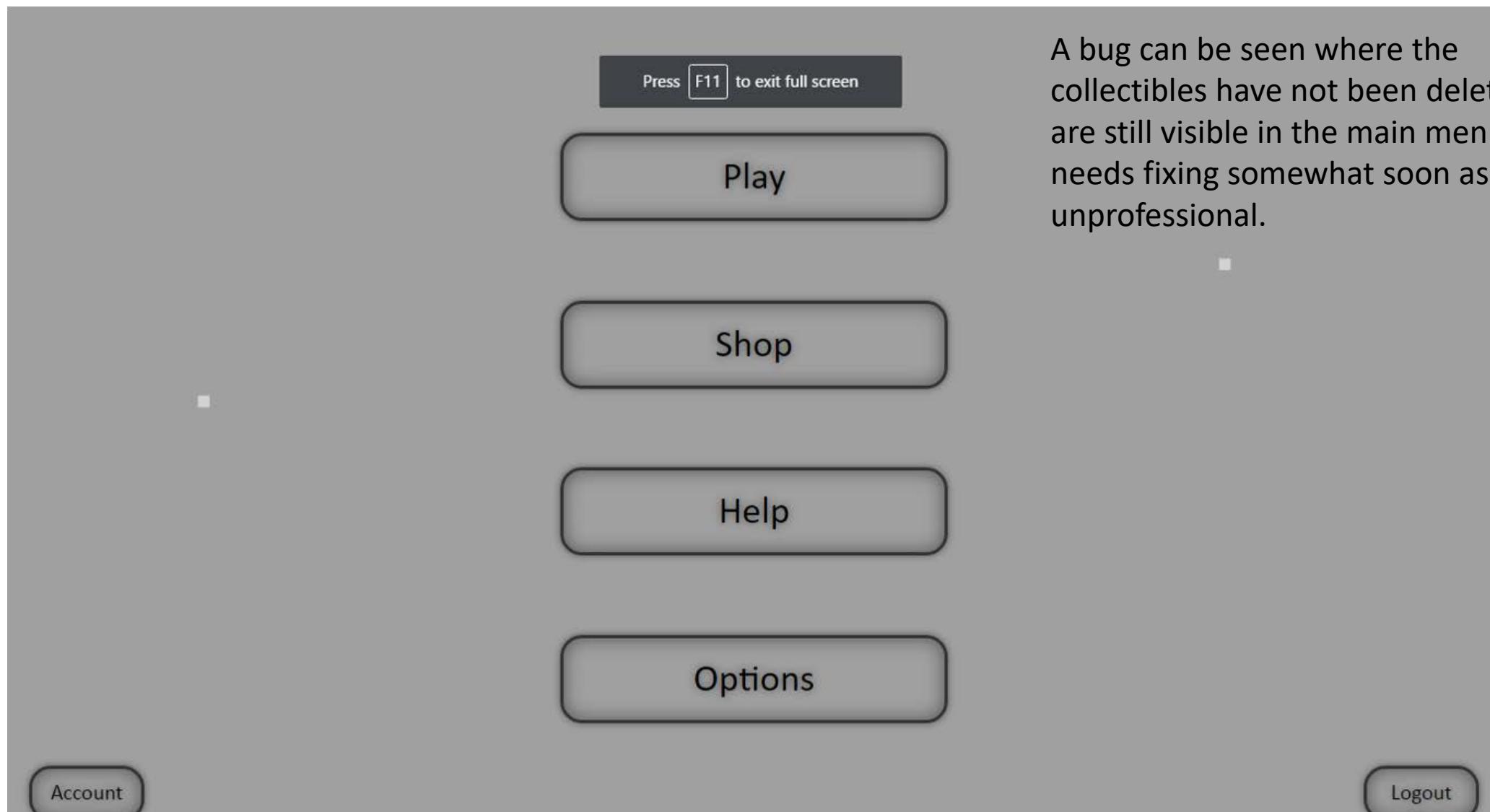
```
for (i = len - 1; i >= 0; i -= 1) {
    if (collisions[i].id == "renclosure" || collisions[i].id == ""
    || collisions[i].className.animVal == "shadowray" || collisio
    collisions[i].classList.contains("qbit")) {
        if (collisions[i].classList.contains("qbit")) {
            loggedin_user.qbits += 1;
            collisions[i].parentNode.removeChild(collisions[i]);
            console.log(collisions[i]); //WORKINGON
        }
        collisions.splice(i, 1);
    }
    else if (collisions[i].classList.contains("lethal")) {
```

Collision with a qbit, deletes the qbit and increments the qbits obtained. Also removes the element from the array

In-game currency – collectibles basic testing



In-game currency – collectibles basic testing



In-game currency – collectibles

A procedure to generate qbits, and prevent them from being spawned inside level scenery

```
function create_bit(i, lvl) {  
    var xpost, ypost, qbit_check, collisions, collisions_node, len, qbit_rect = svg.createSVGRect();  
    xpost = getRandomInt(25, lvl[1].width - 50);  
    ypost = getRandomInt(25, lvl[1].height - 50);  
    createrectangle(xpost, ypost, "15", "15", "", "qbit" + i, "qbit");  
    qbit_check = document.getElementById("rqbit" + i);  
    qbit_rect.x = qbit_check.left;  
    qbit_rect.y = qbit_check.top;  
    qbit_rect.width = qbit_check.width;  
    qbit_rect.height = qbit_check.height;  
    collisions_node = svg.getIntersectionList(qbit_rect, null);  
    len = collisions_node.length;  
    collisions = [];  
    for (i = 0; i < len; i += 1) {  
        collisions.push(collisions_node[i]);  
    }  
    for (i = len - 1; i >= 0; i -= 1) {  
        if (collisions[i].id == "renclosure" || collisions[i].id == "cplayer" || collisions[i].id == "rpain" || collisions[i].id == "rhb"  
        || collisions[i].classList.contains("shadowray") || collisions[i].classList.contains("lethal_animation_hidden") ||  
        collisions[i].classList.contains("qbit")) {  
            collisions.splice(i, 1);  
        }  
        else {  
            qbit_check.parentNode.removeChild(qbit_check);  
            create_bit(i, lvl);  
        }  
    }  
}
```

Generates a normal array from the node list generated in the intersection list

Iterates for each element in the array

Removes all non-physical objects from the array. If items still remain, then the qbit is deleted and respawned elsewhere

In-game currency – collectibles

```
record_times = getrecordtimes();
qbits = getusersbits(loggedin_user.id);
document.getElementById("login_form").reset();
levelshown = -1;
window.setTimeout(function () {
    arrowr();
    loggedin_user.qbits = 0;
    for (n = 0; n < qbits.length; n += 1) {
        loggedin_user.qbits += Number(qbits[n]);
    }
}, 200);
```

When a user logs in, their qbits are calculated based off how many they've collected

Iterates for each qbit in the array

```
for (i = 0; i < objects; i += 1) {
    qbits_lvl = Number(qbits[mod(levelshown, levelsdata.length)]);
    for (i = 1; i < objects; i += 1) {
        create_bit(i, lvl);
    }
}
```

Gets the number of qbits to generate for a given level

Iterates for each remaining qbit to be collected

Gets the number of qbits collected by the user, and negates this from how many need to be generated, then spawns them

In-game currency – collectibles

```
collisions[i].classList.contains("qbit")) {  
    if (collisions[i].classList.contains("qbit")) {  
        collisions[i].parentNode.removeChild(collisions[i]);  
        qbits_lvl += 1; //WORKINGON  
    }  
    collisions.splice(i, 1);  
}
```

On collision with a qbit, changes a different variable

Qbits are now only saved if the player has completed a level –
so won't save if they die in it

```
play = false;  
updateuserbits(loggedin_user.id, String("l" + mod(levelshown, levelsdata.length)), qbits_lvl); //WORKINGON  
loggedin_user.qbits = Number(qbits_lvl - qbits[mod(levelshown, levelsdata.length)]) + Number(loggedin_user.qbits);  
qbits[mod(levelshown, levelsdata.length)] = qbits_lvl;  
if ((time_final < current_time * (mod(levelshown, levelsdata.length) + 1)) || (current_time * (mod(levelshown, levelsdata.length) + 1) < time_final)) {  
    play = true;
```

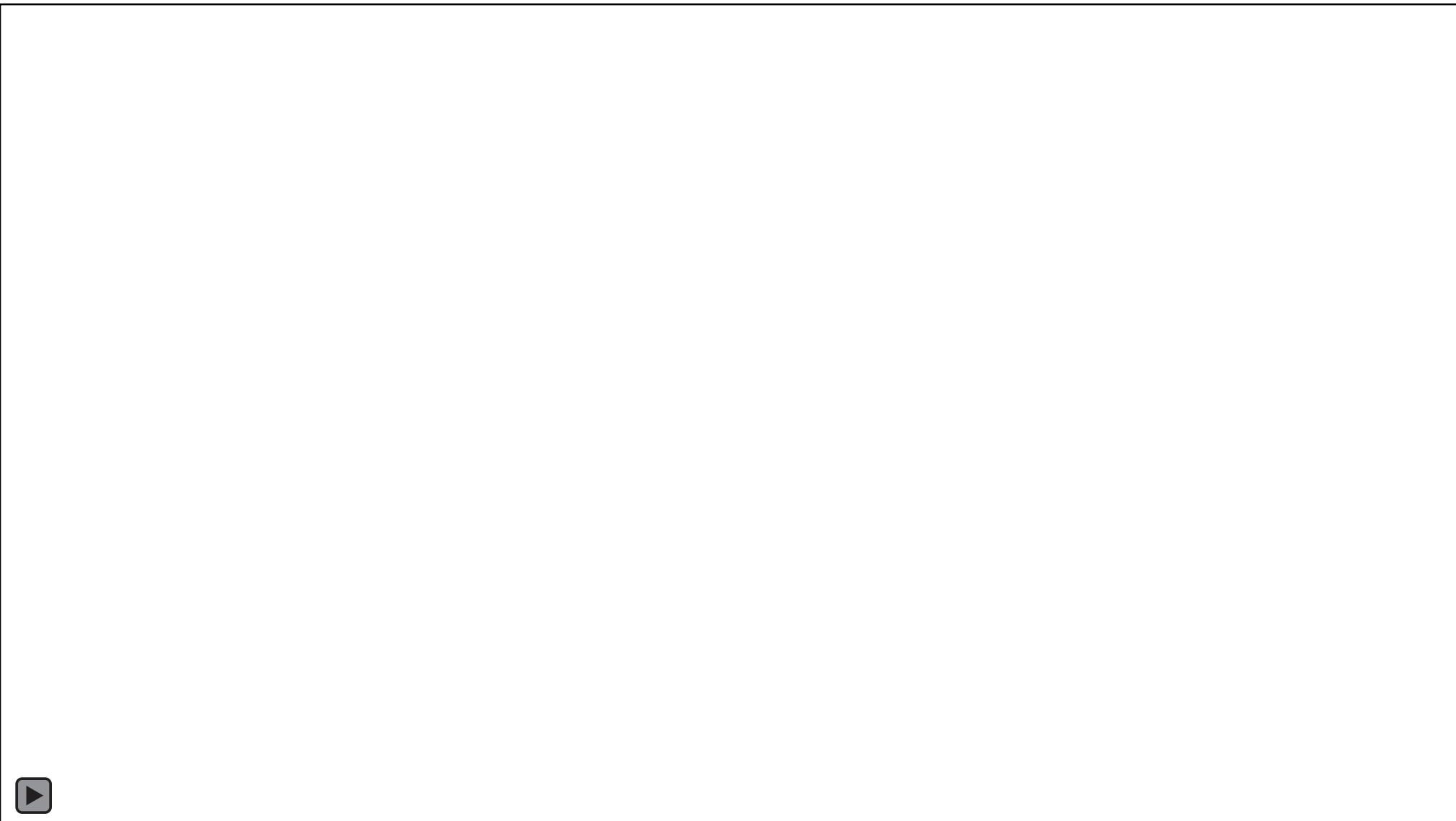
Deletes all the qbits from the canvas (preventing the
previously shown bug)

```
kill = document.getElementsByClassName("qbit");  
while (kill[0]) {  
    kill[0].parentNode.removeChild(kill[0]);  
}
```

If an element exists, then it is removed

This test includes a manual button shown in the top left which displays the collected number of Qbits for the last level played, then across all levels, then the total sum of qbits collected. It is only there for testing purposes and will not be shown to the user.

In-game currency – collectibles testing



Game immersion - SFX

- Sounds effects are a very basic form of audio included in virtually every single game on the market.
- They are used to alert the user that an event has occurred, and often imitate physics collisions.
- I wish to add sound effects (SFX) to my game.
- These SFX will be triggered by certain events such as the player colliding with a wall.

Game immersion – SFX

An array containing all the sound elements

```
sounds = { //stores the game sounds
    radar: /*the various radar ping sounds*/ [(new Audio("assets/sounds/radar_ping1.ogg")), /*the format for each sound file is the same, but the path varies*/ (new Audio("assets/sounds/radar_ping2.ogg")), (new Audio("assets/sounds/radar_ping3.ogg"))],
    hard_hurt_collide: [(new Audio("assets/sounds/slam_collision1.ogg")), (new Audio("assets/sounds/slam_collision2.ogg")), (new Audio("assets/sounds/slam_collision3.ogg"))],
    hard_collide: [(new Audio("assets/sounds/rock_impact_hard1.ogg")), (new Audio("assets/sounds/rock_impact_hard2.ogg")), (new Audio("assets/sounds/rock_impact_hard3.ogg")), (new Audio("assets/sounds/rock_impact_hard4.ogg")), (new Audio("assets/sounds/rock_impact_hard5.ogg")), (new Audio("assets/sounds/rock_impact_hard6.ogg"))],
    soft_collide: [(new Audio("assets/sounds/rock_impact_soft1.ogg")), (new Audio("assets/sounds/rock_impact_soft2.ogg")), (new Audio("assets/sounds/rock_impact_soft3.ogg"))],
    pickup_click: [(new Audio("assets/sounds/pickup1.ogg")), (new Audio("assets/sounds/pickup2.ogg")), (new Audio("assets/sounds/pickup3.ogg")), (new Audio("assets/sounds/pickup4.ogg")), (new Audio("assets/sounds/pickup5.ogg")), (new Audio("assets/sounds/pickup6.ogg")), (new Audio("assets/sounds/pickup7.ogg")), (new Audio("assets/sounds/pickup8.ogg")), (new Audio("assets/sounds/pickup9.ogg"))]
},
```

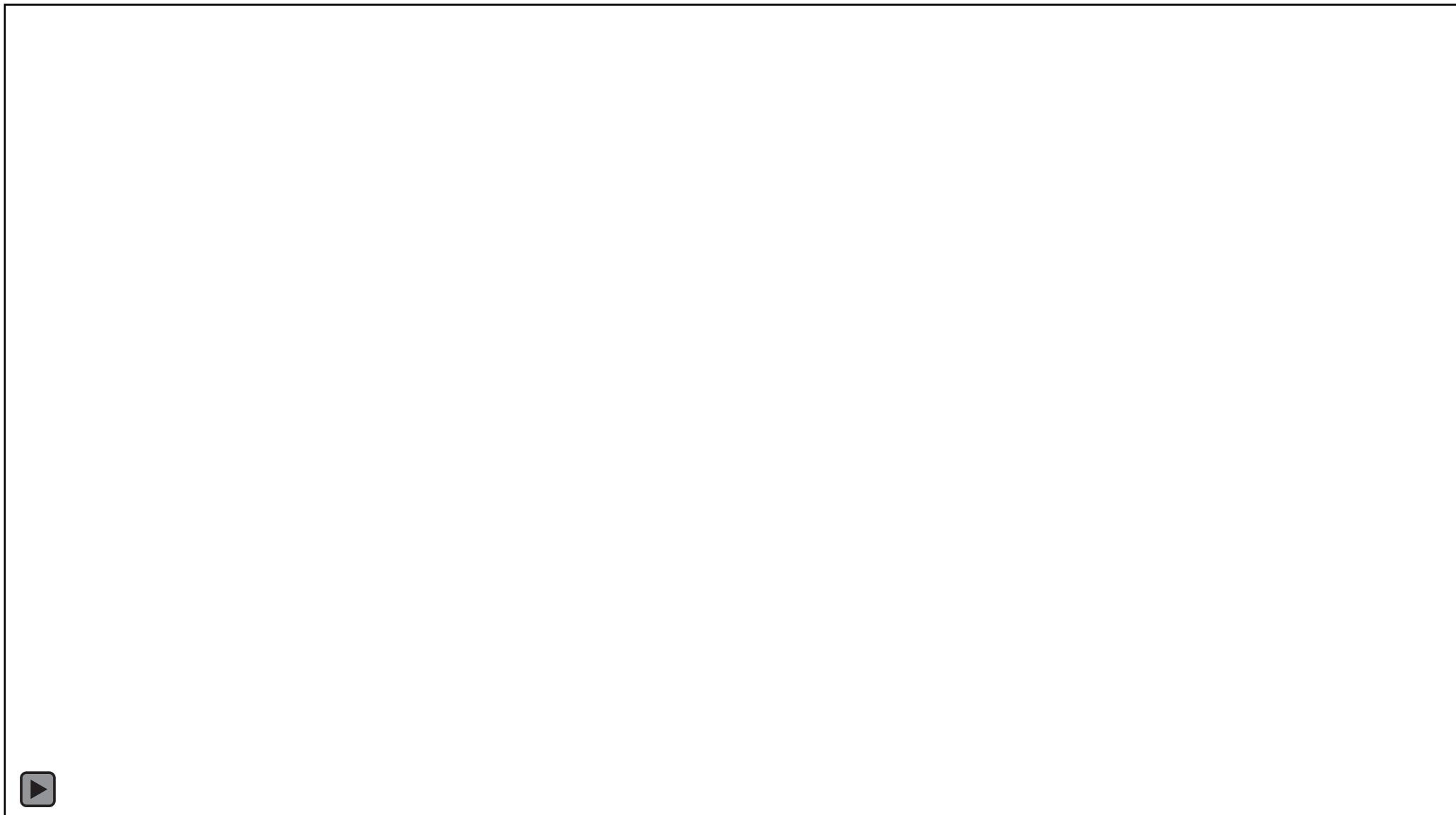
A procedure that plays the sound effect at the volume indicated by the SFX slider in the settings menu. The audio file itself is randomly selected for more variety.

```
function play_sfx(type) {
    var audioplay = sounds[type][getRandomInt(0, sounds[type].length - 1)].cloneNode();
    audioplay.volume = sfx_volume.value;
    audioplay.play();
}
```

```
comp_main.addEventListener("onstage_update", update_stage);
play_sfx("radar");
flash += timer - now; //timer function / \ f
```

An example of the usage, plays the radar sound effect when the flash module is used

Game immersion – SFX testing (has audio)



Game immersion – ambience

- Another common part of most games is ambience. It is often used to set an atmosphere for a specific scene or location.
- Common ambience effects include weather, running water and wind.
- I believe that a windy ambience will heighten the adventure aspect of the game, as it will make the levels seem more abandoned.
- The ambience effect should only be audible during game play, and not in the menus.
- The ambience sound should fade in and out smoothly, and not having a jarring cut on or off

Game immersion – ambience

```
function ambience_change() {  
    var time_play = getRandomInt(3, 30),  
        time_wait = getRandomInt(1, time_play - 2);  
    play_ambience("base", time_play);  
    if (getRandomInt(1, 10) === 1) {  
        var amb_other_name = Object.keys(ambience),  
            amb_other = getRandomInt(0, amb_other_name.length - 1);  
        play_ambience(amb_other_name[amb_other], getRandomInt(5, 15));  
    }  
    window.setTimeout(function () {  
        ambience_change();  
    }, (time_wait) * 1000);  
}
```

Changes the base ambience effect, and causes additional ambience to occasionally play as well

Determines how many times the base sound will play, and then how long to wait before changing the base sound

Plays a random different ambience file sometimes

Calls the procedure after a time delay (recursion)

```
function ambience_fadein() {  
    (function myLoop(i) {  
        window.setTimeout(function () {  
            ambience_multiplier_fade = (60 - i) / 60;  
            ambience_multiplier = ambience_multiplier_fade * ambience_multiplier_music;  
            if (--i) myLoop(i);  
            else {  
                ambience_multiplier_fade = 1;  
                ambience_multiplier = ambience_multiplier_fade * ambience_multiplier_music;  
            }  
        }, 16.666666666666667, [i]);  
    })(60);  
}
```

A loop that waits until the scripts inside have finished before running the next loop

Increases the audio by 1/60th 60 times in a second, fading the volume from muted to full

Game immersion – ambience

```
function ambience_fadeout() {  
    (function myLoop(i) {  
        window.setTimeout(function () {  
            ambience_multiplier_fade = i / 60;  
            ambience_multiplier = ambience_multiplier_fade * ambience_multiplier_music;  
            if (--i) myLoop(i);  
            else {  
                ambience_multiplier_fade = 0;  
                ambience_multiplier = ambience_multiplier_fade * ambience_multiplier_music;  
            }  
        }, 16.666666666666667, [i]);  
    })(60);  
}
```

A loop that waits until the scripts inside have finished before running the next loop

```
ambience = { //stores the game ambience  
    base: /*the various base sounds*/ [(new Audio("assets/sounds/ambience_base1.ogg")), (new Audio("assets/sounds/ambience_base2.ogg")), (new  
    Audio("assets/sounds/ambience_base3.ogg")), (new Audio("assets/sounds/ambience_base4.ogg"))],  
    air: [(new Audio("assets/sounds/ambience_air1.ogg")), (new Audio("assets/sounds/ambience_air2.ogg"))]  
},  
...  
initialise();  
ambience_change();
```

An array containing all the ambience audio elements

```
initialise();  
ambience_change();
```

Called at the beginning when the game begins

```
clear();  
ambience_fadeout();  
document.getElementById("1").  
  
clear();  
ambience_fadeout();  
document.getElementById("1").
```

Called when the user either completes a level or dies

Game immersion – ambience

```
function play_ambience(type, duration) {
    var audioplay = ambience[type][getRandomInt(0, ambience[type].length - 1)].cloneNode(),
        i, amb_time_begin = Date.now();
    audioplay.volume = 0;
    audioplay.play();
    (function myLoop(i) {
        window.setTimeout(function () {
            audioplay.volume = (ambience_volume.value * ambience_multiplier * ((60 - i) / 60));
            if (--i) myLoop(i);
            else {
                audioplay.volume = ambience_volume.value * ambience_multiplier;
                var check_amb_fade_out = setInterval(function () {
                    audioplay.volume = (ambience_volume.value * ambience_multiplier);
                    if (audioplay.currentTime + 0.1 >= audioplay.duration) {
                        audioplay.currentTime = 0.1;
                    }
                })
                if (Date.now() - amb_time_begin > Number(duration) * 1000 - 1000) {
                    clearInterval(check_amb_fade_out);
                    (function myLoop(i) {
                        window.setTimeout(function () {
                            audioplay.volume = (ambience_volume.value * ambience_multiplier * i / 60);
                            if (--i) myLoop(i);
                            else {
                                audioplay.volume = 0;
                                audioplay.pause();
                            }
                        }, 16.666666666666667, [i]);
                    })(60);
                }
            }
        }, 16.666666666666667);
    })(60);
}, 16.666666666666667, [i]);
})(60);
}
```

Plays a given audio file a given number of times (passed parameters)

A loop to fade the audio in over 1 second

Loops the audio file when it is 0.1 seconds from finishing

When the audio has played enough times, it is muted

A loop to fade the audio out over 1 second

Game immersion – ambience testing



Game immersion – background music

- Yet another common feature to games is background music. It is, again, commonly used to set the atmosphere for a game, and is something that people often associate with a game (for example the Mario © games have a distinctive introduction).
- I wish to incorporate background music into my project.
- I will need to ensure that all music is obtained lawfully and is available for distribution.
- When music plays, I wish to decrease the volume of the ambience effect so that the background music is more prominent.

Game immersion – bg music

```
function play_music(type) {  
    var audioplay = music[type][getInt(0, music[type].length - 1)].cloneNode();  
    audioplay.volume = music_volume.value;  
    audioplay.play();  
    ambience_fadeout_music();  
    window.setTimeout(function () {  
        ambience_fadein_music();  
    }, (audioplay.duration - 1) * 1000);  
}  
  
function music_change() {  
    var music_other_name = Object.keys(music),  
        music_other = getInt(0, music_other_name.length - 1),  
        time_wait = getInt(400, 600);  
    play_music(music_other_name[music_other]);  
    window.setTimeout(function () {  
        music_change();  
    }, (time_wait) * 1000);  
}  
  
ambience_change();  
music_change();  
var checkinit = set
```

Plays the audio file

Fades the ambience effect to half of its original volume

Fades the ambience effect back in after the music finishes

Determines which audio will play, and how long it will wait before playing a new music

Plays the music type

Calls the procedure after a time delay (recursion)

Calls the procedure on the project loading

Game immersion – bg music

```
function ambience_fadein_music() {  
    (function myLoop(i) {  
        window.setTimeout(function () {  
            ambience_multiplier_music = (120 - i) / 120;  
            ambience_multiplier = ambience_multiplier_fade * ambience_multiplier_music;  
            if (--i) myLoop(i);  
            else {  
                ambience_multiplier_music = 1;  
                ambience_multiplier = ambience_multiplier_fade * ambience_multiplier_music;  
            }  
        }, 16.666666666666667, [i]);  
    })(60);  
}
```

A loop to fade the audio in over 1 second

```
window.setTimeout(function () {  
    ambience_multiplier_fade = (60 - i) / 60;  
    ambience_multiplier = ambience_multiplier_fade * ambience_multiplier_music;|
```

```
function ambience_fadeout_music() {  
    (function myLoop(i) {  
        window.setTimeout(function () {  
            ambience_multiplier_music = (i + 60) / 120;  
            ambience_multiplier = ambience_multiplier_fade * ambience_multiplier_music;  
            if (--i) myLoop(i);  
            else {  
                ambience_multiplier_music = 0.5;  
                ambience_multiplier = ambience_multiplier_fade * ambience_multiplier_music;  
            }  
        }, 16.666666666666667, [i]);  
    })(60);  
}
```

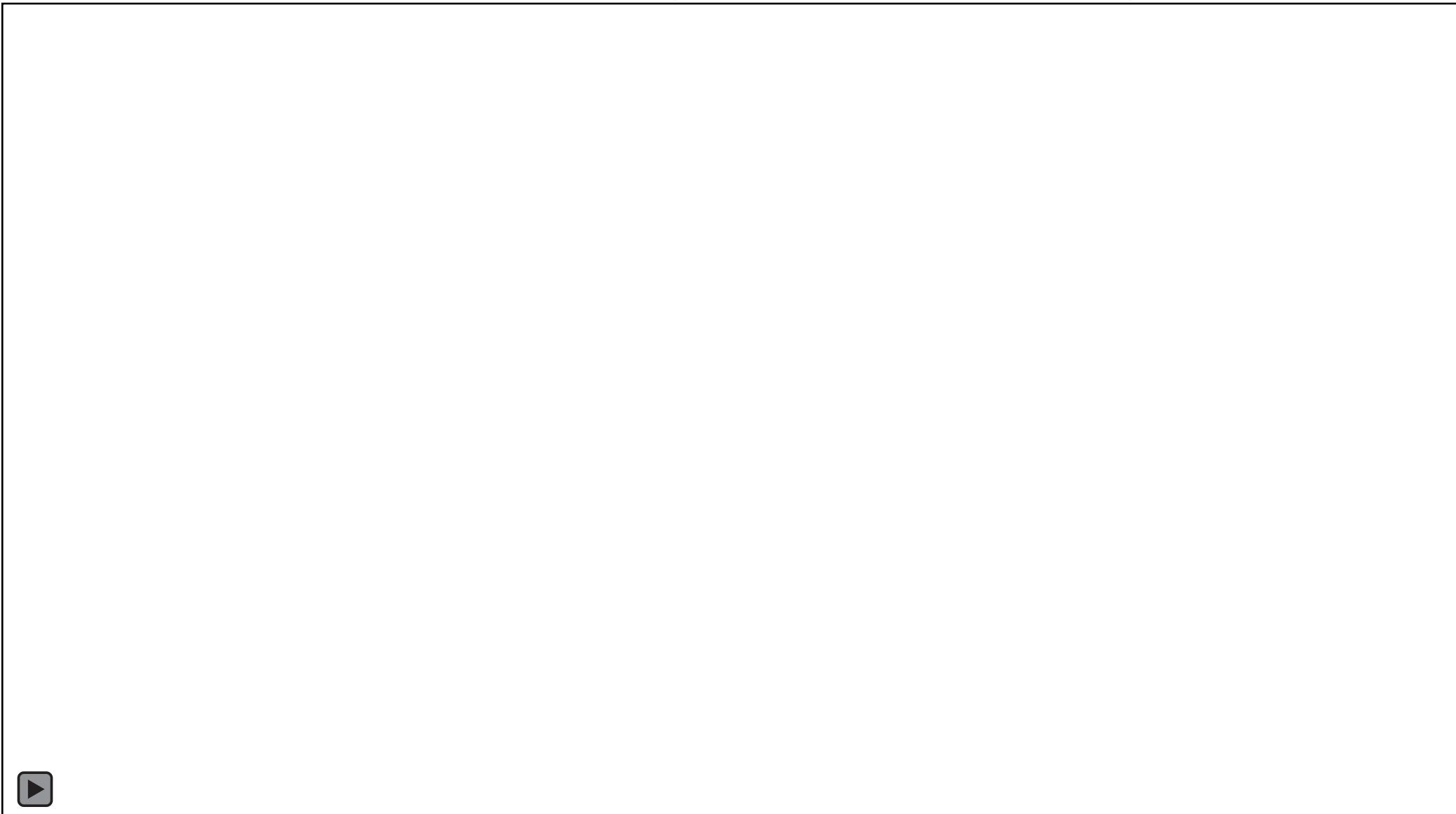
A loop to fade the audio out to half over 1 second

Game immersion – bg music

An array containing all the music audio elements

```
music = { //stores the game music
    buzz_light: [(new Audio("assets/sounds/music_buzz_light1.ogg"))],
    chase: [(new Audio("assets/sounds/music_chase1.ogg"))],
    dream: [(new Audio("assets/sounds/music_dream1.ogg")), (new Audio("assets/sounds/music_dream2.ogg"))],
    float: [(new Audio("assets/sounds/music_float1.ogg")), (new Audio("assets/sounds/music_float2.ogg")), (new
    Audio("assets/sounds/music_float3.ogg"))],
    saw: [(new Audio("assets/sounds/music_saw1.ogg"))],
    wet: [(new Audio("assets/sounds/music_wet1.ogg"))]
};
```

Game immersion – bg music testing

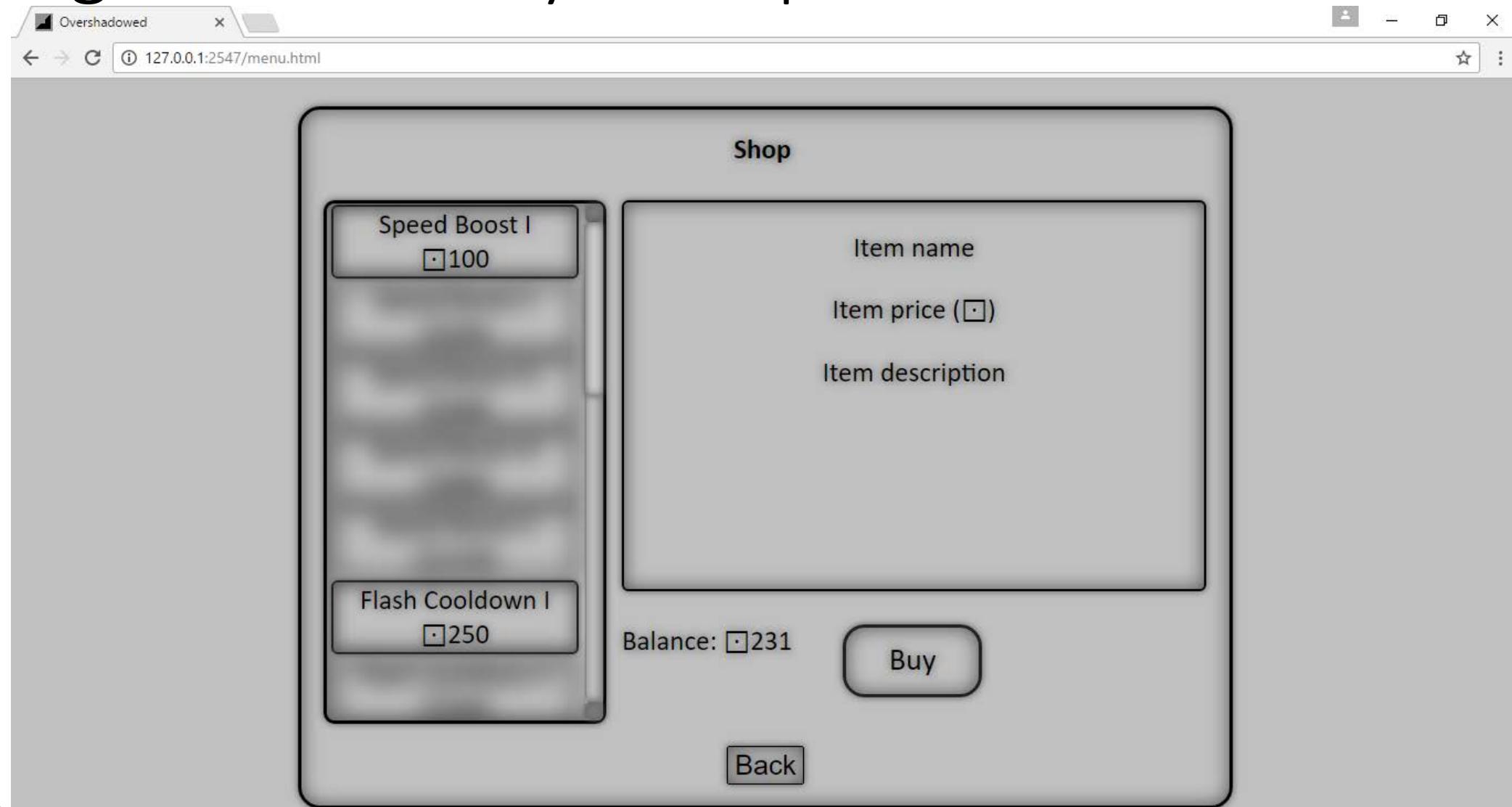


In-game currency – shop

- Currently the in-game currency has no way to be spent. Introducing an in-game shop to sell the qbits will allow the user to spend their qbits on upgrades.
- The shop will have 2 kinds of purchasable items: upgrades and aesthetic changes.
- Upgrades will change gameplay, making it possible to have better times in a level.
- Aesthetics will change the appearance of in-game elements, such as shadow colour.
- Some items will be locked, and previous upgrades must be bought to unlock them.

The original plan for arrows has been replaced by a normal scrollbar, as I believe this to be more user friendly.

In-game currency – shop



In-game currency – shop

```
function setupsaves() {
    var str = "(pid",
    str2 = "(pid",
    n;
    for (n = 0; n < levels.length; n += 1) {
        str += ",l" + levels[n][0].id + "items,l" + levels[n][0].id + "time";
    }
    str += ")";
    for (n = 0; n < buyables.length - 1; n += 1) {
        str2 += "," + buyables[n].name.replace(/\ /g, "_"); //use _ not space
    }
    str2 += ")";
    db.transaction(function (tx) {
        tx.executeSql('CREATE TABLE SAVEPROGRESS ' + str);
        tx.executeSql('CREATE TABLE UPGRADES ' + str2);
    });
    window.setTimeout(function () {
        createsavefile("3718274321299354", "ADMIN", true, false);
    }, 200);
    window.setTimeout(function () {
        document.getElementById("level_name_text").innerHTML = "generated";
    }, 400);
}
```

A new table has been introduced to save which upgrades have been bought

Iterates for each purchasable in the array

In-game currency – shop

```
function createsavefile(id, name, admin, update) {
    var str = "(pid",
    str2 = '(\" + id + "")',
    str3 = "(pid",
    str4 = '(\" + id + "")',
    obj, n;
    for (n = 0; n < levels.length; n += 1) {
        str += ",l" + levels[n][0].id + "items,l" + levels[n][0].id + "time";
        if (admin) {
            str2 += ',"0","0"';
        }
        else {
            str2 += '","",""';
        }
    }
    for (n = 0; n < buyables.length - 1; n += 1) {
        str3 += ',' + buyables[n].name.replace(/\ /g, "_");
        str4 += ',"0"';
    }
    str += ")";
    str2 += ")";
    str3 += ")";
    str4 += ")";
    db.transaction(function (tx) {
        tx.executeSql('INSERT INTO SAVEFILES (id,name,admin,qbits,reset) VALUES (' + id + "," + name + "," + admin + "," + "0" + ",false")');
        tx.executeSql('INSERT INTO SAVEPROGRESS ' + str + ' VALUES ' + str2);
        tx.executeSql('INSERT INTO SETTINGS (pid, key0, key1, key2, key3, key4, spacebar, music, sfx, ambience, aesthetic) VALUES (' + id +
        "," + "87" + "," + "83" + "," + "65" + "," + "68" + "," + "32" + "," + "false" + "," + "0.7" + "," + "0.9" + "," + "0.6" + "," + "null" + ')');
        tx.executeSql('INSERT INTO UPGRADES ' + str3 + ' VALUES ' + str4);
    });
    if (update === true) {
        obj = {};
        obj.id = id;
        obj.name = name;
        obj.admin = admin;
        obj.qbits = 0;
        obj.reset = false;
        logindetails.push(obj);
    }
}
```

Creates the query based off the upgrades array

Iterates for each purchasable in the array

Executes the SQL insert query

In-game currency – shop

```
function getusersupgrades(id, admin) {  
    var output = {};  
    output.list = [];  
    output.qbits = 0;  
    db.transaction(function (tx) {  
        tx.executeSql('SELECT * FROM UPGRADES WHERE pid=' + id + "", [], function (tx, results) {  
            var o;  
            for (o in results.rows[0]) {  
                if (results.rows[0][o] == "1") {  
                    output.list.push(o.replace(/\_/g, " "));  
                    if (!admin) {  
                        output.qbits += buyables[o - 1].price;  
                    }  
                }  
            }  
        }, null);  
    });  
    return output;  
}
```

Returns an array of the bought upgrades for a user

Iterates through the SQL query output

If the user isn't an admin, decrements their qbits

```
function updateuserupgrades(id, upgrade) {  
    db.transaction(function (tx) {  
        tx.executeSql('UPDATE UPGRADES SET ' + upgrade.replace(/\_/g, "_") + '=1' WHERE pid=' + id + "");  
    });  
}
```

Updates the SQL table to have bought an upgrade

In-game currency – shop

```
var kill = document.getElementsByClassName("shop_list"),
    shop_buy, name_span, price_span, line_break, temp_list, i, n;
while (kill[0]) { _____
    kill[0].parentNode.removeChild(kill[0]);
}
```

Iteratively deletes each and every item displayed in the shop item list

```
for (i = 0; i < buyables.length - 1; i += 1) {
    shop_buy = document.createElement("div");
    shop_buy.setAttribute("id", "shop_buy" + i);
    shop_buy.setAttribute("class", "shop_list large_glow");
    name_span = document.createElement("span");
    name_span.setAttribute("id", "shop_buy_name" + i);
    name_span.setAttribute("class", "non_click");
    name_span.innerHTML = buyables[i].name;
    line_break = document.createElement("br");
    line_break.setAttribute("class", "non_click");
    price_span = document.createElement("span");
    price_span.setAttribute("id", "shop_buy_name" + i);
    price_span.setAttribute("class", "non_click");
    price_span.innerHTML = "¤" + buyables[i].price;
    if (!buyables[i].requirements.every(function (val) {
        return bought.indexOf(val) >= 0;
    })) {
        shop_buy.classList.add("locked");
    }
    if (bought.indexOf(buyables[i].name) != -1) {
        shop_buy.classList.add("shop_item_bought");
    }
    shop_buy.appendChild(name_span);
    shop_buy.appendChild(line_break);
    shop_buy.appendChild(price_span);
    document.getElementById("shop_inner_menu").appendChild(shop_buy);
}
```

Iterates through an array, creating an element for each buyable item in the item list

Checks if the item is locked or not, and makes it blurred if it is locked

In-game currency – shop

```
document.getElementById("shop_balance").innerHTML = "Balance: " +loggedin_user.qbits;
temp_list = document.getElementsByClassName("shop_list");
for (i = 0; i < temp_list.length; i += 1) {
    temp_list[i].addEventListener("click", function (event) {
        var me = event.target,
            temp_list = document.getElementsByClassName("shop_selected"),
            me2 = Number(me.id.replace(/shop_buy/g, '')),
            i;
        for (i = temp_list.length - 1; i >= 0; i -= 1) {
            temp_list[i].classList.remove("shop_selected");
        }
        me.classList.add("shop_selected");
        if (me.classList.contains("locked")) {
            document.getElementById("shop_item_description").classList.add("locked", "encoded");
        }
        else {
            document.getElementById("shop_item_description").classList.remove("locked", "encoded");
        }
        document.getElementById("shop_item_description_name").innerHTML = buyables[me2].name;
        document.getElementById("shop_item_description_price").innerHTML = "₹" + buyables[me2].price;
        document.getElementById("shop_item_description_description").innerHTML = buyables[me2].description;
        if ((loggedin_user.qbits >= buyables[me2].price || String(loggedin_user.admin) == "true") && !me.classList.contains("locked")) {
            document.getElementById("shop_buy_button").classList.remove("locked");
        }
        else {
            document.getElementById("shop_buy_button").classList.add("locked");
        }
        if (buyables[me2].toggleable == true && bought.indexOf(buyables[me2].name) != -1) {
            document.getElementById("shop_toggle_container").classList.remove("hide");
            document.getElementById("shop_checkbox_container").classList.remove("hide");
        }
        else if (buyables[me2].toggleable == true) {
            document.getElementById("shop_toggle_container").classList.add("hide");
            document.getElementById("shop_checkbox_container").classList.remove("hide");
        }
        else {
            document.getElementById("shop_toggle_container").classList.add("hide");
            document.getElementById("shop_checkbox_container").classList.add("hide");
        }
        if (buyables[me2].name == aesthetic_toggle) {
            document.getElementById("shop_checkbox").checked = true;
        }
    }Continued below
```

Updates the display for the user's balance

Iteratively adds an event to each button on click

When clicked, deselects previously selected element(s)

If the item is locked, makes text unreadable

Sets the item title, description and price

In-game currency – shop

```
        else {
            document.getElementById("shop_checkbox").checked = false;
        });
    }
}

for (i = 0; i < bought.length; i += 1) {
    for (n = 0; n < buyables.length; n += 1) {
        if (buyables[n].name == bought[i]) {
            shop_transaction_modify(buyables[n].change);
            break;
        }
    }
}

if (aesthetic_toggle != "null") {
    reset_shadow_style();
    for (i = 0; i < buyables.length; i += 1) {
        if (buyables[i].name == aesthetic_toggle) {
            shop_transaction_modify(buyables[i].change);
            break;
        }
    }
} else {
    reset_shadow_style();
}
```

Iterates for each bought item in the array

Iterates for each purchasable in the array

Applies changes from upgrades to each bought element

Iterates for each purchasable in the array

Applies aesthetic changes from the selected enabled element

In-game currency – shop

The above script is all wrapped in a procedure

```
function shop_reset() {  
    reset_shadow_style();  
    var kill = document.getElementsByClassName("shop_list")
```

Toggles an aesthetic change on checkbox click

```
document.getElementById("shop_checkbox").addEventListener("click", function (event) {  
    var me = event.target,  
        selected = document.getElementsByClassName("shop_selected")[0];  
    if (me.checked) {  
        aesthetic_toggle = selected.childNodes[0].innerHTML;  
        updateusersettings(loggedin_user.id, "aesthetic", aesthetic_toggle);  
        var selected_id = Number(selected.id.replace(/shop_buy/g, ''));  
        shop_transaction_modify(buyables[selected_id].change);  
    }  
    else {  
        aesthetic_toggle = null;  
        updateusersettings(loggedin_user.id, "aesthetic", aesthetic_toggle);  
        reset_shadow_style();  
    }  
});
```

In-game currency – shop

```
document.getElementById("shop_buy_button").addEventListener("click", function (event) {  
    var me = event.target,  
        me2 = document.getElementsByClassName("shop_selected")[0];  
    if (!me2.classList.contains("locked") && !me2.classList.contains("shop_item_bought") && (String(loggedin_user.admin) == "true" ||  
loggedin_user.qbits >= Number(document.getElementById("shop_item_description_price").innerHTML.replace(/\u00a0/g, ''))) {  
        updateuserupgrades(loggedin_user.id, document.getElementById("shop_item_description_name").innerHTML);  
        if (String(loggedin_user.admin) != "true") {  
            loggedin_user.qbits -= Number(document.getElementById("shop_item_description_price").innerHTML.replace(/\u00a0/g, ''));  
        }  
        document.getElementById("shop_balance").innerHTML = "Balance: " + loggedin_user.qbits;  
        bought.push(document.getElementById("shop_item_description_name").innerHTML);  
        if (!document.getElementById("shop_checkbox_container").classList.contains("hide")) {  
            document.getElementById("shop_toggle_container").classList.remove("hide");  
            document.getElementById("shop_checkbox").checked = true;  
            aesthetic_toggle = me2.childNodes[0].innerHTML;  
            updateusersettings(loggedin_user.id, "aesthetic", aesthetic_toggle);  
        }  
        var me3 = Number(me2.id.replace(/shop_buy/g, ''));  
        shop_transaction_modify(buyables[me3].change);  
        shop_check_locked();  
        alertmsg("Transaction successfully completed.");  
        me2.classList.add("shop_item_bought");  
    }  
    else if (loggedin_user.qbits < Number(document.getElementById("shop_item_description_price").innerHTML.replace(/\u00a0/g, '')) &&  
String(loggedin_user.admin) != "true") {  
        alertmsg("Insufficient funds to make purchase.");  
    }  
    else if (me2.classList.contains("shop_item_bought")) {  
        alertmsg("You have already purchased this item.");  
    }  
    else {  
        alertmsg("This item is locked. Please purchase other items to unlock it.");  
    }  
});
```

Checks item is available to purchase

Makes the transaction and change(s)

Iterates for each item in the parameter array

In-game currency – shop

```
function shop_check_locked() {  
    var i, shop_buy  
    for (i = 0; i < buyables.length - 1; i += 1) {  
        shop_buy = document.getElementById("shop_buy" + i);  
        if (!buyables[i].requirements.every(function (val) {  
            return bought.indexOf(val) >= 0;  
        })) {  
            shop_buy.classList.add("locked");  
        }  
        else {  
            shop_buy.classList.remove("locked");  
        }  
    }  
}
```

Iterates for each purchasable in the array

Checks if each item is locked or not, and makes it blurred if so

Checks the type of each modifier, and updates the variable or changes the style of the element(s)

```
function shop_transaction_modify(data) {  
    var i, mod;  
    for (i = 0; i < data.length; i += 1) {  
        if (data[i].to_change == "var") {  
            mod = data[i].value;  
            if (data[i].change_type == "add") {  
                window[data[i].change] += Number(mod);  
            }  
            else if (data[i].change_type == "mult") {  
                window[data[i].change] *= Number(mod);  
            }  
            else {  
                window[data[i].change] = mod;  
            }  
        }  
        else if (data[i].to_change == "id") {  
            mod = document.getElementById(data[i].change);  
            mod.style[data[i].changing] = data[i].value;  
        }  
        else if (data[i].to_change == "css") {  
            mod = data[i].change;  
            css_getclass(mod).style[data[i].changing] = data[i].value;  
        }  
        else {  
            mod = document.getElementById(data[i].change).classList;  
            if (data[i].value == "add") {  
                mod.add(data[i].changing);  
            }  
            else {  
                mod.remove(data[i].changing);  
            }  
        }  
    }  
}
```

In-game currency – shop

```
function reset_shadow_style() {
    shop_transaction_modify([
        {
            to_change: "css",
            change: ".shadowray",
            changing: "fill",
            value: "#000"
        },
        {
            to_change: "css",
            change: ".shadowray",
            changing: "stroke",
            value: "#000"
        },
        {
            to_change: "id",
            change: "canvas",
            changing: "background-color",
            value: "#000"
        }
    ]);
}
```

Makes the shadows appear as their default style

In-game currency – shop

```
var buyables = [{  
    name: "Speed Boost I",  
    price: "100",  
    description: "Increases movement speed of the player, allowing for faster completion of a level.",  
    requirements: [],  
    toggleable: false,  
    change: [{  
        to_change: "var",  
        change: "speed_multiplier",  
        change_type: "mult",  
        value: 1.2  
    }]  
}, {  
    name: "Speed Boost II",  
    price: "250",  
    description: "Increases movement speed of the player, allowing for faster completion of a level.",  
    requirements: ["Speed Boost I"],  
    toggleable: false,  
    change: [{  
        to_change: "var",  
        change: "speed_multiplier",  
        change_type: "mult",  
        value: 1.2  
    }]  
}, {  
    name: "Speed Boost III",  
    price: "500",  
    description: "Increases movement speed of the player, allowing for faster completion of a level.",  
    requirements: ["Speed Boost II"],  
    toggleable: false,  
    change: [{  
        to_change: "var".  
    }]  
}]
```

All upgrades are stored in this array

An example of an upgrade

In-game currency – shop

```
    }, {
      name: "Pink Shadows",
      price: "1000",
      description: "An aesthetic change to make the shadows appear pink in game.",
      requirements: [],
      toggleable: true,
      change: [
        {
          to_change: "css",
          change: ".shadowray",
          changing: "fill",
          value: "pink"
        },
        {
          to_change: "css",
          change: ".shadowray",
          changing: "stroke",
          value: "pink"
        },
        {
          to_change: "css",
          change: ".obstacle_box",
          changing: "fill",
          value: "pink"
        },
        {
          to_change: "css",
          change: ".obstacle_box",
          changing: "stroke",
          value: "pink"
        },
        {
          to_change: "id",
          change: "canvas",
          changing: "background-color",
          value: "pink"
        }
      ]
    },
    ...
  
```

An example of an aesthetic change upgrade

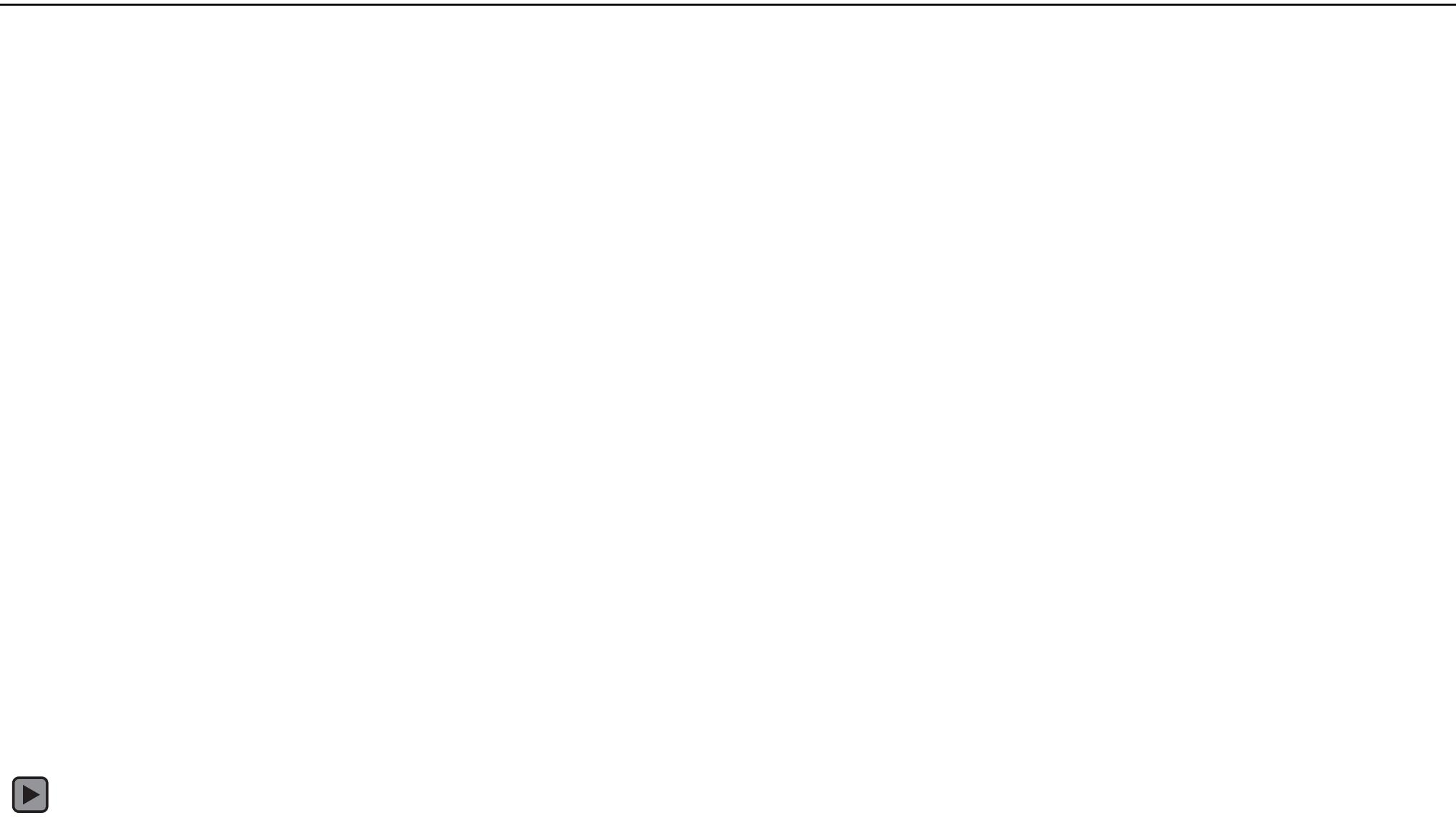
In-game currency – shop testing



In-game currency – shop testing



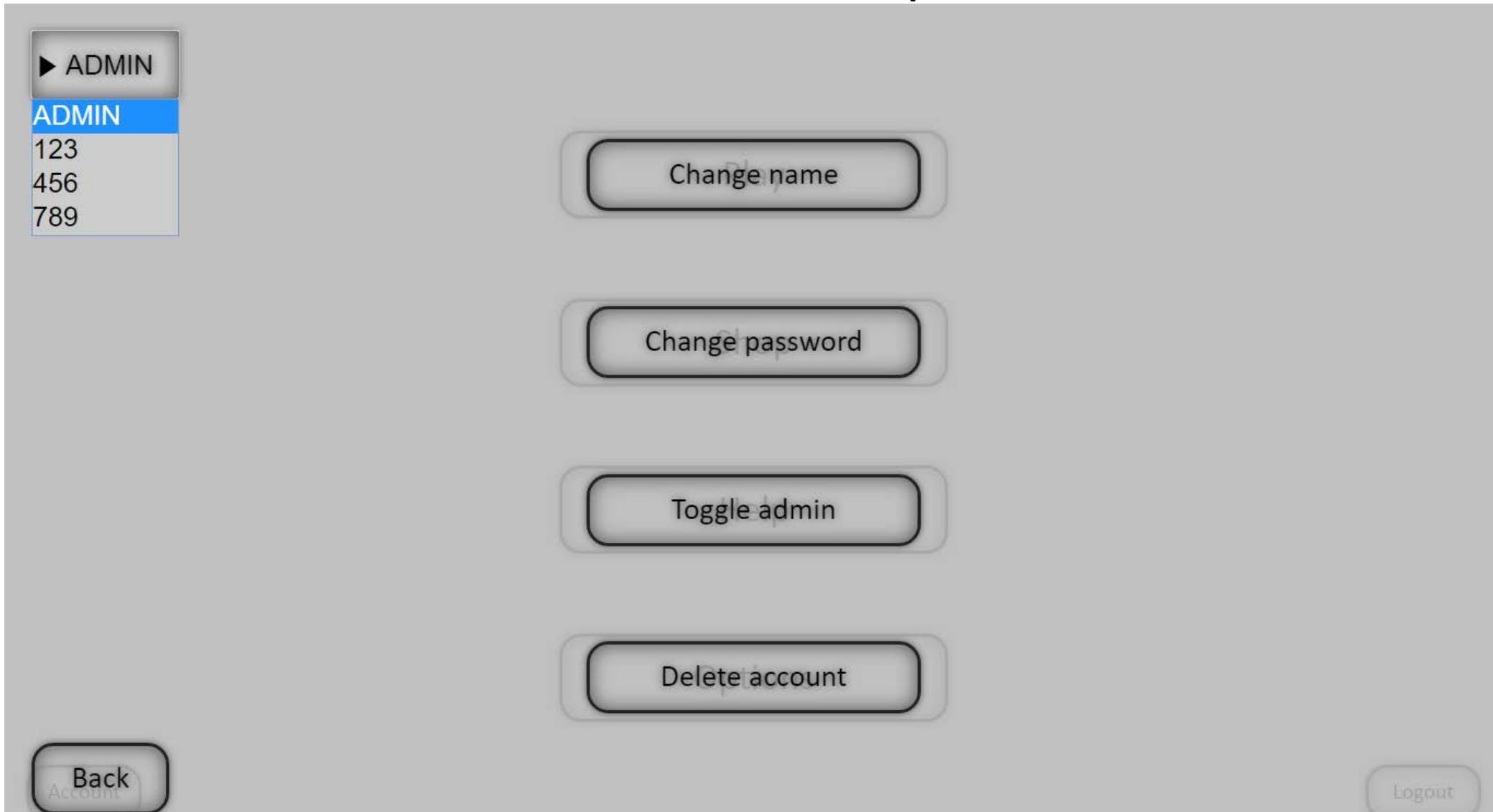
In-game currency – shop testing



User account controls – dropdown

- Currently, administrators cannot remotely edit a different user's account, making them somewhat redundant.
- To rectify this, I am adding a dropdown of user's which will only be visible to admins, and will default to the selected user.
- This will allow admins to remotely access and edit other user's accounts.

User account controls – dropdown



User account controls – dropdown

```
document.getElementById("account_button").addEventListener("click", function () {  
    var temp_mod = document.getElementById("account_settings");  
    temp_mod.classList.remove("fadeoutblur");  
    void temp_mod.offsetWidth;  
    temp_mod.classList.add("fadeinblur");  
    temp_mod.classList.remove("hide");  
    temp_mod = document.getElementById("accounts_user_dropdown");  
    if (String(loggedin_user.admin) == "true") {  
        temp_mod.classList.remove("hide");  
    } else {  
        temp_mod.classList.add("hide");  
    }  
});
```

Only shows the dropdown if the user is an admin

User account controls – dropdown

```
function dropdown_draw(inheritance) {  
    var n, opt,  
        kill = document.getElementsByClassName("options_dropdown");  
    while (kill[0]) {  
        kill[0].parentNode.removeChild(kill[0]);  
    }  
    for (n = 0; n < logindetails.length; n += 1) {  
        opt = document.createElement("option");  
        opt.id = "user_select_option_" + n;  
        opt.classList.add("options_dropdown");  
        opt.value = logindetails[n].id;  
        opt.innerHTML = logindetails[n].name + " &ampnbsp&ampnbsp&ampnbsp&ampnbsp";  
        selector.appendChild(opt);  
    }  
    if (inheritance === null) {  
        selector.value = String(loggedin_user.id);  
    }  
    else {  
        selector.value = inheritance;  
    }  
}
```

Only shows the dropdown if the user is an admin

Loops if an element exists

Removes all of the options in the dropdown

Iterates for each user in the array

Adds options to the dropdown from the user list

Selects the appropriate user (themselves or another)

```
doLevelLocking(); //WORKINGON  
dropdown_draw(null);  
window.setTimeout(function () {
```

Example usage of the procedure (during player login)

User account controls – dropdown

```
document.getElementById("change_button_name").addEventListener("click", function () {
    var i, user_modding;
    for (i = 0; i < logindetails.length; i += 1) {
        if (logindetails[i].id == selector.value) {
            user_modding = logindetails[i];
        }
    }
    var user = user_modding.id,
        name = document.getElementById("new_name").value;
    if (String(user) != "3718274321299354" && String(name).length > 2) {
        changesavefilename(user, name, true);
        user_modding.name = name;
        alertmsg("New name set");
        document.getElementById("name_change").reset();
        dropdown_draw(user);
        record_times = getrecordtimes();
        window.setTimeout(function () {
            arrowr();
            window.setTimeout(function () {
                arrowl();
            }, 750);
        }, 250);
    }
    else if (String(user) != "3718274321299354") {
        alertmsg("Cannot edit the main ADMIN account in this way");
    }
    else {
        alertmsg("Player name must be at least 3 characters long");
    }
});
```

Now iterates through the user list to get the selected user from the dropdown, and makes them the user to change

This is common to all change functions, and is the same so not all have been shown.

Redraws the dropdown (as a name has changed)

User account controls – dropdown

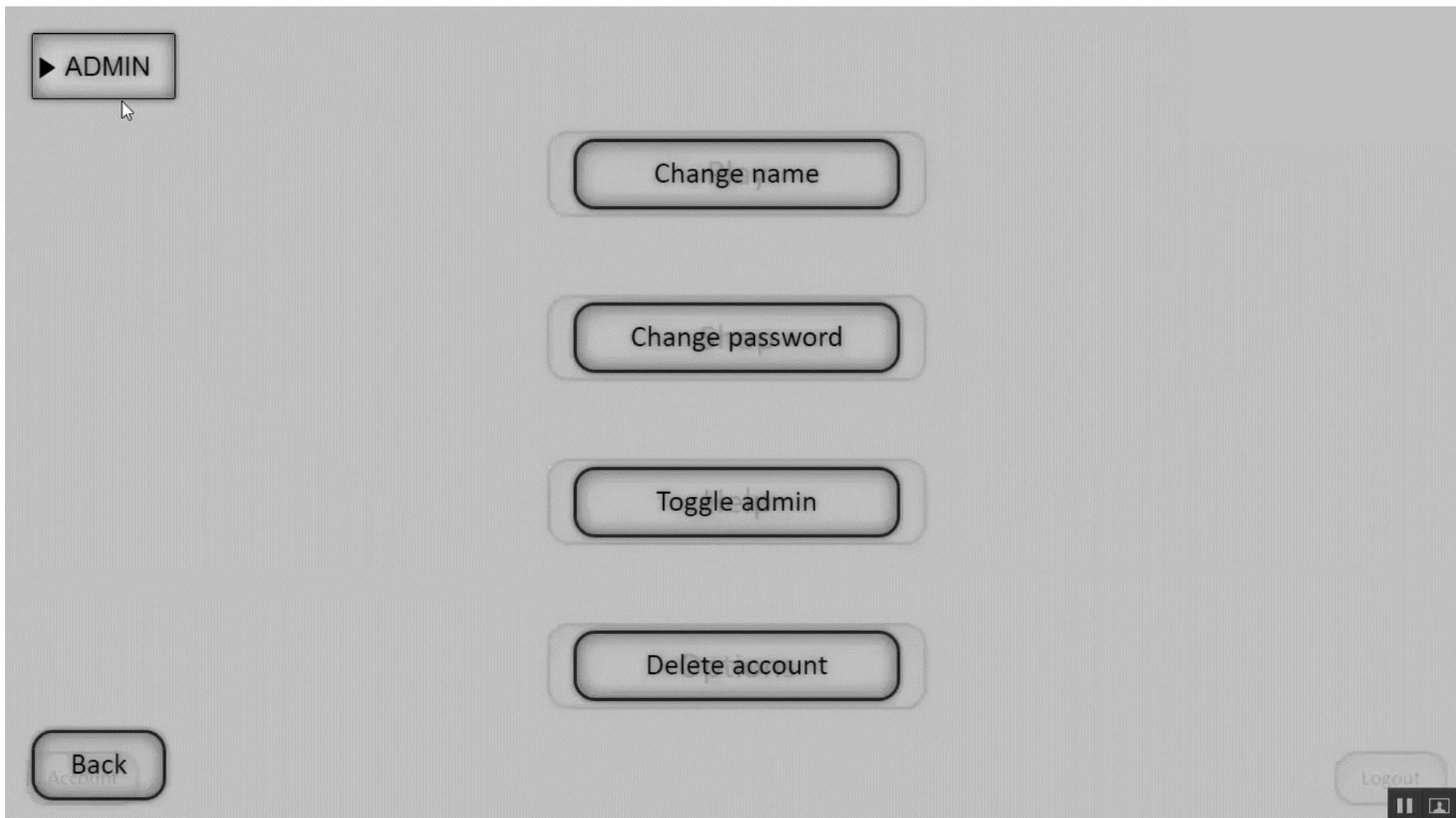
```
var admin_name = logindetails[i].name,  
    user_name = user_modding.name,  
    temp_mod;  
if (String(user_modding.admin) == "false") {  
    changesavefileadmin(user_modding.id, true, true);  
    alertmsg(admin_name + " has authorised " + user_name + " to be an admin");  
    changesavefileadmin(hashed, true, true);  
    user_modding.admin = "true";  
    temp_mod = document.getElementById("accounts_user_dropdown");  
    temp_mod.classList.remove("fadeoutblur");  
    void temp_mod.offsetWidth;  
    temp_mod.classList.add("fadeinblur");  
    temp_mod.classList.remove("hide");  
} else {  
    changesavefileadmin(user_modding.id, false, true);  
    alertmsg(admin_name + " has removed " + user_name + "'s admin powers");  
    user_modding.admin = "false";  
    temp_mod = document.getElementById("accounts_user_dropdown");  
    temp_mod.classList.remove("fadeinblur");  
    void temp_mod.offsetWidth;  
    temp_mod.classList.add("fadeoutblur");  
    window.setTimeout(function () {  
        temp_mod.classList.add("hide");  
    }, 500);  
}  
document.getElementById("admin_confirm").reset();
```

Inside the change admin status function, on password being accepted and status being changed

Shows the dropdown menu (as user is now an admin)

Hides the dropdown menu (as user is not an admin)

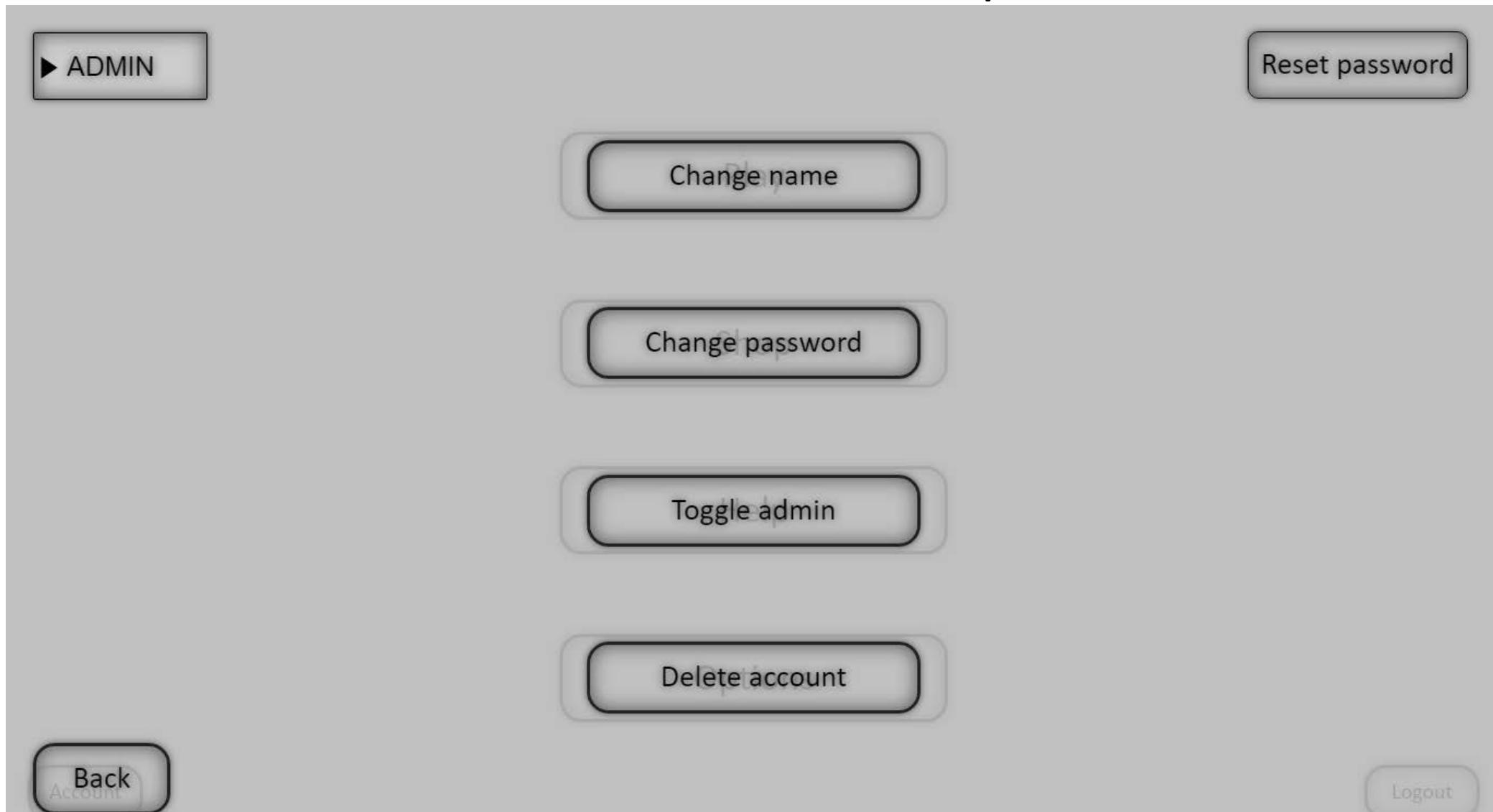
User account controls – dropdown testing



User account controls – reset password

- Currently, administrators cannot remotely change another user's password. This means that if a user has forgotten their password, then they cannot recover their account.
- To rectify this, I wish to add a password reset option which is only available to administrators.
- This will reset their password to 12345, and will then force the user to set a new password upon log in.
- The reset option will be saved in the SQL users table as a new column.

User account controls – reset password



User account controls – reset password

```
i = 0; i < logindetails.length; i += 1) {  
    if (logindetails[i].id == hashed || (logindetails[i].reset == true && pass.value == 12345)) {  
       loggedin user = logindetails[i];
```

Login now also checks if the password needs resetting

```
, 2000,  
if (logindetails[i].reset == true) {  
    logindetails[i].reset = false;  
    setsavefilereset(hashed, false, true);  
    window.setTimeout(function () {  
        var temp_mod = document.getElementById("login_window_reset_password_loggedin");  
        temp_mod.classList.remove("fadeoutblur");  
        void temp_mod.offsetWidth;  
        temp_mod.classList.add("fadeinblur");  
        temp_mod.classList.remove("hide");  
    }, 800);  
}
```

When logged in, shows the new password window and sets the reset to false (making it single use)

User account controls – reset password

```
function setsavefilereset(id, bool, update) {
    db.transaction(function (tx) {
        tx.executeSql('UPDATE SAVEFILES SET reset=' + bool + ' WHERE id=' + id + '');
    });
    if (update === true) {
        var obj = {},
            i;
        obj.id = id;
        obj.reset = bool;
        for (i = 0; i < logindetails.length; i += 1) {
            if (logindetails[i].id == id) {
                obj.name = logindetails[i].name;
                obj.admin = logindetails[i].admin;
                obj.qbits = logindetails[i].qbits;
                logindetails.splice(i, 1, obj);
                logindetails[i].reset = bool;
            }
        }
    }
}
```

Updates the SQL table to save whether the password is reset

Iterates for each user in the array

User account controls – reset password

```
document.getElementById("login_button_password_reset").addEventListener("click", function () {  
    var i, user_modding;  
    for (i = 0; i < logindetails.length; i += 1) {  
        if (logindetails[i].id == selector.value) {  
            user_modding = logindetails[i];  
        }  
    }  
    var user = loggedin_user.username,  
        pass = document.getElementById("login_reset_password"),  
        hashed = hash(user, pass.value, "9007199254740991");  
    if (user_modding.id != "3718274321299354") {  
        if (loggedin_user.id == hashed) {  
            setsavefilereset(user_modding.id, true, true);  
            alertmsg("Password reset to 12345");  
            var temp_mod = document.getElementById("login_window_reset_password");  
            temp_mod.classList.remove("fadeinblur");  
            void temp_mod.offsetWidth;  
            temp_mod.classList.add("fadeoutblur");  
            window.setTimeout(function () {  
                temp_mod.classList.add("hide");  
            }, 500);  
        }  
        else {  
            alertmsg("Password incorrect");  
        }  
    }  
    else {  
        alertmsg("Cannot reset main ADMIN password");  
    }  
});
```

Iterates for each user in the array

Gets the user selected in the dropdown

Checks the admin has entered their password correctly

Sets the reset column/record for the selected user

User account controls – reset password

```
temp_mod = document.getElementById("accounts_user_dropdown");
temp_mod2 = document.getElementById("reset_password_button");
temp_mod.classList.remove("fadeoutblur");
void temp_mod.offsetWidth;
temp_mod.classList.add("fadeinblur");
temp_mod.classList.remove("hide");
temp_mod2.classList.remove("fadeoutblur");
void temp_mod2.offsetWidth;
temp_mod2.classList.add("fadeinblur");
temp_mod2.classList.remove("hide");
```

The reset password button is now shown/hidden at the same time as the user dropdown list

User account controls – reset password testing



Initialisation progress

- Currently, the loading screen is rather unintuitive, and does not display any information on why it is there.
- The first time the project loads, the loading screen does not vanish until the program has initialised, which can take several minutes on slower systems.
- I wish to display the initialisation progress during this time, so the user can know what is happening.

Initialisation progress

```
var checkinit = setInterval(function () {  
    document.getElementById("init_progress").innerHTML = "Progress: " + initialisation_count + "/6";  
    if (initialisation_count == 6) {  
        document.getElementById("init_progress").innerHTML = "Progress: 6/6";  
        clearInterval(checkinit);  
        checkfullscreen();  
        clearloading();  
        window.setTimeout(function () {  
            document.getElementById("init_msg").classList.add("hide");  
        }, 1000);  
    }  
}, 200);
```

Runs check every 0.2 seconds

Updates the progress every 0.2 seconds, and also checks if the initialisation is complete

```
createsaverite("5718274521299554", "ADMIN", true, false);  
initialisation_count += 1;
```

The variable is updated when certain SQL queries are completed

Initialisation progress – testing



Loading animation

- The loading animation is currently very simple and boring. To rectify this, I will change it from a bar to an SVG icon of a gear which rotates slowly to indicate there is work going on behind the scenes.
- This purely aesthetic change will mean that the user will be slightly more engaged during loading times (such as the initialisation).

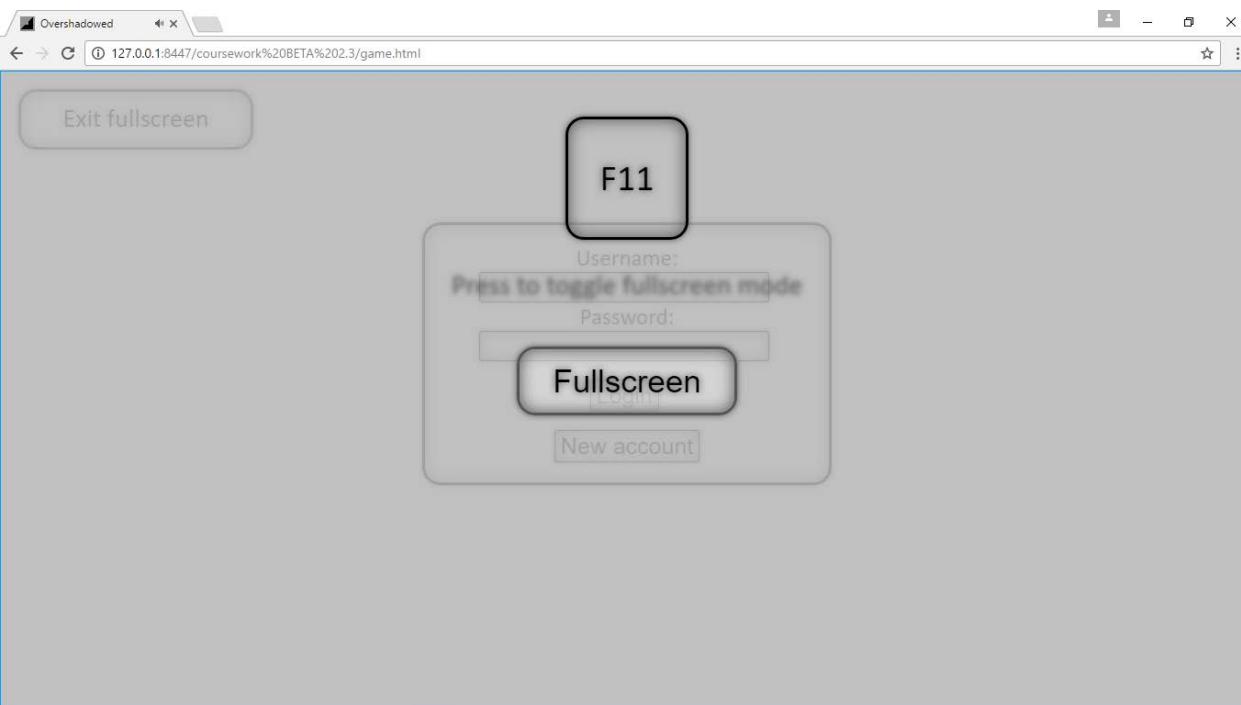
Loading animation – testing



Fullscreen buttons

- Currently the indication that the user is not in fullscreen is just a box with F11 displayed inside it. This will not be very clear to any user that they are meant to press this key. Additionally, Safari has a different key combination to go into fullscreen.
- As a result, I will add clickable buttons to toggle between fullscreen and not fullscreen.

Fullscreen buttons



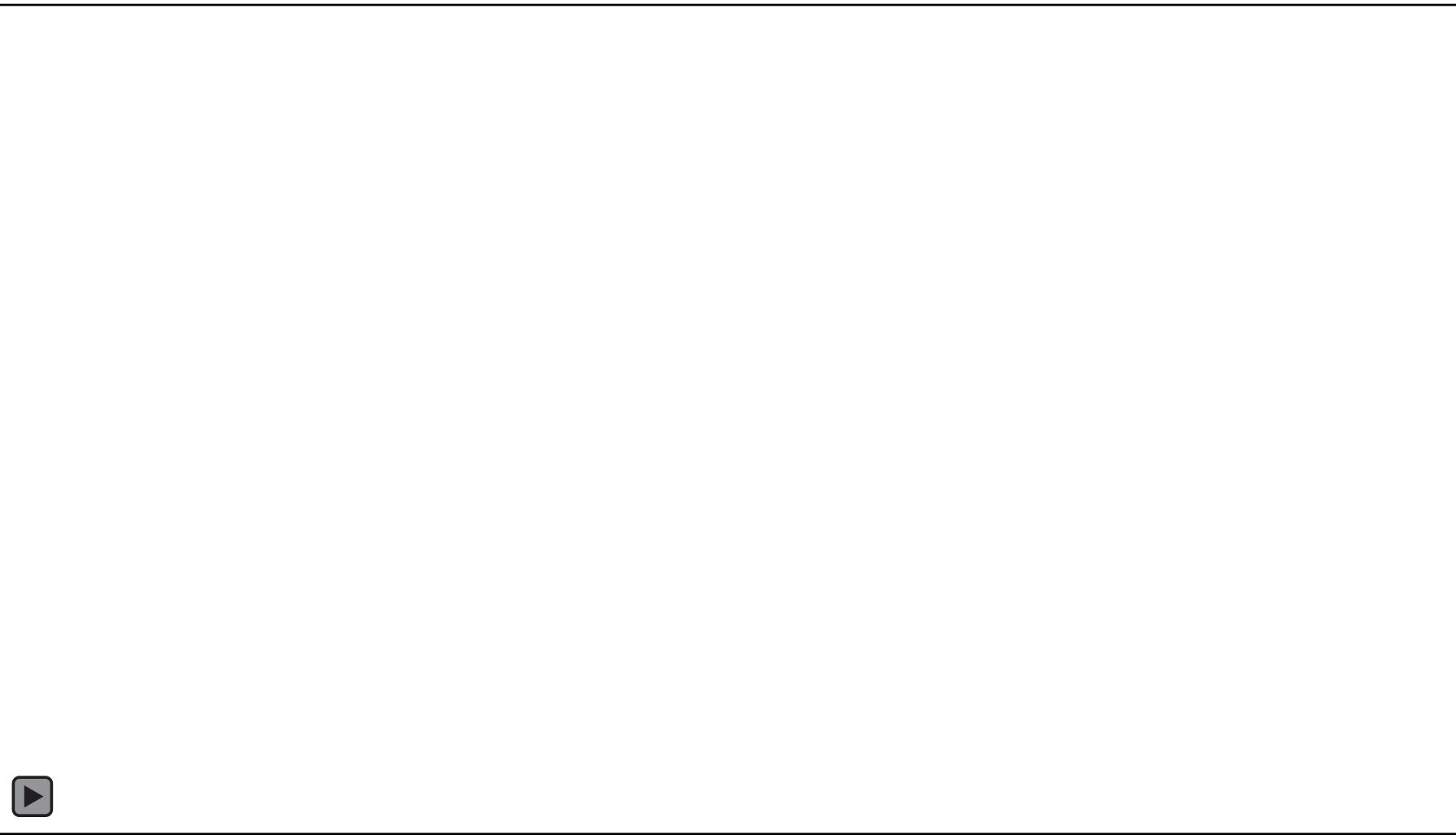
Fullscreen buttons

```
function request_enter_fullscreen() { ━━━━━━━━━━━━━━━━━━━━━━━━━━━━ Enters fullscreen mode
  var el = document.body,
      requestMethod = el.requestFullScreen || el.webkitRequestFullScreen || el.mozRequestFullScreen || el.msRequestFullScreen;
  if (requestMethod) {
    requestMethod.call(el);
  }
  else if (typeof window.ActiveXObject !== "undefined") {
    var wscript = new ActiveXObject("WScript.Shell");
    if (wscript !== null) {
      wscript.SendKeys("{F11}");
    }
  }
}

function request_close_fullscreen() { ━━━━━━━━━━━━━━━━━━━━━━━━━━━━ Exits fullscreen mode
  if (document.exitFullscreen) {
    document.exitFullscreen();
  }
  else if (document.webkitExitFullscreen) {
    document.webkitExitFullscreen();
  }
}
document.getElementById("FS").addEventListener("click", request_enter_fullscreen);
document.getElementById("NFS").addEventListener("click", request_close_fullscreen);
```

Button events that trigger the above procedures

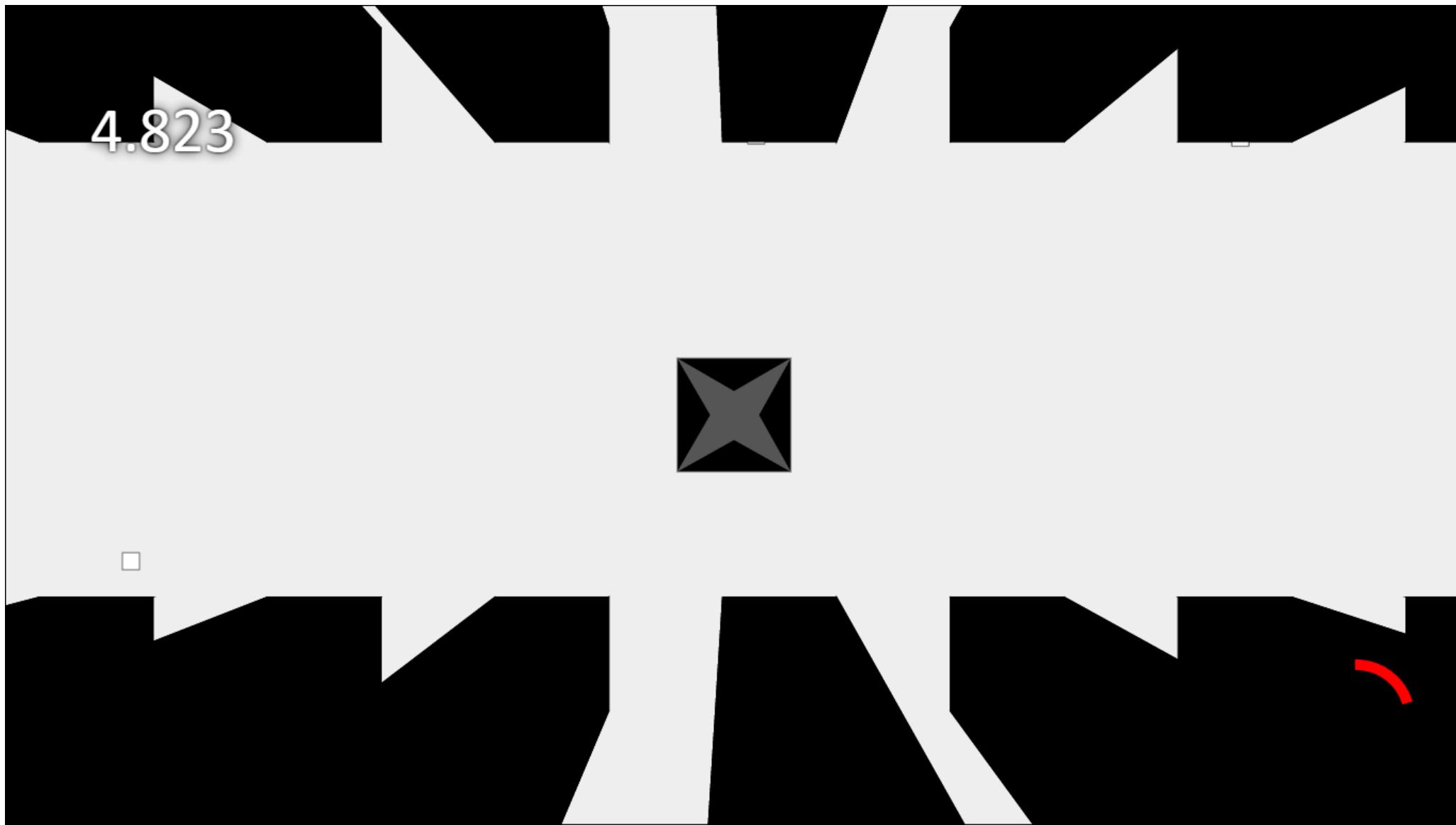
Fullscreen buttons – testing



Player appearance

- Currently the player appears as just a circle. When the hitbox is hidden, this will not make much sense as the hitbox is square and the player is not, so the player will seem to bounce off nothing on object corners.
- To rectify this, I will change the player sprite to be a star overlayed on a square. This means that the hitbox can still be visible, and will just have a different texture above it.
- The circle element will still exist, but simply be hidden beneath the hitbox and star.
- The star will be a polygon that has its position set to that of the hitbox.

Player appearance



Button icons

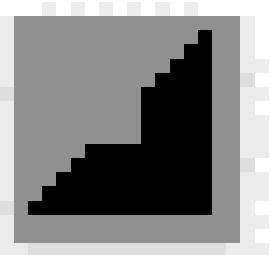
- As stated before, I wish to have buttons showing icons instead of just text. So far this is not the case.
- This will be a very simple visual change, but I believe it will be effective.

Button icons

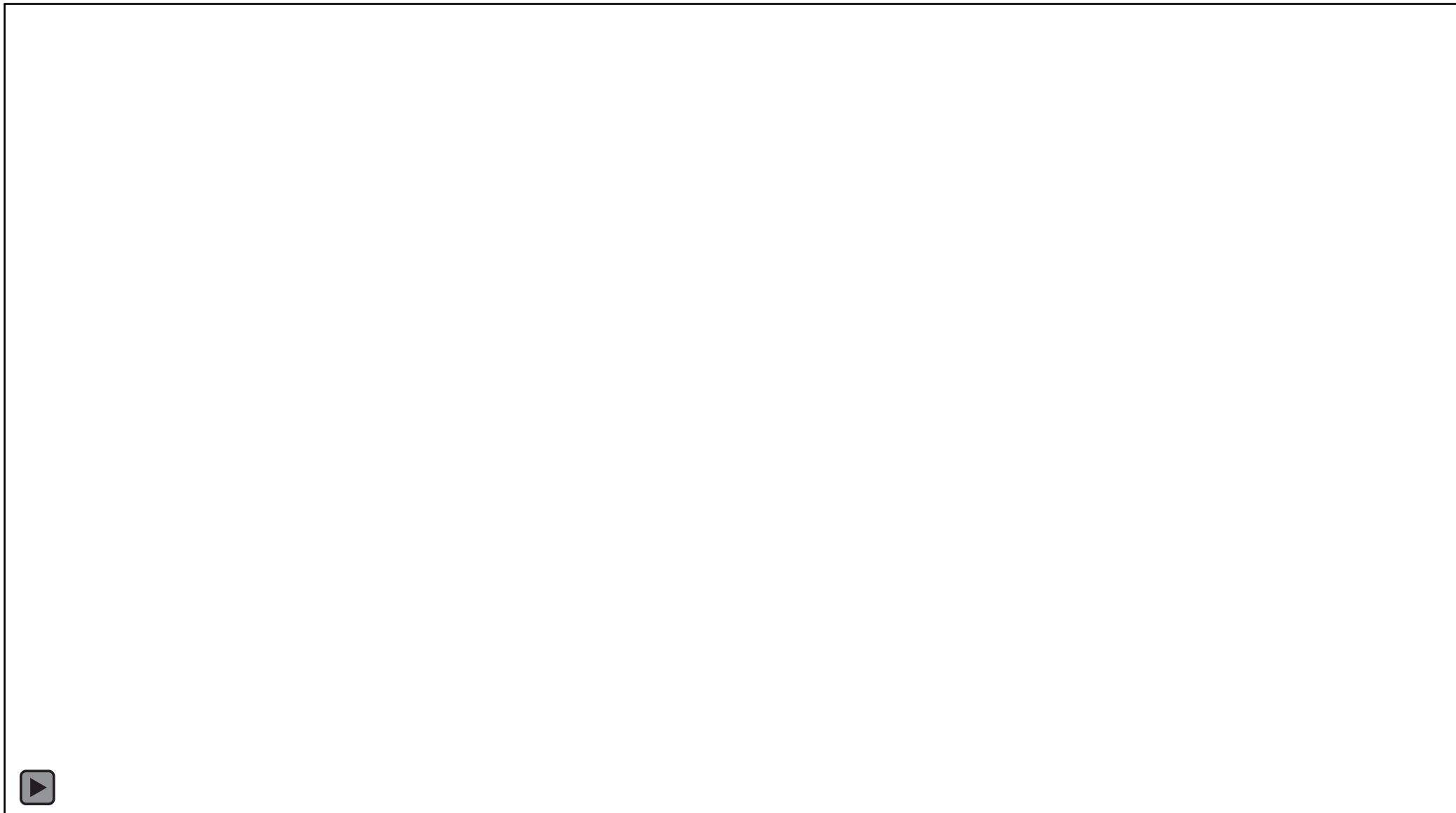


Game splash screen

- Most projects and games have a splash screen which appears while the game is loading. I think that this should be the case for my game, especially as the first initialisation of the databases takes around a minute.
- The design will be based off the tab icon (shown bottom right), but will be made as an SVG element so that it looks just as good on a large screen as a small screen.
- The splash will have animated text (Overshadowed – the game's name) and will fade out after a couple of seconds.



Game splash screen – testing



Credits screen

- As some of the content (such as audio and the big integer library) were not made by me, I need to credit the appropriate creators.
- This will be done in a credits screen.
- The credits themselves will be hyperlinks that the user can click on to visit a website in a new tab.

Credits screen – testing



Achievements

- A common aspect of games is earnable achievements. These are meant to encourage user exploration, and are a small reward for playing the game.
- I will add achievements which, when unlocked, will give a bonus to the player (in-game currency reward). This will make them more interesting for the user.
- Achievements need to be saved per user.

Achievements

```
function setupsaves() {
    var str = "(pid",
    str2 = "(pid",
    str3 = "(pid",
    n;
    for (n = 0; n < levels.length; n += 1) {
        str += ",l" + levels[n][0].id + "items,l" + levels[n][0].id + "time";
    }
    str += ")";
    for (n = 0; n < buyables.length - 1; n += 1) {
        str2 += "," + buyables[n].name.replace(/\ /g, "_");
    }
    str2 += ")";
    for (n = 0; n < achievements.length - 1; n += 1) { _____
        str3 += "," + achievements[n].name.replace(/\ /g, "_");
    }
    str3 += ")";
    db.transaction(function (tx) {
        tx.executeSql('CREATE TABLE SAVEPROGRESS ' + str);
        tx.executeSql('CREATE TABLE UPGRADES ' + str2);
        tx.executeSql('CREATE TABLE ACHIEVEMENTS ' + str3);
        initialisation_count += 1;
    });
    window.setTimeout(function () {
        createsavefile("3718274321299354", "ADMIN", true, false);
        initialisation_count += 1;
        window.setTimeout(function () {
            opensavesdatabase(true);
        }, 500);
    }, 500);
}, 500);
}
```

Creates the achievements table

Iterates through each achievement to add it to the table

Achievements

```
str5 = "(pid",
str6 = '(' + id + '',
obj, n;
for (n = 0; n < levels.length; n += 1) {
    str += ",l" + levels[n][0].id + "items,l" + levels[n][0].id + "time";
    if (admin) {
        str2 += ',"0","0"';
    }
    else {
        str2 += '","",""';
    }
}
for (n = 0; n < buyables.length - 1; n += 1) {
    str3 += ',' + buyables[n].name.replace(/\ /g, "_");
    str4 += ',"0"';
}
for (n = 0; n < achievements.length - 1; n += 1) {
    str5 += ',' + achievements[n].name.replace(/\ /g, "_");
    str6 += ',"true"';
}
str += ")";
str2 += ")";
str3 += ")";
str4 += ")";
str5 += ")";
str6 += ")";
db.transaction(function (tx) {
    tx.executeSql('INSERT INTO SAVEFILES (id,name,admin,qbits,reset) VALUE
    tx.executeSql('INSERT INTO SAVEPROGRESS ' + str + ' VALUES ' + str2);
    tx.executeSql('INSERT INTO SETTINGS (pid, key0, key1, key2, key3, key4
    id + "','"87","83","65","68","32","80","false","0.7","0.9","0.6","null"
    tx.executeSql('INSERT INTO UPGRADES ' + str3 + ' VALUES ' + str4);
    tx.executeSql('INSERT INTO ACHIEVEMENTS ' + str5 + ' VALUES ' + str6);
```

Inside the create user procedure

Iterates through the achievements array

Achievements

```
function getusersachievements(id) {  
    var output = {};  
    output.list = [];  
    output.qbits = 0;  
    db.transaction(function (tx) {  
        tx.executeSql('SELECT * FROM ACHIEVEMENTS WHERE pid=' + id + "", [], function (tx, results) {  
            var o;  
            for (o in results.rows[0]) {  
                output.list.push(results.rows[0][o]);  
                if (results.rows[0][o] == "false") {  
                    for (var i = 0; i < achievements.length; i += 1) {  
                        if (achievements[i].name == o.replace(/_/g, " ")) {  
                            output.qbits += Number(achievements[i].level) * 50;  
                            break;  
                        }  
                    }  
                }  
            }  
            output.list.splice(0,1);  
        }, null);  
    });  
    return output;  
}  
  
function updateuserupgrades(id, upgrade) {  
    db.transaction(function (tx) {  
        tx.executeSql('UPDATE UPGRADES SET ' + upgrade.replace(/\ /g, "_") + '=1' WHERE pid=' + id + "");  
    });  
}
```

Returns an array of the user's unlocked achievements

Iterates through the SQL query output to put it into a more manageable array

Iterates for each achievement in the array

Updates the SQL table when an achievement is unlocked

Achievements

```
function achievement_alert(id) {  
    document.getElementById("achievement_unlocked_achievement_name").innerHTML = achievements[id].name;  
    document.getElementById("achievement_unlocked_achievement_amount").innerHTML = Number(achievements[id].level) * 50;  
    play_sfx("user_achievement");  
    var temp_mod = document.getElementById("achievement_unlocked_popup");  
    temp_mod.classList.remove("fadeoutblur");  
    void temp_mod.offsetWidth;  
    temp_mod.classList.add("fadeinblur");  
    temp_mod.classList.remove("hide");  
    window.setTimeout(function () {  
        temp_mod.classList.add("achievement_unlocked_fadeout");  
        temp_mod.classList.remove("fadeinblur");  
        void temp_mod.offsetWidth;  
        temp_mod.classList.add("fadeoutblur");  
        window.setTimeout(function () {  
            temp_mod.classList.add("hide");  
        }, 750);  
    }, 4000);  
}  
  
bought = getusersupgrades(loggedin_user.id, loggedin_u:  
achieved = getusersachievements(loggedin_user.id);  
    achievements_reset();  
    add_achievement(0);  
    document.getElementById("achievement_unlocked_popup")
```

Generates a popup for when a user unlocks an achievement

Inside the log in script, gets the user's achievements

```
bought = bought.list;  
achieved = achieved.list;  
achievements_reset();  
add_achievement(0);  
document.getElementById("achievement_unlocked_popup")
```

Inside the log in script, updates the achievement display

Achievements

```
else if (collisions[i].classList.contains("lethal")) {  
    health -= 60 * damage_multiplier;  
    play_sfx("hard_hurt_collide");  
    add_achievement(15);————— An example of getting an achievement (when taken damage)  
    if (health > 0 && health < 10) {  
        add_achievement(18);  
    }  
}
```

1

An example of getting an achievement (when taken damage)

```
function add_achievement(achieve) {  
    if (achieved[achieve] == "true") {  
        achievement_alert(achieve);  
        updateuserachievements(loggedin_user.id, achievements[achieve].name);  
        loggedin_user.qbits += Number(achievements[achieve].level) * 50;  
        achieved.splice(achieve, 1, "false");  
        achievements_reset();  
        document.getElementById("shop_balance").innerHTML = "Balance: ☰" + loggedin_user.qbits;  
        if (loggedin_user.qbits >= 1000) {  
            add_achievement(6);  
        }  
    }  
}
```

Adds an achievement

Checks the user hasn't already got the achievement

Updates the shop readout

Achievements

Redraws the achievements menu/screen

```
function achievements_reset() {  
    var kill = document.getElementsByClassName("achievements_list_item"),  
        me, a, b, c;  
    while (kill[0]) {  
        kill[0].parentNode.removeChild(kill[0]);  
    }  
    for (var i = 0; i < achievements.length - 1; i += 1) {  
        me = achievements[i];  
        a = document.createElement("DIV");  
        b = document.createElement("DIV");  
        c = document.createElement("SPAN");  
        a.classList.add("achievements_list_item", "large_glow", "achievements_list_item_" + me.level);  
        b.classList.add("achievements_list_bg");  
        c.innerHTML = "<b>" + me.name + "</b><br>" + me.description;  
        b.appendChild(c);  
        a.appendChild(b);  
        a.setAttribute("id", "achievement_list_" + i);  
        if (achieved[i] == "true") {  
            a.classList.add("mini_locked");  
            document.getElementById("achievements_list_locked").appendChild(a);  
        }  
        else {  
            document.getElementById("achievements_list_unlocked").appendChild(a);  
        }  
    }  
    document.getElementById("achievement_list_1").addEventListener("click", function () {  
        add_achievement(1);  
    });  
}
```

Removes all achievement display elements

Iterates for each achievement in the array

Creates and appends the new achievement display elements

Achievements - testing



High score table

- Currently a user can only see their score and the overall record score, and nothing else. I wish to make the player be able to see everyone's score for each level.
- Rather than change the level selection screen which currently appears to work well, I will add a new screen for the leaderboard table, which will show everyone's scores in descending order.

High score table

```
function table_draw() {  
    var temp_mod = document.getElementById("scoreboard_div"),  
        kill = document.getElementsByClassName("scoreboard_record"),  
        level = mod(leveltime, levels.length),  
        times = getleveltimes("l" + level);  
    temp_mod.style.animation = "fade_out_blur_bg ease-in 0.3s 0s";  
    window.setTimeout(function () {  
        while (kill[0]) {  
            kill[0].parentNode.removeChild(kill[0]);  
        }  
        document.getElementById("table_level_name").innerHTML = levels[level][0].name;  
        highscore_draw(times);  
        temp_mod.style.animation = "";  
        void temp_mod.offsetWidth;  
        temp_mod.style.animation = "fade_out_blur_bg ease-out 0.3s 0s reverse";  
    }, 300);  
}
```

Redraws the highscore table

Removes currently displayed times/users

High score table

```
function highscore_draw(data) {  
    var a, b, c, time, parent = document.getElementById("scoreboard_table"),  
        len = data.length;  
    for (var i = 0; i < len; i += 1) {  
        a = document.createElement("TR");  
        a.classList.add("scoreboard_record");  
        b = document.createElement("TD");  
        b.innerHTML = data[i].name;  
        c = document.createElement("TD");  
        time = Number(data[i].time) / 1000;  
        if (typeof time == "number" && time != 0 && time != Infinity) {  
            c.innerHTML = time;  
        } else {  
            c.innerHTML = "----";  
        }  
        a.appendChild(b);  
        a.appendChild(c);  
        parent.appendChild(a);  
    }  
}
```

Draws the new items into the highscore table

Iterates for each item in the parameter array

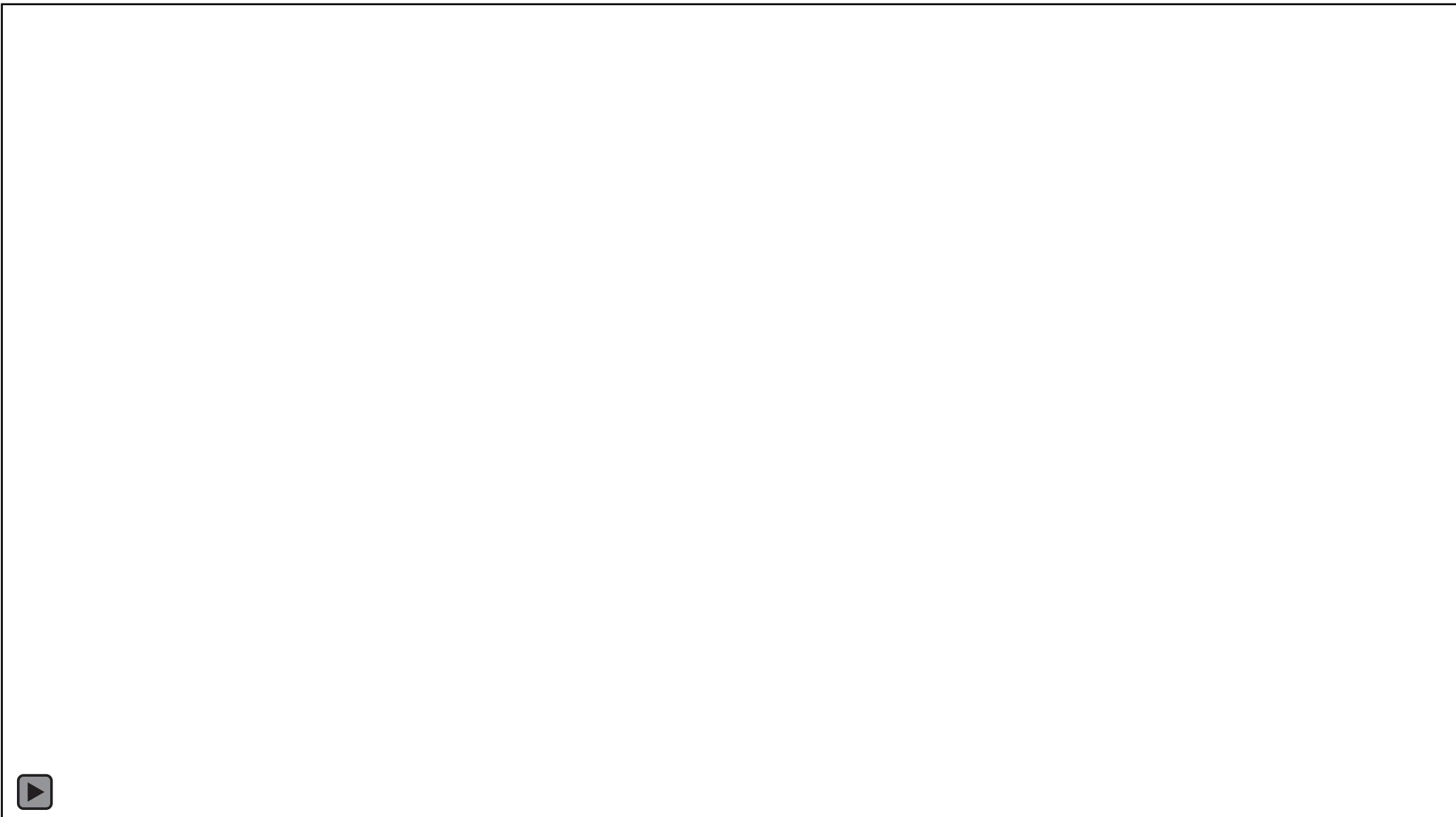
Creates an element for each item in the passed array

Ignores any undetermined times (including admin times)

High score table – testing



Delete account – testing



Capitalising constants

At the moment, constants are not capitalised. This means that external users may struggle to tell the difference between constants and variables. As a result, I am going to go through my code and capitalise all constants (such as the SVG canvas). Some examples are shown below.

```
var SVG = document.getElementById("canvas"),  
    Ambience = { //stores  
        base: /*the vario  
        Audio("assets/sou  
        air: [(new Audio(  
    },  
    Music = { //stores th  
        buzz_light: [(new  
            . . . ]
```

Regular expressions (regex)

My code makes use of regex in several places, mainly inside a replace function to replace all instances of a character/string instead of just one instance of the character/string. I found that this would be more efficient in terms of coding than using a for/while loop to repeatedly run the same command.

A few examples are shown below.

`(/shop_buy/g,` Selects all instances of the string “shop_buy”

`(/_/g,` Selects all instances of the _ character

`(/ /g,` Selects all instances of the space character

`(/\u2022/g,` Selects all instances of the special character character

Example variables

The code has too many variables to reasonably list out, as there are many loops that require a counter variable, and pointers to prevent multiple traversals of HTML's DOM structure which can be time consuming.

As a result, I am going to show a few examples of variables with a variety of data types and scopes, so that it can be understood that I have made use of different variable types.

Type // Hungarian	Name	Scope	Purpose	Example data
Array // rg	rgLevelsdata	Global	Contains the meta data fetched from SQL on each level	[["l1", "name", "description"], ["l2", "name", "description"]]
Array // rg	rgLogindetails	Global	Contains the basic user data fetched from SQL	[["4732843728", "ADMIN", TRUE], ["423", "User", FALSE]]
Pointer // p	pLoggedinuser	Global	Indicates which user is logged in	Logindetails[2]
Object // obj	objDimensions	Global	Stores the target dimensions for the svg canvas	{width: 1920, height: 1080}
Boolean // b	bDrawn	Global	Used to determine if the level has been drawn yet	TRUE
Integer // n	nLevel	Global	The selected level	2
Float // f	fXvelocity	Global	The player's x velocity	2.3452634789

Example variables

Type // Hungarian	Name	Scope	Purpose	Example data
String // dw	dwLevelname	Object	The name of the a level	"Lethal Maze"
String // dw	dwHashed	Local	The string returned from a username/password hash	"7643278432989496"
Pointer // p	pSVG	Global (constant)	"shortcut" to access the svg canvas element	HTML.body.svg
Integer // n	nCounter	Local	Used in a for loop for a counter	12
Float // f	fDuration	Class	Inside the audio element, the length of the audio play time	67.3443
Pointer // p	pSource	Class	Inside the HTML structure, the location of any external files (e.g. script, image, CSS)	HTML.source_folder.assets.sourceforge
Integer // n	nBits	Object	The number of bits in a level	125
Array // rg	rgAUDIO	Global (constant)	"shortcut" to audio elements	[file1, file2, file3, file4, file5]

Debug mode

For developer purposes, I have introduced a debug mode to the program. This prints function/procedure names to the console when they are run, as well as any SQL queries. It can only be triggered by a user by either modifying the source code (adding a debug=true statement) or injecting code via the console. This is because the output is printed to the console, so only a competent end user should be able to access the debug mode. It is not intended for general end user use. It is disabled by default as printing to the console causes a small amount of lag in the program, which can accumulate when printing many lines at once (e.g. iterative SQL queries). This means that the debug mode will not affect the general end user, and is only there for users who plan to modify, add or remove parts of the program. It is easy to add further debug logs as the script is a simple if statement, followed by a log procedure.

Debug mode

```
if (debug) {  
    console.log(levels[i][0]);  
}
```

Checks if the debug variable is true

Prints the list item to the console

```
function changesavefileadmin(id, admin, update) {  
    DB.transaction(function (tx) {  
        tx.executeSql('UPDATE SAVEFILES SET admin=' + admin + '' + WHERE id=' + id + ''');  
    });  
    if (debug) {  
        console.log(id + " admin now " + admin);  
    }  
    if (update === true) {  
        var obj = {},  
            i;  
        obj.id = loggedin_user.id;  
        obj.name = loggedin_user.name;  
        obj.admin = admin;  
        for (i = 0; i < logindetails.length; i += 1) {  
            if (logindetails[i].id == id) {  
                obj.qbits = logindetails[i].qbits;  
                obj.reset = logindetails[i].reset;  
                logindetails.splice(i, 1, obj);  
            }  
        }  
    }  
}  
}
```

```
    }  
    if (debug) {  
        console.log(str + "\n" + str2 + "\n" + str3 + "\n" + str4 + "\n" + str5 + "\n" + str6 + "\n");  
    }  
    if (update === true) {
```

A few examples of the use of the debug feature are shown left. As can be seen, the script is small and easily reusable, so does not inherently require much processing power (the reason that it slows down the program is the console display itself updating – not the JavaScript coding).

Debug mode

Developer Tools - http://127.0.0.1:16593/game.html

Elements Console Sources Network Performance Memory Application Security Audits

top ▾ Filter Info 1 item hidden by filters

18:48:30.831 ▾ Object [sql_connector.js:29](#)
 class: "obstacle_box"
 height: 1363
 id: "17"
 type: "rect"
 width: 125
 x: 5909
 y: 6078
 ► [__proto__: Object](#)

18:48:30.831 ▾ Object [sql_connector.js:29](#)
 class: "obstacle_box"
 height: 400
 id: "17"
 type: "rect"
 width: 125
 x: 7146
 y: 5678
 ► [__proto__: Object](#)

18:48:30.831 ▾ Object [sql_connector.js:29](#)
 class: "obstacle_box"
 height: 950
 id: "18"
 type: "rect"
 width: 172
 x: 4874
 y: 6216
 ► [__proto__: Object](#)

18:48:30.831 ▾ Object [sql_connector.js:29](#)
 class: "obstacle_box"
 height: 138
 id: "19"
 type: "rect"
 width: 1400
 x: 5046
 y: 7028
 ► [__proto__: Object](#)

18:48:30.831 ▾ Object [sql_connector.js:29](#)
 class: "obstacle_box"
 height: 950
 id: "20"
 type: "rect"
 width: 225
 x: 1271
 y: 6216
 ► [__proto__: Object](#)

Console

Example output in Google Chrome's developer tools' console of objects that are to be added to the SQL tables.

Extra debugging tools

Prints all SQL data on a given user

```
function userdump(id) {  
    DB.transaction(function (tx) {  
        tx.executeSql('SELECT * FROM SAVEFILES INNER JOIN SAVEPROGRESS ON id=SAVEPROGRESS.pid INNER JOIN SETTINGS ON id=SETTINGS.pid INNER JOIN  
UPGRADES ON id=UPGRADES.pid INNER JOIN ACHIEVEMENTS ON id=ACHIEVEMENTS.pid WHERE id="' + id + '";', [], function (tx, results) {  
            window.setTimeout(function () {  
                console.log(results);  
            }, 200);  
        }, null);  
    });  
}  
  
function allusersdump() {
```

The query to get all of the data

Prints all SQL data on all users

Extra debugging tools

```
allusersdump();
undefined
▼ SQLResultSet {rows: SQLResultSetRowList, rowsAffected: 0} ⓘ
  ▼ rows: SQLResultSetRowList
    length: 3
    ► 0: Object
    ▼ 1: Object
      Blue_Shadows: "1"
      Click: "false"
      Cyan_Shadows: "0"
      Ding: "false"
      Enhanced_Vision_I: "1"
      Enhanced_Vision_II: "0"
      Fan_appreciation: "false"
      Flash: "false"
      Flash_Cooldown_I: "0"
      Flash_Cooldown_II: "0"
      Flash_Cooldown_III: "0"
      Greater_Friction_I: "0"
      Greater_Friction_II: "0"
      Greater_Friction_III: "0"
      Green_Shadows: "0"
      Grim_Reaper: "false"
      Improved_Shielding_I: "0"
      Improved_Shielding_II: "0"
      Improved_Shielding_III: "0"
      Level_up: "false"
      Looking_good: "true"
      Lost: "true"
      Narcissist: "false"
      Near_Death_Experience: "true"
      New_Look: "false"
      Orange_Shadows: "0"
      Ouch: "false"
      Over_9000: "true"
      Overpriced: "true"
      Ping: "true"
      Pink_Shadows: "0"
      Qbit_Detector: "0"
      Rich_kid: "true"
```

Sample output from code injection via the console

```
userdump("232313656793219");
undefined
▼ SQLResultSet {rows: SQLResultSetRowList, rowsAffected: 0} ⓘ
  ▼ rows: SQLResultSetRowList
    length: 1
    ▼ 0: Object
      Blue_Shadows: "0"
      Click: "true"
      Cyan_Shadows: "0"
      Ding: "false"
      Enhanced_Vision_I: "0"
      Enhanced_Vision_II: "0"
      Fan_appreciation: "true"
      Flash: "true"
      Flash_Cooldown_I: "0"
      Flash_Cooldown_II: "0"
      Flash_Cooldown_III: "0"
      Greater_Friction_I: "0"
      Greater_Friction_II: "0"
      Greater_Friction_III: "0"
      Green_Shadows: "0"
      Grim_Reaper: "true"
      Improved_Shielding_I: "0"
      Improved_Shielding_II: "0"
      Improved_Shielding_III: "0"
      Level_up: "false"
      Looking_good: "true"
      Lost: "true"
      Narcissist: "true"
      Near_Death_Experience: "true"
      New_Look: "true"
      Orange_Shadows: "0"
      Ouch: "true"
      Over_9000: "true"
      Overpriced: "true"
      Ping: "true"
      Pink_Shadows: "0"
      Qbit_Detector: "0"
      Rich_kid: "true"
      Speed_Boost_I: "1"
      Speed_Boost_II: "0"
```

SQL table validation

Up until this point, the SQL tables have just been default text type for each record. This is somewhat silly as SQL has built in validation, which prevents the user from storing incorrect data types (such as text where there should be numbers). I have decided to make use of this built in feature, and (as a result) have also ensured that it is clear which column is the primary key. This will help to ensure that any further development on the SQL tables will be following my previously made structure.

However, due to the way that webSQL and JavaScript interact, I am having to use polymorphism to set all input types as JavaScript text that the SQL then converts to the appropriate types, and convert the output from any SQL queries to text so that the JavaScript will read them correctly. While this may appear to be self defeating, it is not as the primary purpose of input validation for the SQL tables still functions fully as expected. This change is useful in that it helps to improve the ACID (in particular the consistency and durability of data) attributes of my database, meaning that it is more reliable and secure for any future additions that may be made.

I am not hard coding any foreign keys as webSQL does not support them as it should, so it could cause more issues than it resolves. As my code is set up to already change attributes of each table as necessary, it does not seem worth the risk as my current methods all work as they should.

SQL table validation

```
initcommands = ['CREATE TABLE IF NOT EXISTS "RECTANGLES" ( `level` TEXT NOT NULL, `id` TEXT NOT NULL, `x` TEXT NOT NULL, `y` TEXT NOT NULL, `width` TEXT NOT NULL, `height` TEXT NOT NULL, `class` TEXT )', 'CREATE TABLE IF NOT EXISTS "LEVELS" ( `id` TEXT NOT NULL UNIQUE, `name` TEXT, `description` TEXT, `xspawn` TEXT NOT NULL, `yspawn` TEXT NOT NULL, `xexit` TEXT NOT NULL, `yexit` TEXT NOT NULL, `bits` TEXT NOT NULL, PRIMARY KEY(`id`) )', 'CREATE TABLE IF NOT EXISTS "SAVEFILES" ( `id` TEXT NOT NULL UNIQUE, `name` TEXT, `admin` TEXT, `qbits` TEXT, `reset` TEXT, PRIMARY KEY(`id`) )', 'CREATE TABLE IF NOT EXISTS "SETTINGS" ( `pid` TEXT NOT NULL UNIQUE, `key0` TEXT, `key1` TEXT, `key2` TEXT, `key3` TEXT, `key4` TEXT, `key5` TEXT, `spacebar` TEXT, `music` TEXT, `sfx` TEXT, `ambience` TEXT, `aesthetic` TEXT, `fps` TEXT )'];
```

Above: the updated create commands with their appropriate types and primary keys

Below: an example of a number parameter being passed into the SQL as a string (note the ' inside the ") despite being a number field

```
function updateuserbits(id, level, bits) {
  DB.transaction(function (tx) {
    tx.executeSql('UPDATE SAVEPROGRESS SET l' + level + 'items=' + bits + '' WHERE pid=' + id + "'");
  });
}
```

Below: an example of having to parse a string SQL output in an array to a number (integer)

```
qbits = getusersbits(loggedin_user.id);
levelshown = -1;
window.setTimeout(function () {
  level_selector_arrow(1);
  loggedin_user.qbits = 0;
  for (n = 0; n < qbits.length; n += 1) {
    loggedin_user.qbits += Number(qbits[n]);
  }
}, 200);
```

Functions – complexity

I have used a large number of functions and procedures in my project. For this section, I will be treating procedures as a type of function as they are effectively functions that do not return a value. To get all of the functions used in my code, I have injected a small amount of code via the JavaScript console to return a list of all of the functions used in my project. This will only return global functions but almost all of my functions are global (except for the Timer function, bigint library and built in functions such as for the audio or svg elements). I will exclude the built in procedures native to JavaScript as they are not something I have created, so there is no point commenting on their time complexity.

The below pages show a table of all functions, and also gives an worst-case time complexity. Some procedures will use a random number of loops inside them, so will not have a polynomial time complexity.

Due to the large number of functions, I have grouped them together by their general function (e.g. screen redraw) to make them easier to visualise.

Functions – complexity

```
function isFunction(functionToCheck) {
    var getType = {};
    return functionToCheck && getType.toString.call(functionToCheck) === '[object Function]';
}
var str = "",
    win_temp = Object.keys(window);
for (var i = 0; i < win_temp.length; i += 1) {
    if (isFunction(window[win_temp[i]])) {
        str += ("\n" + win_temp[i]);
    }
}
console.log(str);
```

Left: the code injected through the console to return all global functions.

Right: a sample of the output printed to the console from the injected code.

```
createcircle
createrectangle
createpolygon
createfilter
getRandomInt
create_bit
draw
centerplayer
checkcollision
clear
main_game_loop
clear_lethal
play_sfx
ambience_change
play_ambience
ambience_fadein
ambience_fadeout
play_music
music_change
ambience_fadein_music
ambience_fadeout_music
regen_button_sounds
button_click
button_hover
shadowrays
countdown
clear_cookies
highscore_draw
table_draw
set_key_popup
check_character_code
close_key_set_window
cssrules
css_getclass
shop_transaction_modify
shop_check_locked
shop_reset
reset_shadow_style
clear_all_forms
achievements_reset
add achievement
```

Functions – complexity

Group	Functions	Function/procedure	Time complexity (worst case) (k is an unknown variable)
Changing SQL	changesavefileadmin changesavefilepassword changesavefilename setsavefilereset updateusertime updateuserupgrades updateusersettings updateuserachievements updateuserbits removesave	Procedure Procedure Procedure Procedure Procedure Procedure Procedure Procedure Procedure Procedure	$O(1)$ $O(1)$ $O(1)$ $O(1)$ $O(1)$ $O(1)$ $O(1)$ $O(1)$ $O(1)$ $O(1)$
Reading SQL	dolevellocking opensavesdatabase getuserstimes getleveltimes getusersupgrades getuserssettings getusersachievements getusersbits getrecordtimes getlevel getlevelsdata leveltoobj	Procedure Procedure Function Function Function Function Function Function Function Function Function Procedure	$O(kn)$ $O(n)$ $O(kn)$ $O(kn)$ $O(kn)$ $O(kn)$ $O(kn)$ $O(kn)$ $O(kn)$ $O(kn)$ $O(kn)$

Functions – complexity

Group	Functions	Function/procedure	Time complexity (worst case) (k is an unknown variable)
Generating SQL	create_levels create_rects setupsaves regen createsavefile setup	Procedure Procedure Procedure Procedure Procedure Procedure	$O(kn)$ $O(kn)$ $O(kn)$ $O(kn)$ $O(kn)$ $O(kn)$
Audio	play_sfx ambience_change play_ambience ambience_fadein ambience_fadeout play_music music_change ambience_fadein_music ambience_fadeout_music regen_button_sounds button_click button_hover	Procedure Procedure Procedure Procedure Procedure Procedure Procedure Procedure Procedure Procedure Procedure Procedure	$O(k)$ Non-polynomial $O(kn)$ $O(k)$ $O(k)$ $O(kn)$ Non-polynomial $O(k)$ $O(k)$ $O(kn)$ $O(1)$ $O(1)$
Browser utilities	clear_cookies browser_check	Procedure Procedure	$O(1)$ $O(3)$

Functions – complexity

Group	Functions	Function/procedure	Time complexity (worst case) (k is an unknown variable)
Fullscreen interaction	request_enter_fullscreen request_close_fullscreen notfullscreen fullscreen checkfullscreen	Procedure Procedure Procedure Procedure Procedure	O(k) O(k) O(k) O(k) O(k)
GUI display	level_selector_arrow alertmsg achievement_alert dropdown_draw initialiseloading clearloading countdown highscore_draw table_draw set_key_popup close_key_set_window shop_check_locked shop_reset achievements_reset initialise	Procedure Procedure Procedure Procedure Procedure Procedure Procedure Procedure Procedure Procedure Procedure Procedure Procedure Procedure Procedure Procedure	O(k) O(k) O(k) O(k) O(k) O(k) O(k) O(k) O(k) O(k) O(k) O(k) O(k) O(k) O(k) O(k)

Functions – complexity

Group	Functions	Function/procedure	Time complexity (worst case) (k is an unknown variable)
Game setup	createcircle createrectangle createpolygon createfilter create_bit draw clear	Procedure Procedure Procedure Procedure Procedure Procedure Procedure Procedure	$O(k)$ $O(k)$ $O(kn)$ $O(kn)$ Non-polynomial $O(kn)$ $O(kn)$
Game events	use_flash Timer centerplayer main_game_loop checkcollision shadowrays clear_lethal	Procedure Procedure Procedure Procedure Function Procedure Procedure	$O(k)$ $O(kn)$ $O(k)$ Non-polynomial $O(kn)$ $O(kn)$ $O(kn)$
General utilities	levels_inner_length mod getRandomInt cssrules css_getclass clear_all_forms	Function Function Function Function Function Procedure	$O(kn)$ $O(1)$ $O(1)$ $O(kn)$ $O(kn)$ $O(kn)$

Functions – complexity

Group	Functions	Function/procedure	Time complexity (worst case) (k is an unknown variable)
User login	loggedin hash check_character_code	Procedure Function Function	$O(k)$ $O(kn)$ $O(k)$
Aesthetic appearances	shop_transaction_modify reset_shadow_style	Procedure Procedure	$O(kn)$ $O(k)$
Achievements	add_achievement	Procedure	$O(k)$

Testing

Testing

Please see the “Test Results” tab/sheet of the Gantt Chart spreadsheet for full details of my testing results. Below are hyperlinks to each appropriate figure.

<u>1</u>	<u>9.1</u>	<u>17.1</u>	<u>22.3</u>
<u>2.1</u>	<u>9.2</u>	<u>17.2</u>	<u>22.4</u>
<u>2.2</u>	<u>9.3</u>	<u>18</u>	<u>23.1</u>
<u>2.3</u>	<u>10.1</u>	<u>19.1</u>	<u>23.2</u>
<u>3</u>	<u>10.2</u>	<u>19.2</u>	<u>23.3</u>
<u>4</u>	<u>11</u>	<u>19.3</u>	<u>24</u>
<u>5.1</u>	<u>12.1</u>	<u>20.1</u>	<u>25</u>
<u>5.2</u>	<u>12.2</u>	<u>20.2</u>	<u>26</u>
<u>6.1</u>	<u>12.3</u>	<u>20.3</u>	<u>27</u>
<u>6.2</u>	<u>13.1</u>	<u>21.1</u>	
<u>6.3</u>	<u>13.2</u>	<u>21.2</u>	
<u>6.4</u>	<u>14</u>	<u>21.3</u>	
<u>7</u>	<u>15</u>	<u>21.4</u>	
<u>8.1</u>	<u>16.1</u>	<u>22.1</u>	
<u>8.2</u>	<u>16.2</u>	<u>22.2</u>	

Testing

Testing was initially completed by myself so that I could make the game usable and stable.

Now that this stage has been achieved, I will release various versions to people who could fit the end user profile for further testing.

This group of people are called the beta testing team.

The members will not be made aware of who the other beta testers are, as this way they will formulate their own opinion of the program without outside opinions.

There are 5 beta versions of the project, where the lowest (beta 1) has the fewest features. This allows me to see which features people do/don't like, and whether my later additions are suggested as features to be added.

Main menu – beta testing



Testing (finished product)

Testing element	Expected results	Final results	Comments
Fullscreen button	Will toggle fullscreen effect	Toggles fullscreen effect	
Initialisation count	Will increment up to 6, and on 6 will hide the loading screen	Works as expected	
Intro animation	Will have an animated intro	Works as expected	
Login with correct password	Will login to the correct user	Works as expected	
Login with incorrect password	Will fail to login	Login fails	
Login with correct password in wrong order (e.g. password as drowssap)	Will fail to login	User logs in	Change of hashing needed (character position needs to matter)

Testing (finished product)

Testing element	Expected results	Final results	Comments
Login with correct password in wrong order (e.g. password as drowssap) -- 2	Will fail to login	User fails to logs in	Now works as it should
Login with wrong username	Login will fail	Login fails	
Create new user with non-matching passwords	Registering will fail	Register fails	
Create new user with too short passwords	Registering will fail	Register fails	
Create new user with too short username	Registering will fail	Register fails	
Create new user with too short player name	Registering will fail	Register fails, but no alert is shown	Needs to show why it fails

Testing (finished product)

Testing element	Expected results	Final results	Comments
Create new user with too short player name -- 2	Registering will fail	Register fails, alert is shown	Now works as expected
Create new user with correct fields	New user registered	Works as expected	
First time login achievement	The user will get a new achievement the first time they log in	Works as expected	
First time login achievement on second/latter login	There will not be a new achievement	Works as expected	
Main menu buttons	Each button will redirect to the appropriate screen when clicked on	Works as expected	
Logout button	Logs user out and returns to the login screen	Works as expected	

Testing (finished product)

Testing element	Expected results	Final results	Comments
Reload webpage after first initialisation	Far shorter loading time	Expected results	
Changing levels	The level selection screen loops into itself infinitely	Expected results	
Level loading	The full level is generated	Expected results	
Play level countdown	The countdown timer shows when a level is played	Expected results	
Play level countdown fullscreen quit	The countdown timer resets if runs out before re-entering fullscreen	Expected results	
Play level countdown on fullscreen “spam”	As above	Failed – timer displays incorrect time value	Requires fixing as breaks critical game mechanic

Testing – fixing bugs: countdown

Whilst fixing this bug, I have come across scripting duplication so have also optimised this.

Red – original

Blue – updated

This will both reduce processing time as well as the file size.

```
if (document.getElementById("f11_popup_alert1").style.display == "none") {
    var n, timer = document.getElementById("timer_text"),
        pausable = document.getElementsByClassName("can_pause");
    timer.classList.remove("hide");
    timer.classList.add("fadein");
    if (reset === true) {
        time_begin = Date.now();
        time_pause = Date.now();
        document.getElementById("svg_circle_loader").classList.remove("hide");
        var sfx_volume = document.getElementById("sound_effects_slider"),
            temp_sfx_volume = sfx_volume.value;
        sfx_volume.value = 0;
        try {
            flash_timer.pause();
        } catch (err) {}
        flash_timer = undefined;
        use_flash();
        sfx_volume.value = temp_sfx_volume;
        try {
            flash_timer.pause();
        } catch (err) {}
    } else {
        time_pause -= Date.now();
        time_begin += Math.abs(time_pause);
    }
    play = true;
    for (n = 0; n < pausable.length; n += 1) {
        pausable[n].classList.remove("paused");
    }
    try {
        flash_timer.resume();
    } catch (err) {}
} else {
    window.setTimeout(function () {
        if (document.getElementById("f11_popup_alert1").style.display == "none") {
            countdown(reset);
        }
    }, 510);
}
```

```
else {
    var pausable = document.getElementsByClassName("can_pause");
    if (reset === true) {
        time_begin = Date.now();
        time_pause = Date.now();
        for (i = 0; i < pausable.length; i += 1) {
            pausable[i].classList.add("paused");
        }
    }
    try {
        flash_timer.pause();
    } catch (err) {}
    var sfx_volume = document.getElementById("sound_effects_slider"),
        temp_sfx_volume = sfx_volume.value;
    sfx_volume.value = 0;
    try {
        flash_timer.pause();
    } catch (err) {}
    flash_timer = undefined;
    use_flash();
    sfx_volume.value = temp_sfx_volume;
    for (i = 0; i < pausable.length; i += 1) {
        pausable[i].classList.add("paused");
    }
    try {
        flash_timer.pause();
    } catch (err) {}
    document.getElementById("svg_circle_loader").classList.remove("hide");
}

if (document.getElementById("f11_popup_alert1").style.display == "none") {
    var n, timer = document.getElementById("timer_text");
    timer.classList.remove("hide");
    timer.classList.add("fadein");
    if (reset === true) {
        time_begin = Date.now();
        time_pause = Date.now();
    } else {
        time_pause -= Date.now();
        time_begin += Math.abs(time_pause);
    }
    play = true;
    for (n = 0; n < pausable.length; n += 1) {
        pausable[n].classList.remove("paused");
    }
    try {
        flash_timer.resume();
    } catch (err) {}
}
```

Testing – fixing bugs: countdown

```
function notfullscreen() {
    document.getElementById("f11_popup_alert1").style.display = "block";
    play = false;
    if (document.getElementById("pause_screen").classList.contains("hide") && document.getElementById("countdown").classList.contains("hide")) {
        time_pause = Date.now();
        var pausable = document.getElementsByClassName("can_pause");
        for (var i = 0; i < pausable.length; i += 1) {
            pausable[i].classList.add("paused");
        }
        try {
            flash_timer.pause();
        } catch (err) {}
    }
}
```

I added the part highlighted in blue so the timer is only updated if the countdown is not already shown, as well as the menu not being paused, meaning that the bug is now fixed.

Testing – changing the generation mechanism

```
function create_levels() {
    db.transaction(function (tx) {
        var i;
        for (i = 0; i < levels.length; i += 1) {
            tx.executeSql('INSERT INTO LEVELS (id, name, description, xspawn, yspawn, xexit, yexit, bits) VALUES (?,?,?,?,?,?,?,?)', [levels[i][0].id, levels[i][0].name, levels[i][0].description, levels[i][0].xspawn, levels[i][0].yspawn, levels[i][0].xexit, levels[i][0].yexit, levels[i][0].bits]);
            if (i == levels.length - 1) {
                setupsaves();
                getlevelsdata();
                initialisation_count += 1;
            }
        }
    });
}
```

As the db.transaction command is asynchronous, it is faster to nest the for loop inside the function rather than the other way around. It is also quicker to use the ? inside values and then import from an array when running hundreds of queries.

I have changed the code to be in this format now for the initialisation stage at the beginning of the program as it is approximately 2000 times quicker (queries took around 0.4 seconds and now take 0.2 milliseconds).

This reduces loading time to around 10 seconds rather than 5 minutes on the first launch.

This method will also never fail to run a query as it is not asynchronous, so will not have multiple queries attempting to complete at the same time.

Testing – changing the generation mechanism

This procedure has also been modified as with the one on the previous page to make it more stable and faster.

```
| function create_rects() {
|   db.transaction(function (tx) {
|     var n, levelt, i;
|     for (n = 0; n < levels.length; n += 1) {
|       levelt = levels[n];
|       for (i = 1; i < levelt.length; i += 1) {
|         tx.executeSql('INSERT INTO RECTANGLES (level, id, x, y, width, height, class) VALUES (?,?,?,?,?,?,?,?)', [levelt[0].id,
|           levelt[i].id, levelt[i].x, levelt[i].y, levelt[i].width, levelt[i].height, levelt[i].class]);
|         if (i == levelt.length - 1 && n == levels.length - 1) {
|           initialisation_count += 1;
|         }
|       }
|     }
|   });
| }
```

Objectives – reliability (testing)

Objective	How to check if works/exists	Success and/or comments
Key pressing (which key)	Play level, move	Success
Username length	Try too short username	Success
Username characters	Try bad characters in username	Success
Password length	Try too short password	Success
Password characters	Try bad characters in password	Failed – see below
Password hashing	Check SQL tables for hashed password	Success
Password's matching (registration)	Try mismatching passwords	Success
Control setting (changing keys)	Change control scheme, play level	Success
Loading user data	Login with a pre-existing user	Success
Saving user data	Logout with changed user data	Success

Password hashing bug

As can be seen, the character order doesn't apply to the hashing algorithm, which is a major flaw. This needs immediate correction as it is a serious error with data security.



Password hashing bug fix

Code in the red boxes shows the original, and blue shows the new code. The position of the character is now multiplied by it's value, so position matters.

```
for (i = 0; i < a.length; i += 1) {
    if ((a[i].charCodeAt() > 47 && a[i].charCodeAt() < 58) || (a[i].charCodeAt() > 64 && a[i].charCodeAt() < 91) || (a[i].charCodeAt()
> 96 && a[i].charCodeAt() < 123)) {
        usern += Math.pow(a[i].charCodeAt(), 3);
    }
    else {
        alertmsg("Username can only contain upper/lower case letters and numbers");
        return null;
    }
}
for (i = b.length - 1; i > 0; i -= 1) {
    if ((b[i].charCodeAt() > 47 && b[i].charCodeAt() < 58) || (b[i].charCodeAt() > 64 && b[i].charCodeAt() < 91) || (b[i].charCodeAt()
> 96 && b[i].charCodeAt() < 123)) {
        passw += Math.pow(b[i].charCodeAt(), 2);
    }
}

if (i == 0, i < a.length, i += 1) {
    if ((a[i].charCodeAt() > 47 && a[i].charCodeAt() < 58) || (a[i].charCodeAt() > 64 && a[i].charCodeAt() < 91) || (a[i].charCodeAt()
> 96 && a[i].charCodeAt() < 123)) { //checks if the character is a-z, A-Z, 0-9
        usern += Math.pow(a[i].charCodeAt(), 3) * (i + 1) + i; //processes the username 1 character at a time
    } else {
        alertmsg("Username can only contain upper/lower case letters and numbers");
        return null; //prevents further script execution
    }
}
for (i = b.length - 1; i > 0; i -= 1) {
    if ((b[i].charCodeAt() > 47 && b[i].charCodeAt() < 58) || (b[i].charCodeAt() > 64 && b[i].charCodeAt() < 91) || (b[i].charCodeAt()
> 96 && b[i].charCodeAt() < 123)) {
        passw += Math.pow(b[i].charCodeAt(), 2) * (i + 2) + i;
    }
}
```

Password hashing bug fix (evidence)



Objectives – maintainability (testing)

Objective	How to check if works/exists	Success and/or comments
Meaningful variable names	Check new coder can understand variable purpose	Success
Meaningful function names	Check new coder can understand function purpose	Success
Comments in code	Check new coder knows what is happening in code	Limited success – code is also annotated in this documentation
Logical use of functions	Make sure not to make many useless functions that rely on other functions to create messy code	Success
Logical array structuring	Make sure arrays follow a sensible format that can be understood	Success
Backup code	Keep code backups in case of serious error	Success
Up to date documentation	Write documentation while coding	Success

Possible SQL injection

It has come to my attention that it is in fact potentially possible to inject SQL to the tables. This is a serious security issue as it means that players can add, remove, or modify data. The method of injecting SQL is through the player name field of the registration form. The username and password fields are not vulnerable as their contents are hashed, so are not input as their raw data to any SQL queries.

To fix this, I am going to prevent the user from using some characters that are typically used in SQL code (“ , ‘ , ` ; and -) which are not commonly used in player names. It will slightly reduce the availability of player names, but this is worth the cost for player security.

This fix is not entirely necessary as it would be very tricky for the user to create an SQL query that could cause any damage, as webSQL will only execute the first query in a string and ignore all later ones which is why I've had to use iteration to generate the rectangles and levels table. As most of the string for the SQL query is predefined, it is unlikely that a hacker could create anything that could cause damage to the table, but if webSQL were to be updated to allow multiple queries per string then there could be a problem, so this bug fix helps to future proof the project.

Possible SQL injection

Checks if the player name with removed characters still matches the original player name

```
if (pass1 == pass2 && user.length > 5 && pass1.length > 5 && uname.length > 2 && (uname.replace(/;|-|"|'|`/g,"") === uname)) {
```

```
} else if (uname.length <= 2) {
    alertmsg("Player name must be at least 3 characters long");
}
else if (uname.replace(/;|-|"|'|`/g,"") !== uname){
    alertmsg("Player name contains unauthorised characters");
}
else {
```

Checks if the player name with removed characters doesn't match the original player name

Alerts the user that an error has occurred

Evaluation

Evaluation – design

I believe that the program design is very close to what the final project looks like. This is good because it means that I have kept a consistent art style throughout the project, which has not been altered by CDS design department, showing that it is inherently user friendly. The most noticeable change is that all buttons have an icon on them which transitions to text on mouse over, whereas my initial design (diagrams) had only text on buttons, however was stated in my objectives as being something that would be worth adding. This change makes the game more appealing and intuitive as all buttons are simple in style, and unique to their text.

I also changed the arrows in the menu selection screen to be full-height rather than just normal buttons, as I felt that it would look more similar to currently existing games. I believe that it also looks much better than my original plan, as it fits into the theme better (all other buttons are rounded rectangles).

The shop has also been changed from arrows to a scrollbar, which is more easily recognisable than buttons and makes more sense as it is a list of items, so allows for faster and easier user navigation. Furthermore, the list of purchasable items was changed to be 1 button per element, which is more logical than a fixed number of buttons as it allows all to be displayed with fewer dynamic calculations.

The only other significant change has been that the player is no longer a circle, but a textured/filled square. This was because of collision calculations being based around a square shape rather than a circle, so meant that colliding with corners worked better and looked more accurate.

If I were to make any changes, it would be to use actual button elements instead of stylised div elements for the menus. This is because a user can use the tab button on their keyboard to navigate between buttons on the screen, but cannot do this with div elements. This would make the program more accessible to some users, especially if they had problems using a mouse.

Another change would be to the fullscreen detection algorithm, as currently the program will believe that the user is not fullscreen if they have zoomed in, as the screen size will not match the page size. This is a potential issue for users with sight impairments, as they cannot see as easily and cannot adjust for this.

Evaluation – design

My program has some very good usability features, such as the Snackbar alert that is commonly used to alert the user whether an action that they have attempted has been completed or not (e.g. for logging in, changing a password). The Snackbar is highly visible, and appears for a fixed duration, displaying text for the user. It is relatively unobtrusive as it only covers a small area of the screen, meaning that the user can ignore it and use the project if they so choose.

Another usability feature is the achievement notification. This is far more intrusive as it occupies the entire screen, and cannot be clicked “through”, meaning that the user must wait for the popup to vanish before continuing. It does not, however, interrupt game play as the user can still use the keyboard, so while it covers the screen, it is semi-transparent so the user can continue and complete a level, even while the popup is visible.

All buttons change appearance when the mouse is moved over them, which is a clear indication that the button is interactable. They then return to their normal appearance when the mouse is not touching them. Text box inputs also have a similar effect, but this is more visible when they are selected (clicked on). The buttons are normally black, and the dark glow around them changes size on mouse over. However, the confirm delete account button glows red, to show that the user is about to do a very major and irreversible action.

Evaluation – beta testing

User feedback has shown that the GUI style has never been a problem, as the only suggestion of a change between the open BETAs were that buttons should have icons as well as text (something that I had originally planned on adding but had considered low priority), but was made as more of a passing comment than a major issue.

In BETA 4 and 5, the new icons were commented on as being attractive, and no further suggestions were made to the GUI plan.

Users were easily able to identify what parts of the screen were text, buttons, input fields, sliders and scrollbars, showing that the design worked well.

The public BETA scheme proved to be highly informative as it allowed users on different platforms to test my program in different resolutions. This brought up issues such as qbits spawning inside obstacles, safari not being recognised as a valid browser, incorrect fullscreen keys for safari, and a major bug with restyling the CSS classes that caused a critical crash in some browsers.

It was also good in that people who weren't as experienced with computers were able to use the program and understand what it did and how to use it with little/no prior knowledge.

During the BETA period, there were few new additions, and users were given a changelog so that they could see what had been altered and comment on these features, which gave more targeted feedback.

Evaluation – beta testing

The BETA test program itself was not without flaws. People said that they would prefer to have the project available as a website so could save time by not having to download the files. I attempted to make this possible for BETAs 4 and 5 but it was not a success because there was a delay in audio playing. This was not a problem for background music as this did not have to play immediately upon being called, nor for ambience effects as all plays were delayed by the same amount so overall still looped. However, event dependant sounds (such as button clicks) being delayed by approximately 3 seconds made the project somewhat less engaging, so I chose to stick to the downloaded version for testers.

The online version can still be found at <http://pmeagles.azurewebsites.net/overshadowed/game.html> but has not been updated to the latest version as has been discontinued.

Evaluation – maintenance & limitations

The program itself is relatively future proof as there are currently no big competitors to internet browsers for displaying HTML, CSS and JavaScript content. Those that do exist are typically bespoke pieces of software intended for very niche markets, so are unlikely to ever hit the main user market. JavaScript has been around since 1995 and has continued to increase in popularity, replacing embedded Java and Flash content in almost all applications. This means that it is unlikely to be replaced any time soon, so this project should be able to run for quite some time. CSS and HTML have also both existed for considerable lengths of time, and, again, are unlikely to be replaced in the foreseeable future.

In terms of the coding itself, the project should continue to run for a considerable length of time as the features that it uses are not experimental, nor are they outdated. This, unfortunately, does not apply to the webSQL module which is, if anything, the least futureproofed part of coding. This is because webSQL has been deprecated, but considering this was in 2008 and the module has not been removed in the past 9 years, it is likely that it will remain in any browsers that already include it for quite some time. It is unlikely that any browsers will add support for the module, however. This means that if any of the target browsers cease to be in the market, then the project will stop working.

The programming techniques used should allow for future development and maintenance as there are comments in all of the CSS and HTML code, as well as the majority of the JavaScript code. Combined with this documentation, it should be sufficient for a programmer to modify, add or remove parts of the program confidently. Meaning function, procedure, class and variable names were chosen, so these should be easy to follow and understand meaning that further additions and changes are possible and should be easy to make.

Additions to the image and audio files should be very easy, as they simply need to be put into a folder and the file added to an array for the audio elements, and the image referenced as the source of any HTML image element. The image type I used (SVG) is relatively new but has a lot of support, and is considered to be a futureproof file type as it is becoming more and more used, as it can be shown at any size and preserve image quality. The audio files (ogg vorbis) are compressed but high quality audio files that take up little storage space. They are easy to produce, and there are many free programs to convert other file types to this type. However, few OS's have a built in player for the file type, so a developer would have to download and use freeware to play the file type on.

Evaluation – partial successes

Feature	Comment
Password hashing	This initially failed but has since been fixed, as the first algorithm/code did not take into account the location of each character, so was ignored at first. This was only noticed and fixed after the project had been completed, but was very critical.
Countdown timer pause	The original code for this failed because the timer was set each time the player paused the game/left fullscreen, which broke the timer if the countdown had already begun. This was fixed and a better solution implemented that avoided this bug.
Initialisation time	Originally the initialisation process for the first time the project was launched was not only incredibly slow, but occasionally failed an SQL query. This was because of asynchronous callbacks, which were removed and a faster (and more stable) system implemented instead.
User input	Very briefly, the user was not able to type in the input boxes as my keylogging system prevented keys from doing their normal commands. This was fixed immediately as it was a critical bug which prevented the game from working.
Highscore table	This sometimes fails to redraw correctly, but other times works fine. This implies that it may be called before other necessary processes have completed, but with the limited time remaining to complete the project, it is not possible to find the cause of the bug and correct it.
SQL injection	It was possible to inject SQL through the player name field. This potentially serious vulnerability issue has been fixed, but was not noticed until a very late stage of the project.

Further development

If I had had more time and resources, I would have liked to have made this project better suited to cross browser compatibility. As it is, the project does not work in Firefox and the Internet Explorer families, which are two of the most used browsers, which is due to the dependency on the webSQL implementation. This reduces my target audience somewhat, as people are unlikely to download a new browser just to use my project. I would have also liked to port the project to mobile devices, so remove the dependency on a keyboard (the game has also been tested on an iOS iPad with Bluetooth keyboard and ran the same as on a computer, so touchscreen capabilities seem to already be built in). A common way of doing this would be to have a virtual joystick overlay in the playing screen, allowing the user to use a thumb or finger to move around. It would not require too much extra to enable this porting, as JavaScript is already compatible with all common mobile devices (Apple, Windows and Android). However, if the product were to be standalone, it would most likely need to be made and compiled in a language that is best suitable for the processor of the device (such as C# for Windows applications). This would be far more complex as it would require an entirely new version for each programming language, and so may be difficult as it would take a lot of time and experience. Additionally, problems may arise that are language specific, so each platform may have a slightly different version to each other which could be an issue, as it may affect gameplay.

I would also have liked to have implemented the enemy AI feature that I had detailed in my initial objectives for the project. The reason that it was not implemented was because it would be time consuming to make, and would not have added sufficient usability to make it worthwhile. I do not feel that the enemy AI's would have been particularly computationally expensive, as they did not have any complex path-finding algorithms involved. The most complex would have been the "brute's" pathing around a predetermined graph, as it would have to calculate which node to return to if the player left its visible region and it returned from aggressive pathing to passive pathing.

If the project were to be released as a standalone, I would have also liked it to be able to compare the scores of users from other networks to their own via the internet. This would mean that the databases would be kept on a server. This would allow for the user to compare themselves to a global network, as well as access their save from devices in different networks.

Further development

Additionally, I would have liked to improve the password security, as it is currently limited to alphanumeric values (which is a limitation of JavaScript). Adding in symbolic values would have made usernames and password more secure. Furthermore, my hashing algorithm does not involve salt so is less secure than is a commonly accepted standard for data security. It would also be good to have a built in password strength indicator, which could do basic checks on whether there was a combination of numbers, symbols, upper case letters and lower case letters, and could be made more advanced to include a dictionary check to reduce the risk of hackers gaining access to secure data.

The levels themselves could also be expanded so that they could work with polygons and circles as was in the original plan. This was sadly not possible for me with my current programming knowledge, but is theoretically possible as the player could be made to check whether they were intersecting a diagonal line between 2 vertices of a polygon. However, this would be very computationally expensive, so could make the program run too slowly to be as usable as it currently is. If the program were in a compiled language, this would be less of an issue as they typically run much faster than JavaScript which is a translated programming language.

It would also have been good to keep the brightness and contrast sliders, and not have to remove them, but unfortunately they reduced framerate so much that it made the game unplayable. This is partly due to my laptop not having a proper graphics card, but as I was using it as a benchmark, it meant that it being slow on here made it not viable to keep in the game.

Bibliography

All used resources are listed below. This includes the websites used during the program's development. All resources used are allowed under creative commons, open license or with explicit permission of the creator. The files in the program should not be distributed or modified further than their use within the program.

- Development resources
 - Coding how-to [\[1\]](#) [\[2\]](#) [\[3\]](#)
 - Audio [\[1\]](#) [\[2\]](#) [\[3\]](#)
 - Library [\[1\]](#)
 - Images [\[1\]](#) [\[2\]](#)
- Document resources
 - Pac-Man [\[1\]](#)
 - Super Mario Kart [\[1\]](#)
 - Luminesca [\[1\]](#)