[User Interface 3](#_Toc116591481)

[Game Engine Market Research & competitor analysis 3](#_Toc116591482)

[Most popular engines for 2D 5](#_Toc116591483)

[Construct gaming engines 7](#_Toc116591484)

[Game engines that use LUA 8](#_Toc116591485)

[Educational game engines 10](#_Toc116591486)

[Applicable UX Laws, Principles and Guidelines 11](#_Toc116591487)

[Guidelines 11](#_Toc116591488)

[Laws 11](#_Toc116591489)

[Principles 12](#_Toc116591490)

[Theories - Fogg's behaviour model 13](#_Toc116591491)

[Business & user requirements 13](#_Toc116591492)

[Concept/Mind map 15](#_Toc116591493)

[Personas 15](#_Toc116591494)

[Card Sorting 22](#_Toc116591495)

[User Journey 24](#_Toc116591496)

[User Journey 1 24](#_Toc116591497)

[User Journey 2 25](#_Toc116591498)

[Site Map 27](#_Toc116591499)

[Website sitemap 27](#_Toc116591500)

[Game Engine sitemap 28](#_Toc116591501)

[Wire Frames 28](#_Toc116591502)

[User evaluation 31](#_Toc116591503)

[Low Fidelity 32](#_Toc116591504)

[Dark patterns (dp) game 36](#_Toc116591505)

[Medium fidelity 37](#_Toc116591506)

[Website homepage 38](#_Toc116591507)

[Website community page 39](#_Toc116591508)

[Documentation page 40](#_Toc116591509)

[Tutorial page 41](#_Toc116591510)

[About page 42](#_Toc116591511)

[Game engine editor 42](#_Toc116591512)

# User Interface

This report looks at the process of designing a prototype user interface for the proposed game engine(HorizonGE) of group 2.

# Game Engine Market Research & competitor analysis

This market review looks at popular and relevant game engines with mainly 2D game construction capabilities. The Game engines covered in this section include the following: GODOT, Unity, GameMaker, construct engines, solar2D, Love engine, Scratch and an academic research game called Simplified game engine (SGE).

GODOT is a community driven open-source game engine to help beginners get into game development, Godot focus’ mainly on the ability to create 2D games. In building the game engine, third-party APIs such as OpenGL ES, a subset of openGL used for embedded systems like smartphones a video game consoles (Wikipedia Contributors, 2022b), Godot also now supports vulkan (Godot Engine, 2021) which is used to render the custom GUI toolkit are used. Little information is provided on other main feature of the godot engine but some recon on their open source github (godotengine, 2022) would suggest in-house physics and audio libraries. The scripting languages available to use are GDScript or C#.

The Unity game engine provides a strong solution to users looking to create 2D games. The engine has a beginner friendly onboarding process and provides superior graphics to the likes of Godot (Yuriy Denisyuk, 2022). It is regarded as one of the best engines to consider when choosing a game engine to build a 2D game (Doucet & Pecorella, 2021). Scripting can be done with C#.

GameMaker, another popular game engine tries to also target novice users with their drag and drop style editor while also providing scripting capabilities with their own Gamemaker language (GML). Third-party APIs such as Direct3D for windows and OpenGL for mac and Linux are employed on this engine (Wikipedia Contributors, 2022c).

Doucet & Pecorella (2021) shows Unity to be the most popular game engine through the quantity of games on Steam built using Unity, notably also showing GameMakers’ popularity. Yuriy Denisyuk (2022) shows why Godot may also be a suitable choice to use as a 2D game engine as it provides industry strength tools and beginner friendly practices.

Following on from the most popular game engines for 2D, Solar2D an LÖVE engine are included in this review as these two engines make use of Lua as their scripting language. There is evidence in the industry of gaming engines that due to its concise nature and easy learning, Lua can be a good choice when it comes to selecting a scripting language for a game engine. (Ogunleke, 2022; stackoverflow, 2011).

The construct family of engine is a good case as the most recent release of the construct engine has been moved from a desktop application with construct classic and construct 2 to the browser with construct 3. Construct classic is written in C++ while construct 2 uses NW.js to create its desktop application. Both of these engines are still available for download but have since been retired in 2013 and 2020 respectively. Construct 3 however is fully supported by Scirra, an in-browser game engine written in JavaScript, uses a drag and drop GUI with optional JavaScript programming. (Wikipedia Contributors, 2021)

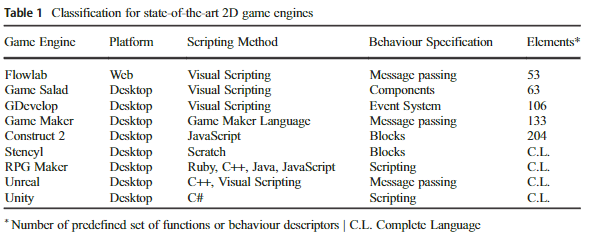
Solar2D is an open-source engine with GitHub source code provided below, solar2D is an official fork of the previous corona SDK which is no longer commercially supported. Solar2D layers integrated Lua scripting on top of C++/OpenGL to build its graphic applications, while also providing extensive API calls for animations, graphics, audio, box2D physics and networking. (Wikipedia Contributors, 2022).

The LÖVE engine is another open-source cross-platform game engine providing Lua as the scripting language and built with C++. The LÖVE engine uses third-party APIs such as Box2D for physics and OpenGL and SDL for video and sound. Free code camp provides a beginner's tutorial with Love2d game engine and lua (Carnes, 2022), however love2d is an API framework with lua and has no user interface (LÖVE - Free 2D Game Engine, 2022).

Scratch is not a game engine, but a programming language aimed at teaching programming to teenagers. The language is a drag and drop block like structure and comes built into the online editor. The learning process is encouraged through game build exercises. The online editor is broken down into the “block palette”, “the coding area” and “the staging area”. The block palette houses all the different coding blocks available to the user, the coding area is where the user connects different blocks to build their program and the staging area is where the program runs from.

(Wikipedia Contributors, 2022c)

The final engine reviewed is the simplified game engine (SGE) academic research by (Chover et al., 2020). In this research paper the authors review some of the current market solutions as of 2020 and provides the table shown below:



(Chover et al., 2020)

Chover et al. (2020) also justify the position of their research of creating a game engine with reduced set of actions and conditions “*Even though these efforts are very significant steps forward towards the game development complexity reduction, the use of this kind of software still requires high technical profiles and specific training, thereby excluding most of the potential users of these technologies”.* Their research showed that for student with little to no programming using their SGE of reduced rule sets and conditions it was easier to grasp the challenges of programming a 2D video games.

There are a lot of other heavy weight industry strength gaming engine used by game making companies, most of these engines were designed and built in-house for the purposes of the game developers to make commercialised games, some of these engines are also licensed out providing additional income. A list of some of these game engines can include Epic Games’ - Unreal Engine, The Quake family of engines (Quake III, medal of Honour), Half-life source engines (Half-life 2, portal), Rockstar Advanced gaming engine (RAGE) (Grand Theft Auto & Red Dead Redemption series’) and CRYENGINE (RYSE: Son of Rome) (Gregory, 2019, pp. 31-37). These engines however provide full scale AAA project solutions which goes beyond the scope of this project.

A granular breakdown of each engine surveyed can be seen below:

## Most popular engines for 2D

**GODOT**

Open source: Yes - [github](https://github.com/godotengine/godot) (godotengine, 2022)

Monetary model: Free to use

2D or 3D: 2D mainly but also 3D

Level of coding: Mix of user graphical interface and GDScript coding.

Learning curve: GODOT is novice and beginner friendly.

Advantages: More dedicated to 2d development, easy to get going, focus on simplicity

Disadvantages: GODOT is trying catch up with unity in most aspects (performance, Visuals)

Languages:

Main: C++

Scripting: GDScript (similar to python), C#

CPU: although inferior to the likes of unity, still has quite high performance in part due to the scene-based approach, meaning all aspects of a game are dissected into scenes, can be loaded, and unloaded as necessary, keeping memory usage low. (Yuriy Denisyuk, 2022)

Graphics: GODOTs graphics for 2D games are not considered to be as strong when compared to the likes of unity and the GODOT shaders are inferior (Yuriy Denisyuk, 2022)

Third-party API’s: For rendering Godot uses GLES3, this maps to OpenGL 3.3 on desktop, OpenGL ES 3.0 on mobile and WebGL 2.0 on the web, (Godot docs, 2022a), citing that in the long run it is easier to use these implementations (Linietsky, 2017). OpenGL and Vulkan are supported more than Direct3d to maintain cross-platform compatibility and open standards.

Godot uses its own GUI Toolkit that is rendered using OpenGL or Vulkan to render. (Godot docs, 2022b)

Godot has its own in-house physics engine and audio engine. (godotengine, 2022)

Mobile/pc/console: Create games on windows, mac & Linux. Cross platform deployment on most platforms.

Features: Scene based approach, node-based user interface, visual editor.

Review: good for simple beginner games and getting to grips with game dev

**UNITY**

Open source: No

Monetary model: Free tier

2D or 3D: 2D and 3D

Level of coding: Minimal to none, can use C#.

Learning curve: Novice friendly but has a bigger learning curve due to the greater quantity of features. (Yuriy Denisyuk, 2022)

Advantages: cross –platform support, performance optimisation, asset store, package creation

Disadvantages: a lot of features still behind $1500 paywall

Languages:

Main: C++/C#

Scripting: C#, drag’n’drop gui

CPU: high performance capable of AAA level games with high-quality graphics, powerful rendering engine.

Graphics: offers superior visuals for 2D to the like of Godot.

API’s: In-house “OctaneRender” engine (Unity Technologies, 2017). Also supports OpenGL, Vulkan, DirectX, Havok

Mobile/pc/console: Unity editor is supported on windows mac and Linux, supports building games for more than 19 consoles.

Unique features: Asset store,

Review: Unity good for beginner and bigger scale projects if you are looking for higher scale performance and visuals

**GameMaker**

Open source: No

Monetary model: free to use

2D aspects: Mainly 2D

Level of coding: minimal, drag and drop style

Learning curve: novice

Advantages: strong sprite based 2D capabilities rivalling Godot and unity, speed, simplicity, cross platform, good for debugging

Disadvantages: small ecosystem, limited assets available, needs more features

Languages:

Main: C++/c#

Scripting: GameMaker language (mashup of JS & C++) (Game Designing, 2022).

Graphics: primarily designed for 2D, makes use of “out-of-the-box” raster graphics, vector graphics and 2D skeletal animations. Also includes a standard library for drawing 2D primitives.

API’s: Direct3D on windows, OpenGL on mac and Linux, OpenGL ES on Android and IOS

Mobile/pc/console: cross-platform editor and games

Unique features: Known for pixel art games

## Construct gaming engines

**Construct classic**

Open source: yes - [github](https://github.com/aolko/construct) (aolko, 2022)

Monetary model: Free

2D or 3D: 2D

Level of coding: minimal has GUI

Learning curve: minimal

Advantages: Free, IDE

Disadvantages: Discontinued for newer versions

Languages:

Main: C++

Scripting: drag and drop

CPU: industry quality performance

Graphics: 2D pixel genre

API’s: DirectX

Mobile/pc/console: solely windows due to reliance on DirectX

Features: IDE

**Construct 3**

Open source: No

Monetary model: Monthly subscription

2D or 3D: 2D

Level of coding: minimal has GUI

Learning curve: minimal

Advantages: Free trial no signups, IDE

Disadvantages: monthly licence

Languages:

Main: JavaScript

Scripting: drag and drop, optional programming in JS

CPU: responsive in browser IDE, industry quality performance

Graphics: 2D pixel genre

API’s: DirectX

Mobile/pc/console: solely windows due to reliance on DirectX

Features: IDE, useful guided tour for free, good for beginners

## Game engines that use LUA

**Solar2D**

Open source: Yes - [github](https://github.com/coronalabs/corona) (coronalabs, 2022).

Monetary model: Free

2D or 3D: 2D

Level of coding: Need to know Lua

Learning curve: learning Lua

Advantages: complete list of API calls and templates (incl. Audio and graphics), in browse compiler to try out the engine, Lua is relatively easy to learn, it’s free, fast development

Disadvantages: closed source, requires internet connection for a device build, no visual editor

Languages:

Main: C++

Scripting: Lua

CPU: industry strength 2D game engine

Graphics: good 2D graphics

API’s: Box2D, OpenGL

Mobile/pc/console: mainly mobile applications for android and iOS, but also desktop apps for windows, mac, and Linux

Features: Used for its extensive range of API calls for 2D games

**LÖVE engine**

Open source: Yes – [github](https://github.com/love2d/love) (love2d, 2022)

Monetary model: Free

2D or 3D: Only 2D

Level of coding: Need to know Lua

Learning curve: learning Lua

Advantages: cross-platform

Disadvantages: Lack of power, not as many modules as others, no GUI, no 3D

Languages:

Main: C++

Scripting: Lua

CPU: industry strength 2D game engine

Graphics: good 2D graphics

API’s: Box2D for physics, OpenGL, OpenGL ES and SDL for graphics and SDL for audio and input

Mobile/pc/console: mainly mobile applications for android and iOS, but also desktop apps for windows, mac, and Linux

Features: OpenGL support, touchscreen & UTF-8 support, luasocket and lua-enet libraries for network communication TCP/UDP

**Scratch**

Open source: Yes - [github](https://github.com/LLK/) (Scratch, 2022)

Monetary model: yes

2D or 3D: 2D

Level of coding: Designed for individuals with no coding experience

Learning curve: none, designed to learn game development

Advantages: no coding experience is necessary, minimalist, great for learning the basic concepts of programming

Disadvantages: low computational speed, low-res graphics

Languages:

Built: HTML 5 & JavaScript

Scripting: Drag’n’drop blocks into visual editor

CPU: Designed for making basic games, slow speeds, low frame rates

Graphics: Basic 2D game graphics

API’s: In-house

Mobile/pc/console: PC

Features: Learning game development environment

(Wikipedia Contributors, 2022c)

## Educational game engines

**Simplified Game Engine SGE**

Open source: No

Monetary model: N/A

2D or 3D: 2D

Level of coding: Designed for individuals with no coding experience

Learning curve: none designed to learn game development

Advantages: no coding experience is necessary

Disadvantages:

Languages:

Main: Not specified, judging from there use of Box2D it looks like C++ is involved

Scripting: reduced set of actions and conditions in the SGE editor

CPU: Designed for making basic games

Graphics: Basic 2D game graphics

API’s: Box2D, Matter, pixi, in-house rendering engine

Mobile/pc/console: PC

Features: Learning game development environment

(Chover et al., 2020)

# Applicable UX Laws, Principles and Guidelines

Before moving onto the discussion of the business and user requirements of the game engine we can discuss some UX laws to keep in mind when trying to design the user interface.

## Guidelines

Guidelines for navigation:

Breadcrumbs – Breadcrumbs assist the user in navigation, this navigation may be useful when the user is navigating the games file structure. (Lin, 2017)

Consistency – Consistency and familiarity with layout can help users get to grips more quickly and find the option they are looking for. (Lin, 2017)

Tabs - We will use tabs as a model for navigations providing user options within the interface and website. (Nielsen, 2016)

Search bar – A search bar can provide a quick work around for the user looking for a particular integrated option that may be deeper than surface level. (Babich, 2017)

Call-to-action button – Having a call-to-action button provides a clear message to the user of the product on offer and how to engage with it, in our case we want the user to download the HorizonGE game engine and or the dark patterns game. As Messaki (2020) *“A call to action is not a button. It's an invitation to get what you need”.*

## Laws

Hick’s Law –Hick Law states that the more choices available to the user, the more time it takes for the user to decide. This law may be applicable to the game engine UI as the intended user is to be novice game developer, reducing the complexity choice can allow the game developer to pick up the necessary basics quicker and focus on the creativity of the game play. (Mads Soegaard, 2020)

Jakob’s Law – Users want familiarity, most people nowadays will have some familiarity with computer programs as they have become a necessary part of life, they will be familiar with particular layouts such as toolbar ribbons in the document editors and in websites. With this in mind, we can think back to some of the guidelines discussed. Tabs can be used as a familiar form of navigation to users and in turn providing consistency. (Nielsen, 2000)

Law of proximity – states that things that are close together, tend to be grouped together by the human eye. The law of proximity falls into gestalt psychology which notes that humans have a natural inclination to find patterns in the world around them (Fitzgerald, 2022). We can keep this I mind when implementing our card sorting exercises later and further into designing the website.

## Principles

Nielsen’s usability principles can be kept in mind when designing our user interface:

1. **Visibility of system status** - Implementing a logging system and file explorer functionality will allow the user to maintain an idea of the stage that they’re at and the processes being implemented.
2. **Match between system and the real world** – The idea behind using a drag and drop to piece together block like structure allows the user to make connection to the ‘natural mapping of real-world puzzle pieces.
3. **User control and freedom** – Learning programming and game development can be an intense undertaking and part of the process is problem solving of error. Providing the user with different aspects of control over their program can help alleviate some of these issues. Community chat boards and documentation may provide answer to some problems along with undo and redo buttons to provide emergency exit to the user.
4. **Consistency and standards** – We can link this principle to Jakob’s Law, where the user may have tried an editor of a programming language or tried out another game engine like unity or Godot mentioned above, it is with this familiarity that might guide the user towards staying with this game engine. Alternatively, as our game engine is intended towards beginner maintaining industry standard and familiarity may help our target user to transfer their acquired skills across to bigger scale and more complex gaming engines.
5. **Error prevention** – Part of learning to program and develop games is encountering errors and problem solving to find solutions to these issues. By providing the new game developer with a good logging system and clear errors we can help the game developer to develop this skill set to increase their abilities.
6. **Recognition rather than recall** – Using a clear layout for the user interface can help the user identify the option or tool needed faster, as opposed to recall. The reduced learning of clear layout structure can ease the learning curve of the game developer.
7. **Flexibility and efficiency of use** – With flexibility and efficiency of use, this may be out of scope for this project, however deeper level functions for efficiency of use may be considered for future growth of the game engine.
8. **Aesthetic and minimalist design** – Aesthetic and minimalist design will help maintain linear train of thought and minimise confusion when the game developer is undertaking the learning curve.
9. **Helping users recognise and recover from users' errors** – This principle maintains similarity to principles three and five as Documentation, Community chat boards and clear logging systems will help the game developer retrace their errors and find the solution to their problem.
10. **Help and Documentation** – Finally, we have mentioned how help and documentation will be provided to the Game Developer.

(Nielsen, 2020)

The above principles of Jakob Nielsen will be of great value when implementing them into group two’s proposed user interface for the game engine. Due to the nature of creating a game engine for beginner game developers, encountering problem solving issues is a necessary part of programming and build games, however, applying these guidelines and principles to the game engine can layout and present these problems in a clear and helpful manor allow the user to build and develop their skills for when they make the leap to higher steppingstones.

Theories - Fogg's behaviour model

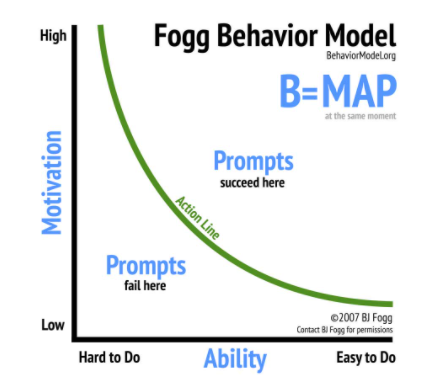


Figure by BJ Fogg retrieved from behaviourmodel.org

Additionally, we can look at the Fogg behaviour model. The model suggests that for someone to do something they need the motivation, the ability and a trigger to do so. In the case of the game engine, the trigger is to learn game design, their ability may be low as they do not know where to start with other game engines. The user may have some motivation as they would like to learn a new skill. When applying this to our game engine, we hope to increase the user's ability by reducing barriers to entry and maintaining good motivation for the user to increase the chances of success.

(Fogg, 2022)

# Business & user requirements

Having looked at some of the 2D solutions that the market offers, we can now look at some of the business and user requirements for this project. The requirements are specific to users who are beginners to game development.

1. The user must have the ability to create and manage their own 2D games projects (file explorer).
2. The user must have the ability to work with a GUI interface.
3. The game engine should have drag and drop style features to implement the different programming basics.
4. The user must have the ability to call API functions.
5. The Game engine will provide game middleware capabilities form open libraries to cover sound, graphics, animations, physics.
6. The game engine should be able to build games offline.
7. The Game engine must have scripting capabilities (C#, Lua).
8. The game Engine should allow the user to implement AI into the game being built.
9. XY coordinate system/ a design plane.
10. The game engine should have scene management capabilities
11. The game engine should have an assets store (Sprites, Tilemaps)
12. The game engine should have a particle system.
13. Game Engine should have ability to draw objects within the Engine.
14. Game Engine should allow events to trigger action (Eg. If hit enemy character dies)
15. Game Engine should create an executable game file.
16. Game Engine should be able to give objects different capabilities (solid wall, pass through etc.)
17. Game Engine should be able to import objects from the system locally.
18. Game Engine should have a community section on its website that will allow members of the community to collaborate on projects.

**Prioritisation chart**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Req. ID | Name of Req. | Description | Priority | User Contact |
| 1 | Project management |  | P0 | Game developer |
| 2 | GUI |  | p0 | Game developer |
| 3 | Drag and drop coding chunks |  | P0 | Game Developer |
| 4 | API calls |  | p3 | Game developer |
| 5 | middleware | Third-party APIs (OpenGL etc...) | P0 | Game developer |
| 6 | Offline | Build games offline | P0 | Game developer |
| 7 | Scripting | Scripting languages | P0 | Game developer |
| 8 | AI |  | P1 | Game developer |
| 9 | XY coordinate system/ design plane |  | P0 | Game developer |
| 10 | Scene management |  | P0/P1 | Game developer |
| 11 | Asset Store | Sprites tile maps | P1 | Game developer |
| 12 | Particle system |  | P3 | Game developer |
| 13 | Draw objects |  | P1 | Game developer |
| 14 | Trigger event | Hitting an enemy | P1 | Game developer |
| 15 | Executable game file |  | P0 | Game developer |
| 16 | Objects have different capabilities |  | P0 | Game developer |
| 17 | Import assets | Import objects locally | P0 | Game developer |
| 18 | Community support | Section on website to dedicated to community support, wiki & docs | p0 | Game developer |

Key

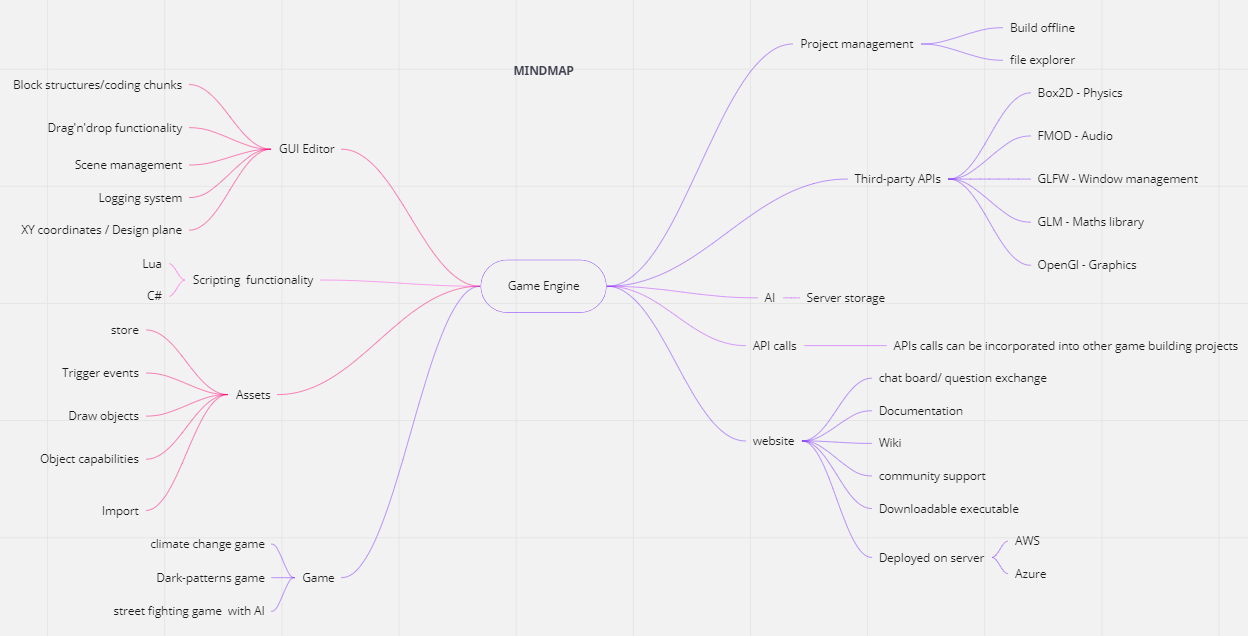
P0 Needed to demo

P1 Needed to launch

P2 Needed to sell

P3 Needed to grow

# Concept/Mind map



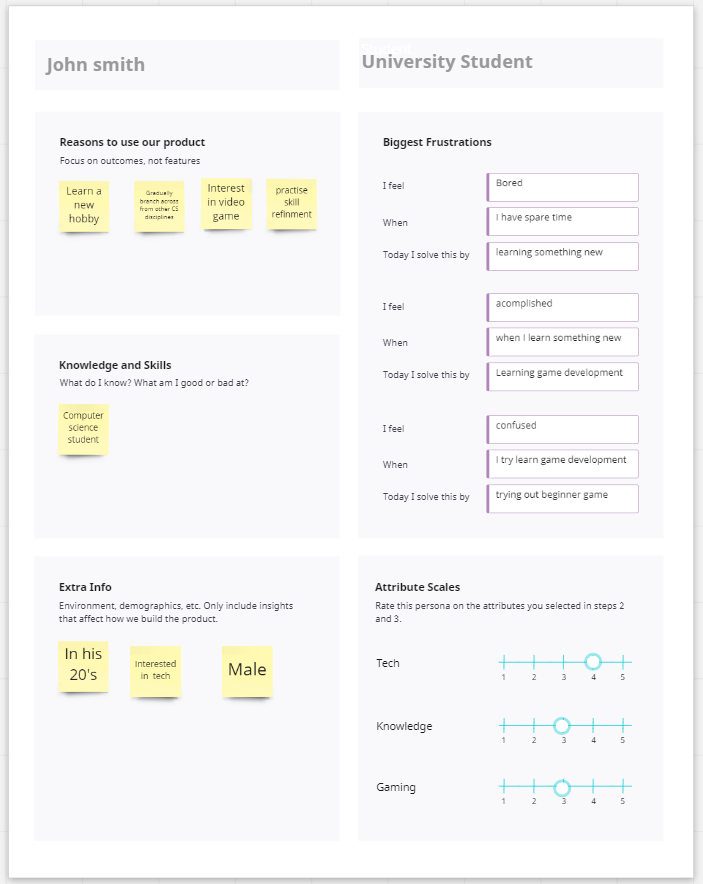
Above we can see a Mind Map tool used to generate and pull creative & non-creative ideas. Using the mind map allows us to visualise all the aspects described in the business and user requirements while also containing other related ideas. The next process when designing the user interface is to introduce a user persona.

Personas

Below we can see the first persona of a person who might use our proposed game engine. John smith is a student in his 20’s who might be studying computer science or a similar field. John might be looking for a new hobby in his spare time. As he used to be a gamer in his childhood, John thinks it might be useful to try to put to the test some of his programming knowledge and combine it with his interest in computer games to create his own game. Not knowing where to start, John googles game making for beginners and stumbles across the community for our game engine. John decides to give the horizon game engine a go as it seems to be a good place to learn the basic functionalities of a game engine. He downloads the executable file from the website. Having downloaded from the website, he realises the is also a game that can be played based on something called ‘dark-patterns’, intrigued John plays the game to find out more. He learns about ‘dark-patterns’ in UX design through playing the game.



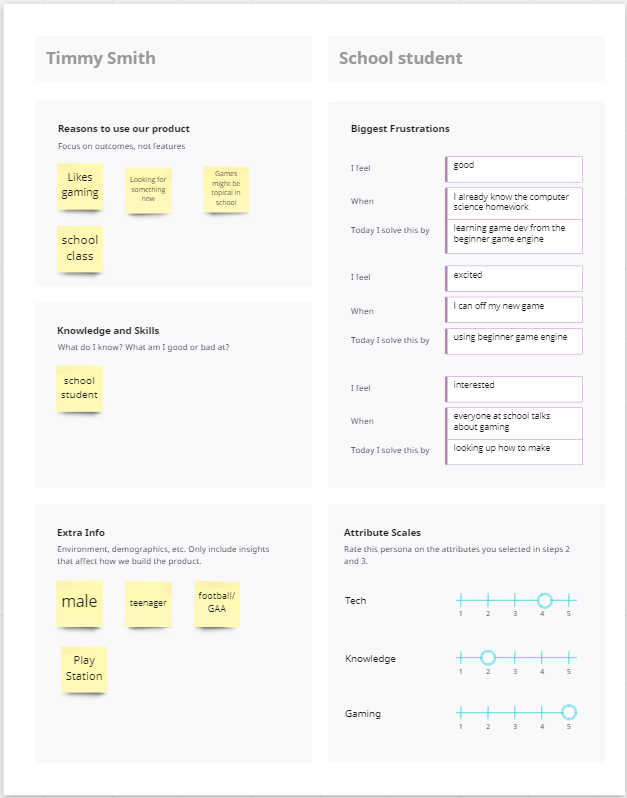
(Castanié, 2018)



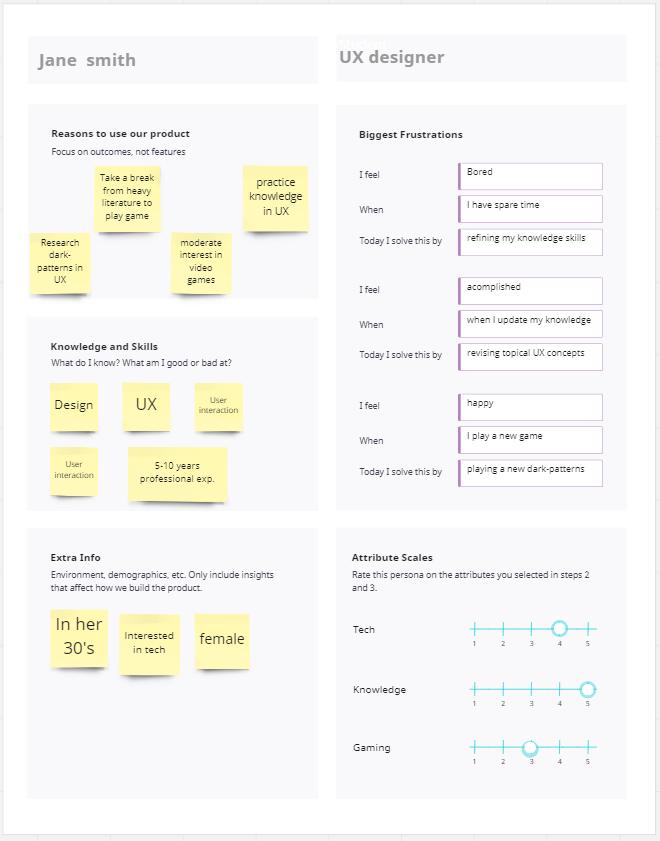
The second persona we have named Timmy Smith. He is a teenager in secondary school. Currently, videos have become topical to talk about in between classes. As Timmy’s interest in gaming grows, he decides he wants to make his own game, so he googles game making for beginners and discovers our beginner game engine. An alternative scenario for Timmy might be, he discovers it through a beginner's computer science class as part of the curriculum or as an extra-curricular module.



(Soberanes, 2018)

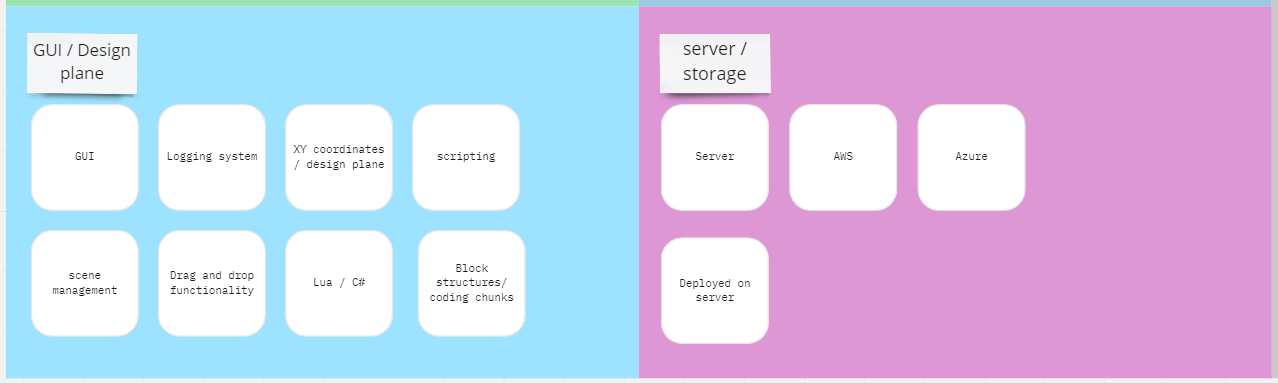
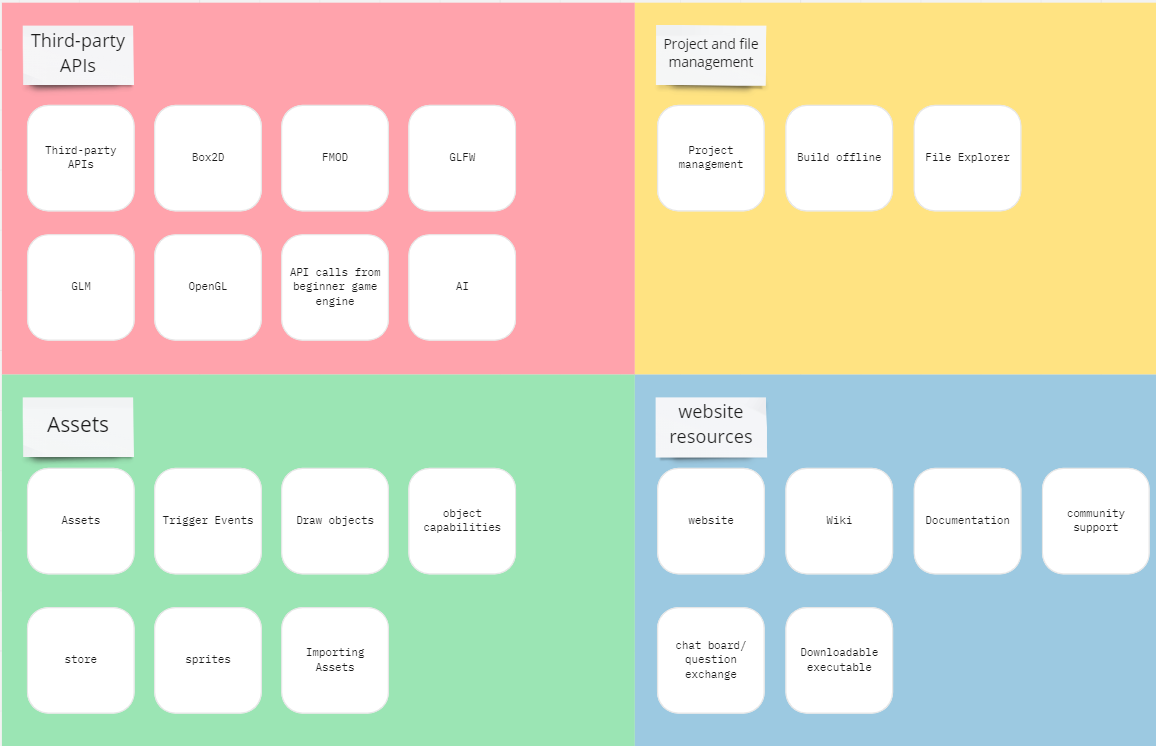


The third persona we can see is Jane smith, a UX designer, with 5-10 years' professional experience working in the user design space. Jane might be looking to upskill herself and decides to do some research into dark-patterns. In doing so, she googles dark-patterns and stumbles across the horizon game engine dark-pattern game. Jane may have been looking at some heavier literature of theory previously and decides to try a lighter method of revision by trying out our game.



Card Sorting

Now that we have established our personas, we can move onto card sorting. Card sorting is the next important step in designing the user interface of the game engine. As Sherwin (2018) describes, card sorting is a UX research method in which study participants group individual labels into criteria that make sense to them. This UX research method can reveal some primitive models relating to the user interfaces being designed. Below we can see a card sorting exercise for the game engine and websites user interface.



As we can see there are 6 broad categories that we can divide our cards into, Third-party APIs, Project and file management, Assets, Website, GUI / design plane and finally, Server/Storage. There are 36 cards allocated into these categories. In looking at these groups we can see that two main areas have emerged, the game engine and features itself and the website where the community resources can be accessed and where the game engine executable file can be downloaded from. Now, that we have identified what the possible groups might look like, we can begin to identify what a user Journey might look like.

# User Journey

The next step is to identify some user journeys. Gibbons (2018) describes a visualisation of the process that person goes through in order to accomplish a goal. A user journey shows some possible routes that one of our personas might take throughout the process of building the game with the game engine or playing the game itself. Below we can see these journeys

## User Journey 1



Above we can see the first user journey of John Smith, he decides he would like to start learning game making but doesn’t know where to start. He decides to google how to make video games and stumbles across a result that looks like it can help him out. This result leads him to the game engine community chat board on a thread about game making. John like the sound of this new game engine and decides to give it a go. He downloads the game engine but needs to provide an email address and password first. The game engine downloads successfully and opens at the entry point. John also leaves open a tutorial to follow from the website.

## User Journey 2





The second user journey follows Timmy Smith who already has the game engine downloaded. The first step is to open the game engine and the game tutorial on the website. Next Timmy opens a new project for the new game and following that identifies a new sprite from the new characters and objects window. Next the new sprite is clicked and dragged into the design plane and a new behaviour is applied to it from the object properties window. Timmy hit the execute button to see if the new sprite and behaviour module works, he also checks the console for any errors that may have occurred or auxiliary information.

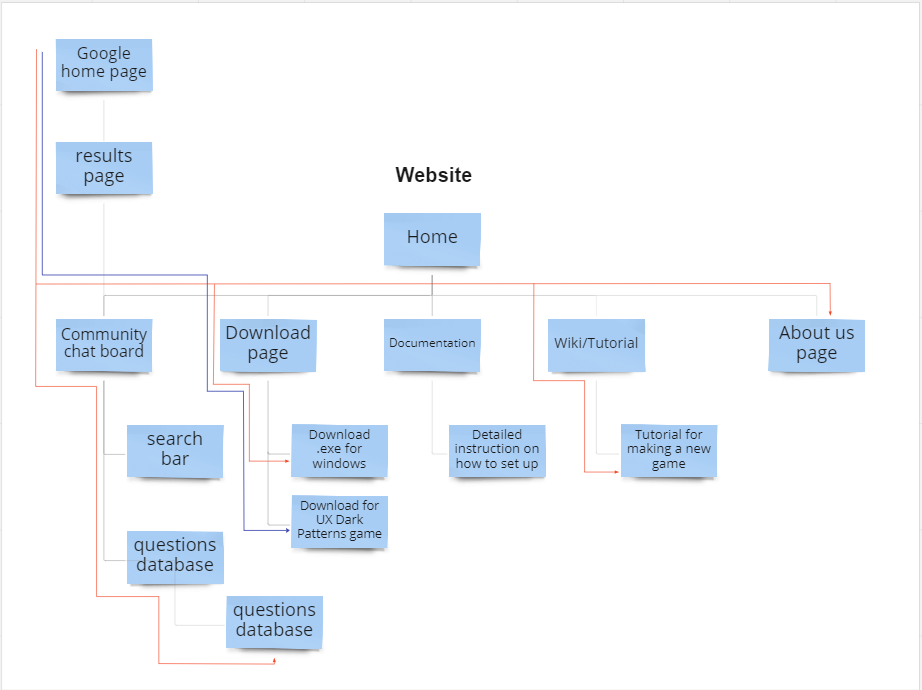


The third user journey we can see depicts the third persona Jane smith, googling UX dark patterns, stumbling across the horizon game engine dark pattern game and deciding to give it a go to brush up on her skills.

Now that we have seen how these user journeys can be implemented, we can move onto how they flow through the site map of the website and game engine editor.

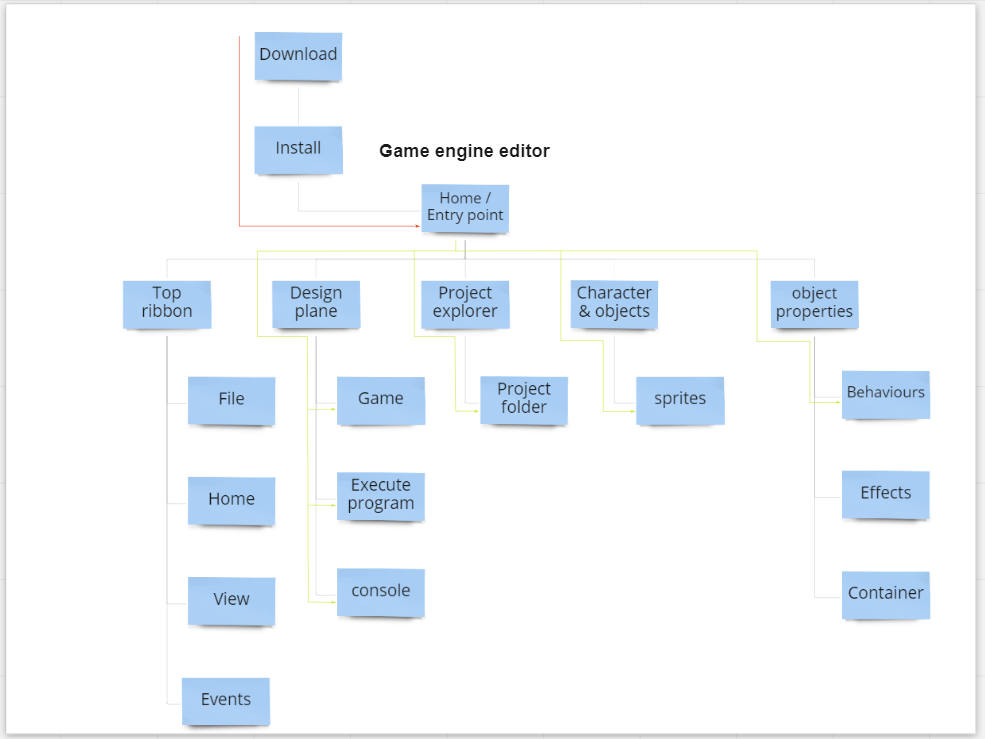
# Site Map

## Website sitemap



Above we can see the sitemap for the website, we can see the user journey of John smith and Jane smith through the orange and purple trail respectively.

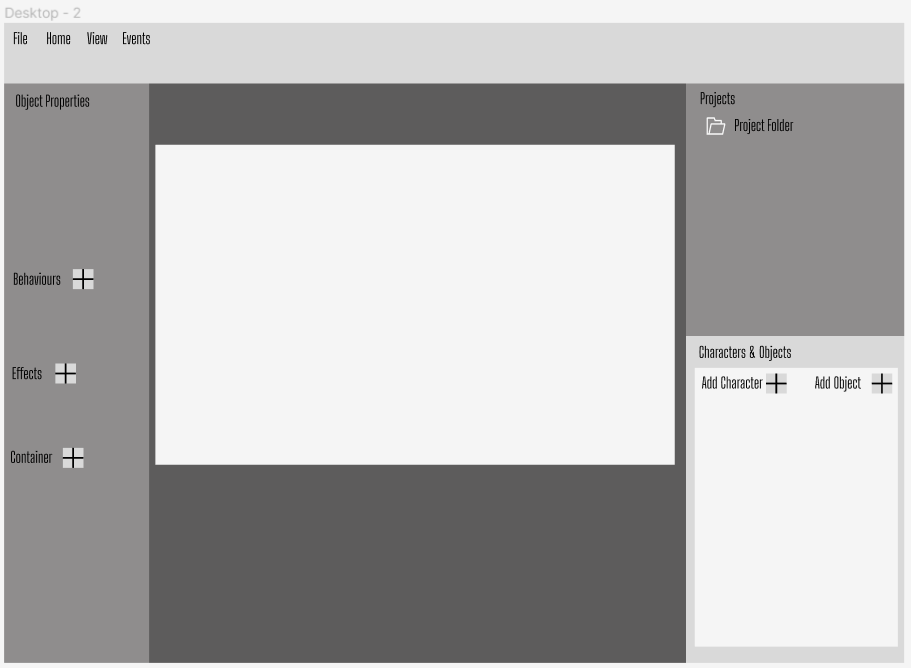
## Game Engine sitemap

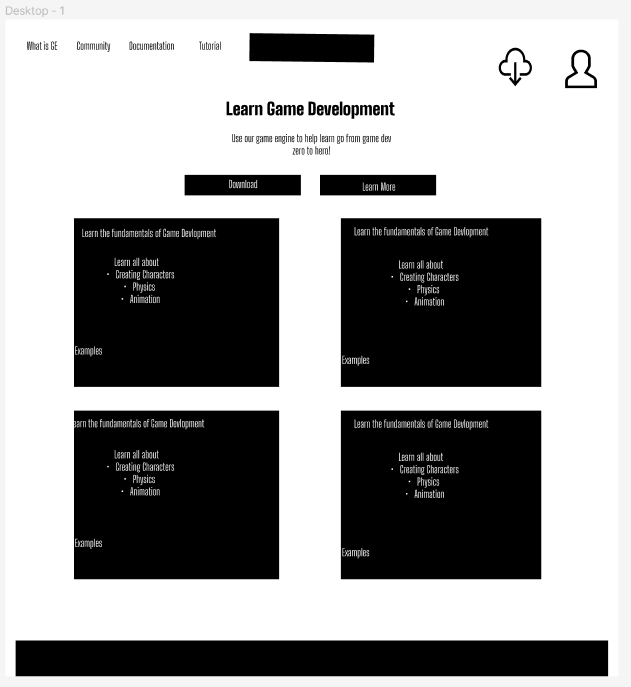


Above we can see the game engine sitemap of the GUI editor, we can see how John smith downloads and installs the game engine finally landing at the entry point of the game engine. Additionally, we can see the process Timmy smith, in bright green, goes through setting up a new project, selecting a new sprite and adding it to the game and applying a new behaviour to it. Finally, executing the program and checking the console for error and information.

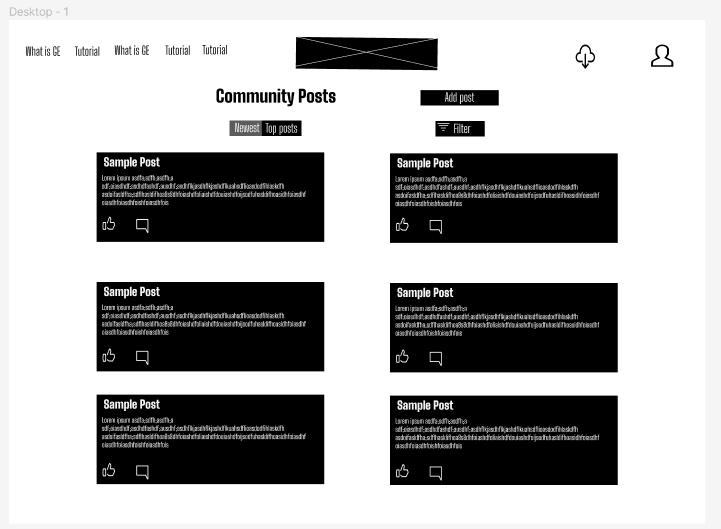
# Wire Frames

We can now start to look at how the game engine graphical user interface, website homepage and website community page might take shape. The purpose of a wireframe is to begin assembling the foundations of the components that will implemented onto our new interfaces. Below we can see the designed wireframes:

This wireframe shows how the components of how the game engine editor might look like. We can see all the major sections described in the sitemap.



The next wireframe is of the website homepage.



The final wireframe is the community posting board.

# User evaluation

1. Evaluation Scope and Perspectives

Website-

1. Did you like the design of website?

2. Did you like the colours of website?

3. Can you navigate to community Dashboard Page easily?

4. Can you ask a question easily from Community Dashboard Page?

5. Can you search related questions on the Dashboard page?

6. Can you find related information from Document page easily?

7. Can you spot the download game engine button easily on the website load?

8. Do you find any challenges while using this website?

Game Engine-

1. Can you create game object easily with using a button ‘Add a game object’

2. Can you add a component without difficulty?

* Can you add a rigidbody component?
* Can you add an audio component?

3. Can you detect flow for adding/changing X and Y positions of game object?

4. Can you see proper navigation or flow for creating any component using our engine?

5. Can you detect the console for logging in to UI design?

6. Did this game engine provide creating game objects/assets for your game genre?

Game-

1. Can you detect entry point to play puzzle related to Roach Model pattern game?

2. Did you like the design and colours of game?

3. Did you like sound effects used for game?

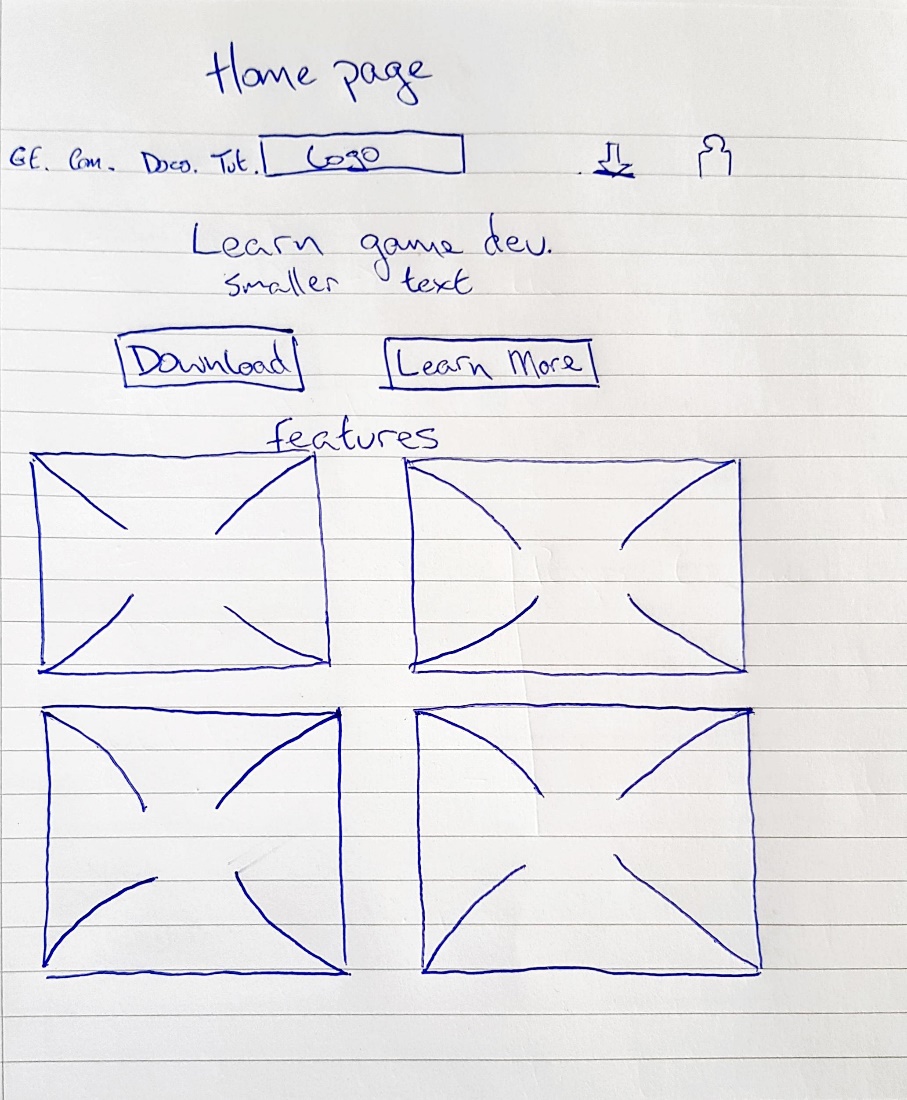
4. Did you understand the concepts of dark patterns taught via game?

5. How long did you play this game?

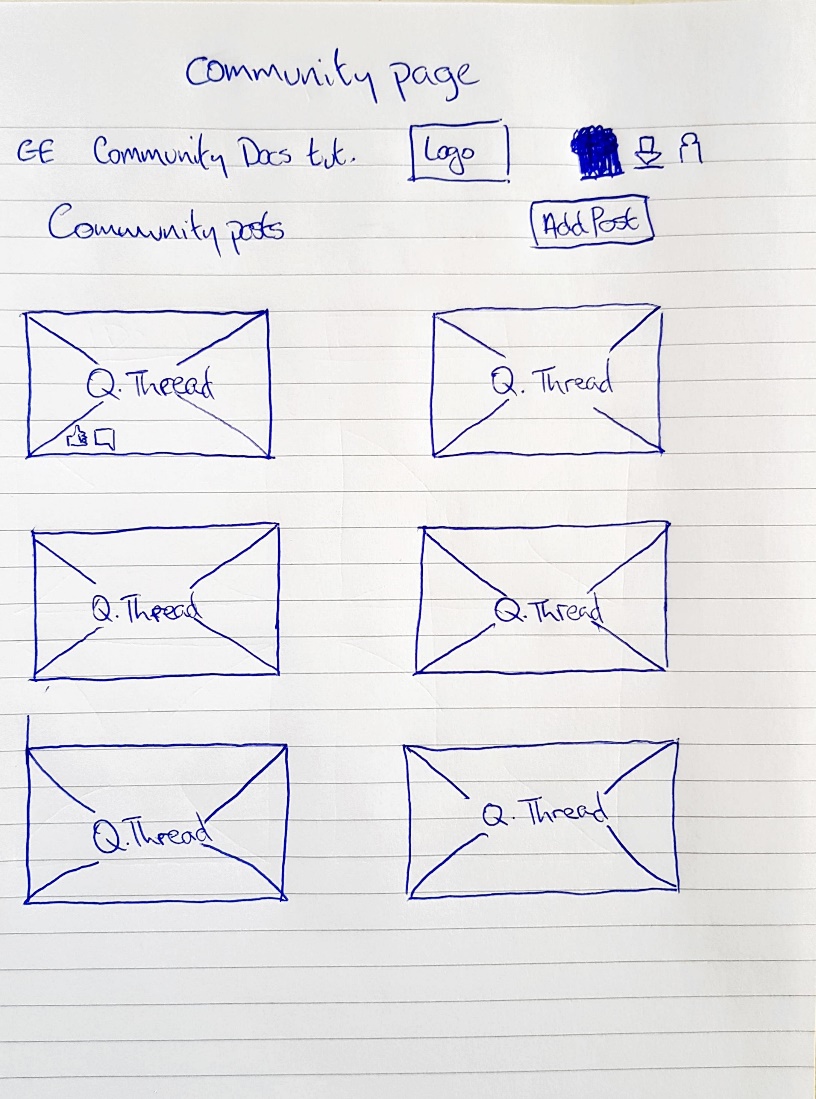
# Low Fidelity

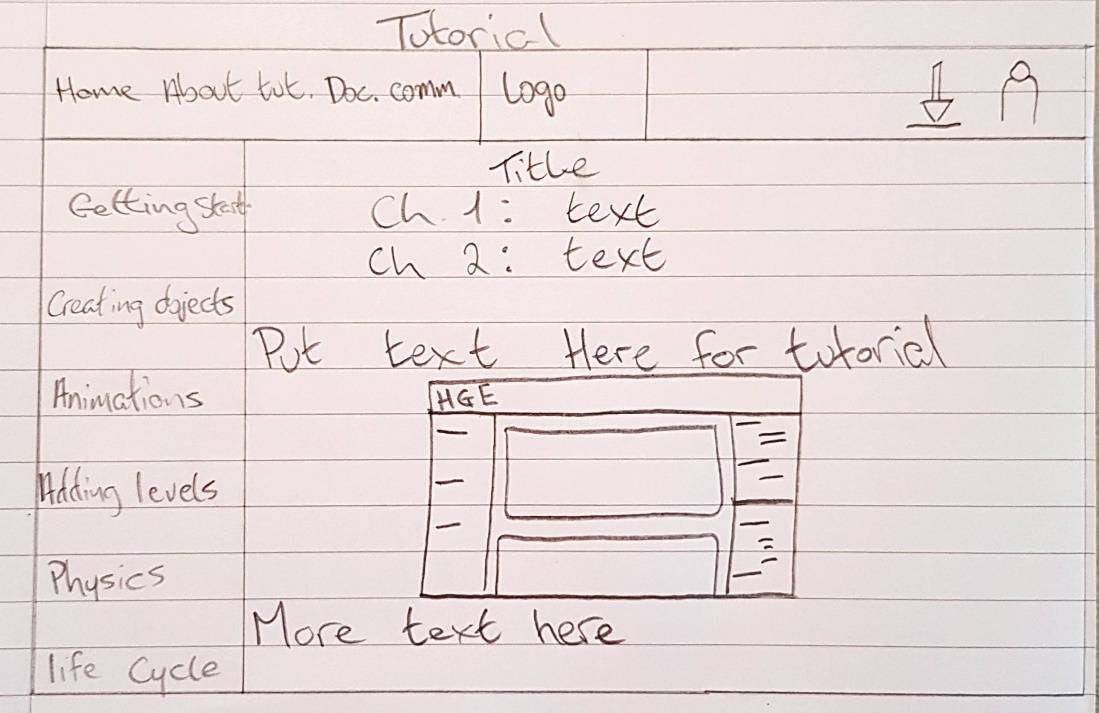
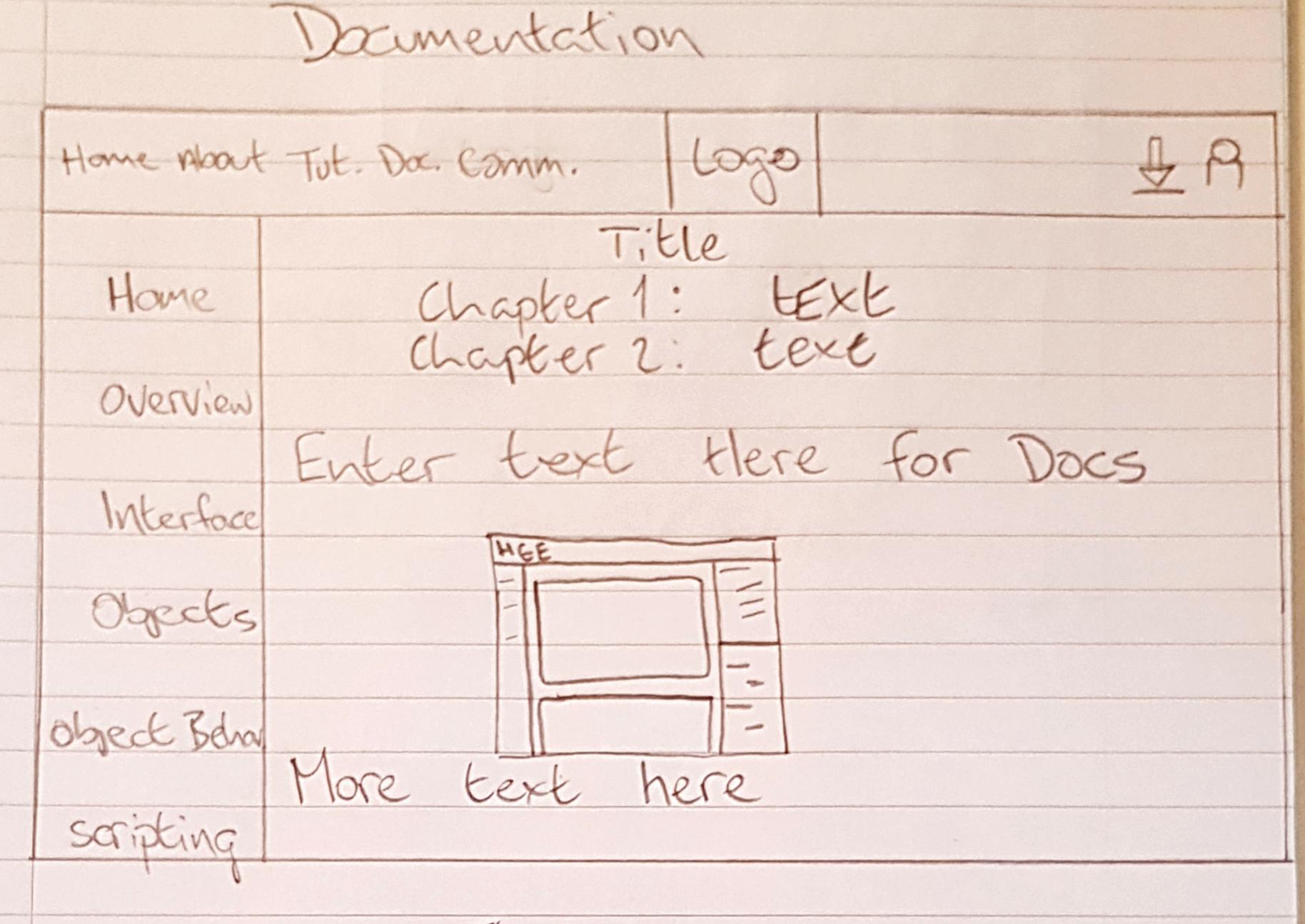
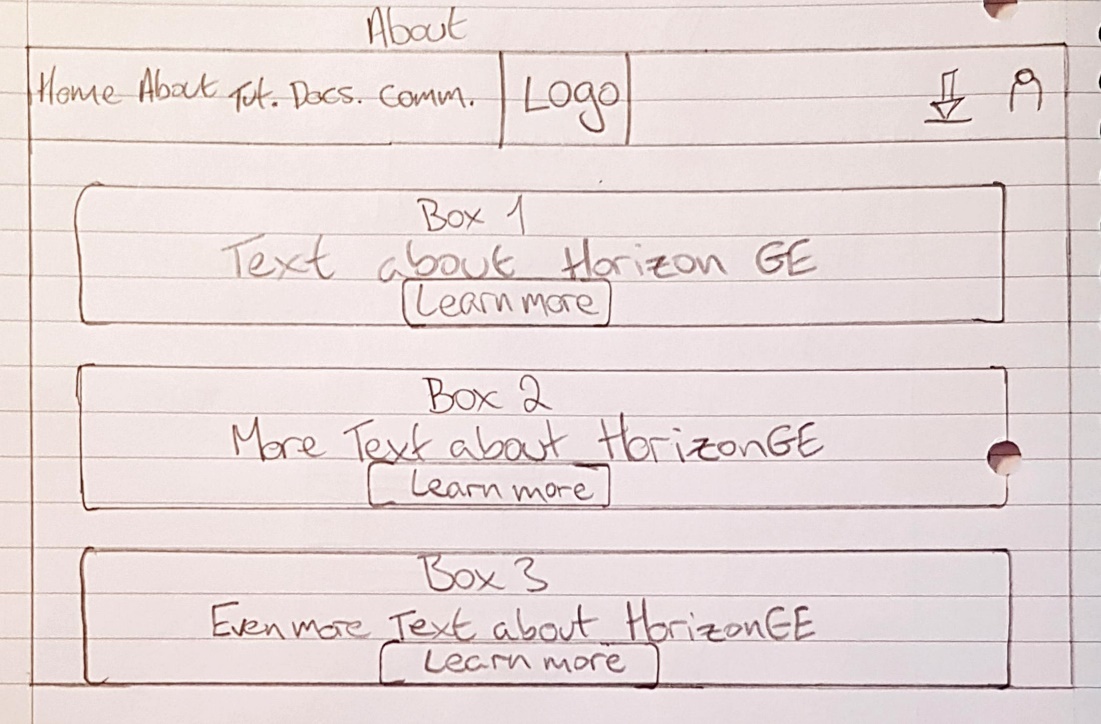
Now it is time to start putting into practise some of the research and data gathered and start sketching what the web pages might look like, we can do this through building a low fidelity prototype. We can also bear in mind Jakob Nielson’s usability principles discussed earlier.

This first low fidelity prototype shows the home page of our proposed website, the page header moves across the top of the pages containing the Logo in the middle, with a clear layout of the pages available to the user on the left-hand side and a download and account icon on the right-hand side. This provides the user with recognition over recall by providing the user with a clear layout of options available. Next, there is a call-to-action component aiming to meet the users eyeline guiding them to directly download the game engine executable and install it. The remainder of the page is used some card components that will showcase some of the features on offer.

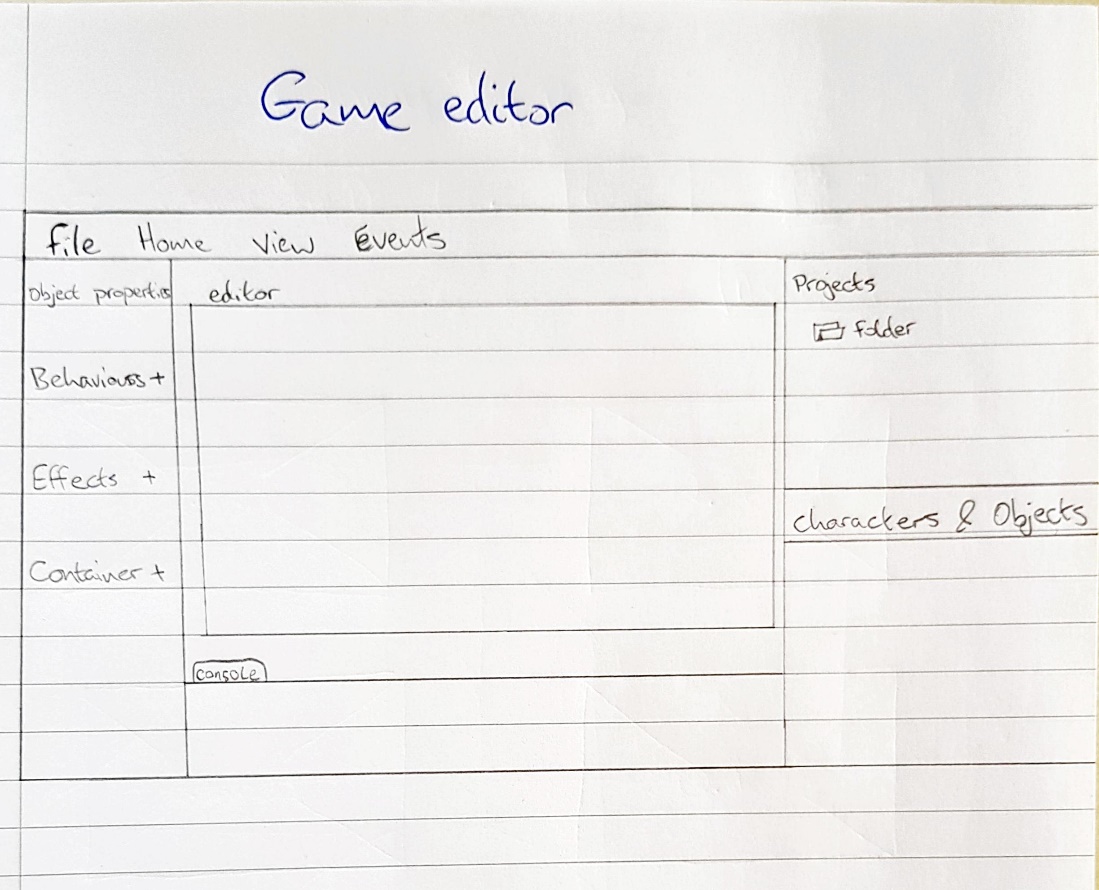


The second low fidelity page show the community page of the website. Here we can see the website header has continued across from the home page. Below this is the community post banner and a further call-to-action for the user to add their own post to the community board containing their question. Finally, we have the community board itself containing all the questions asked by previous users. The “Q. Thread” stands for question thread indicating each question thread.



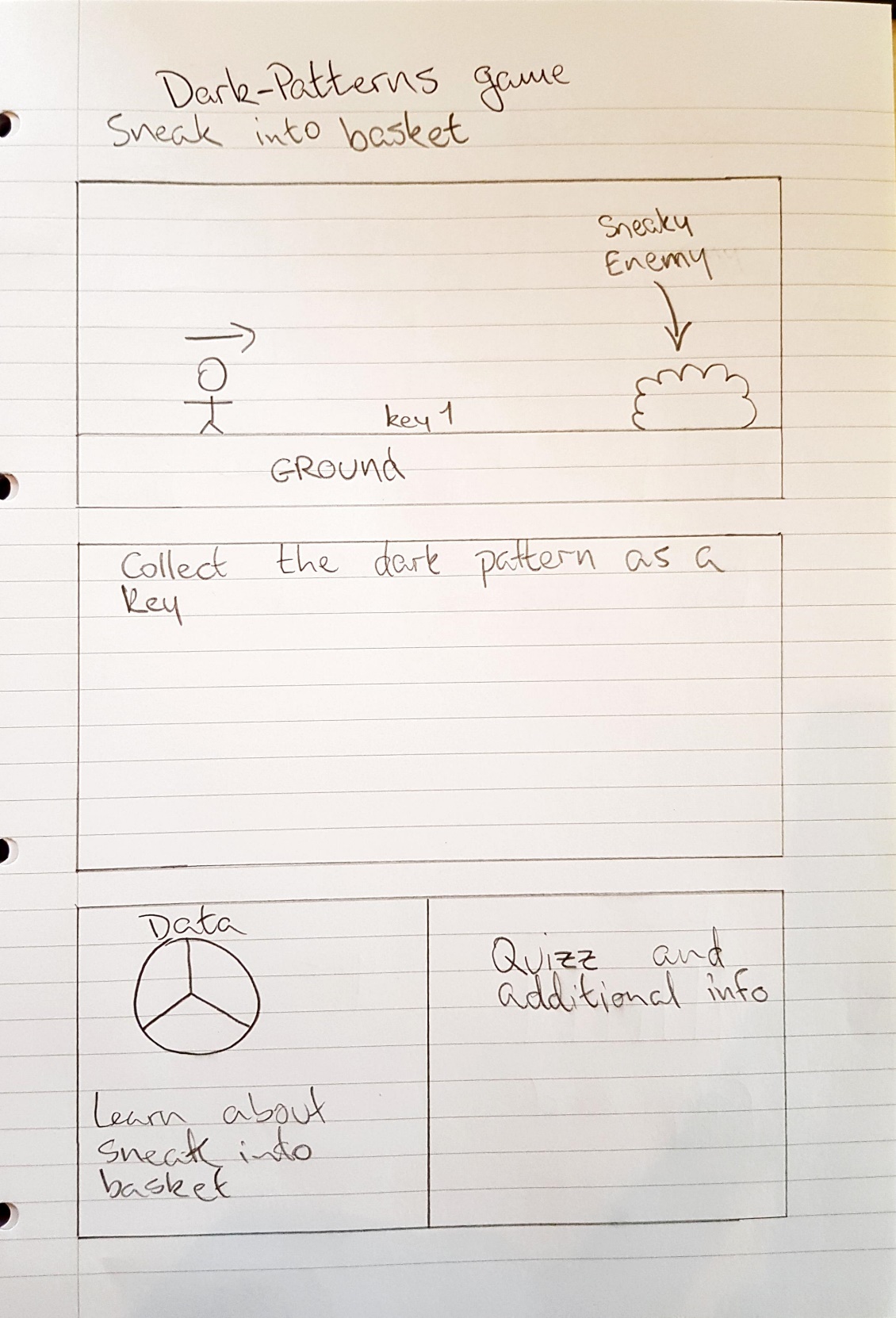


The final low fidelity prototype is the main text editor lobby of the game engine itself. We can see it implements similar block structure to the blocks described in the wireframe. Along the top row we have a ribbon to provide options for the user. To the left of the prototype we an object properties section, this is where the user will be able to apply behaviours and effect onto the characters and objects created. In the centre of the prototype we can see the main editor where the user will build their proposed game, below this is the console area containing output. Finally, to the upper left, is the project explorer where the user can maintain control over all their projects.



These Low fidelities provide a visual aid of our user interfaces and act as a steppingstone towards our final medium fidelity prototypes

## Dark patterns (dp) game

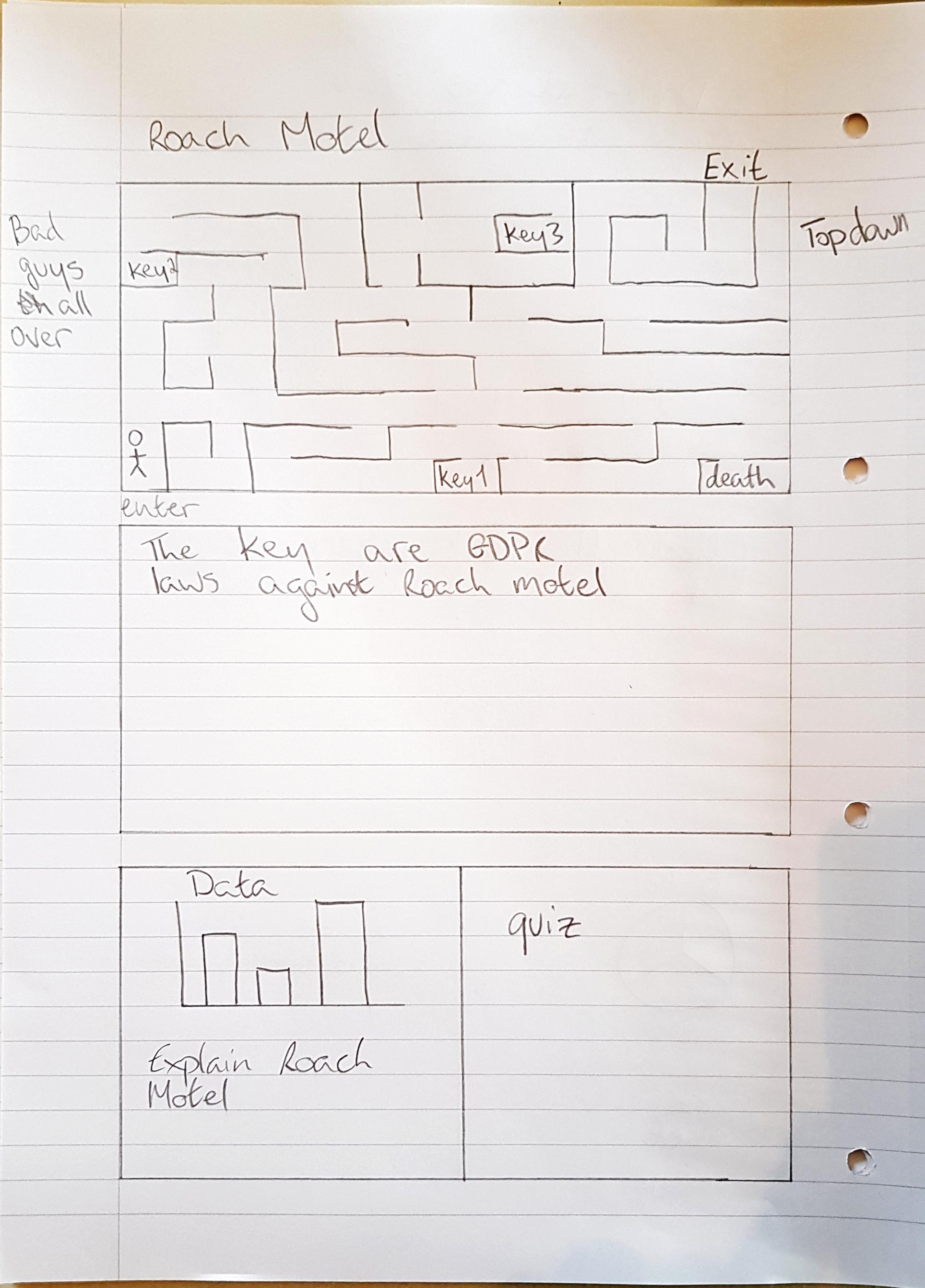


First is a platform style level featuring the sneak into basket dp. I suggest that some of the enemies sneakily hide behind bushes or something like this and pop out to represent sneak into basket. At the end we produce data and information on sneak into basket and a small quizz.

Second is a top-down maze. This can cover roach motel & misdirection dp's. The user will need to collect keys to escape and this struggle to find the exit can represent trying to unsubscribe from a newsletter or some similar roach motel. False exits can represent Misdirection.

Finally, I think we can confirm shame the user into exiting the game, if they exit the game we say you past the confirm shame test and if they say no the game starts again.

If people have further suggestions, I can add them in.

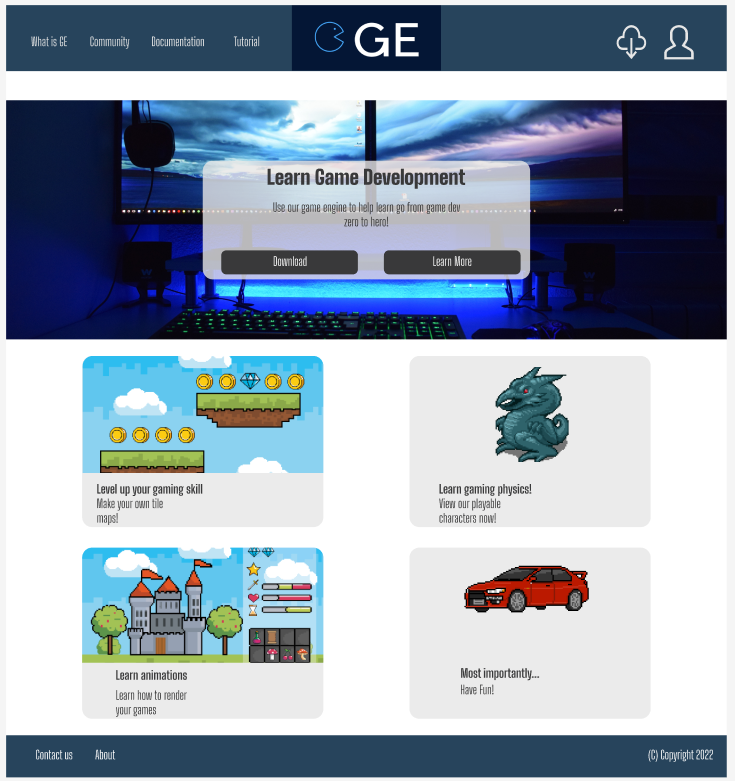


Medium fidelity

Now that we have arrived at the medium prototypes we can provide we can insert a bit more detail and finesse onto our user interfaces. The UI’s maintain similar component structures to the low fidelities.

The first UI is of the website homepage, with regards to the header, the logo has been applied to the middle, the available page tabs have been filled in on the left and the download button and account button remain on the right. Next is the call-to-action on a frosted panel over a desktop background image and finally, the cards showcasing the game engines capabilities and features have been applied. A small footer segment has been added to all website pages.

## Website homepage



Logo - (myfreelogomaker, 2022)

Dual screen PC - (XXSS IS BACK, 2018)

Dragon - (Hyptosis & Zabin, 2012)

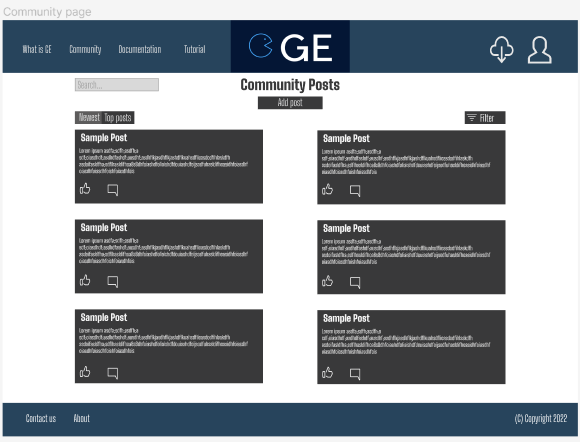
Arcade world (coins) - (stockgiu, 2019)

Arcade world(castle) - (stockgiu, 2019b)

Car - (madmedicsoft, 2022)

## Website community page

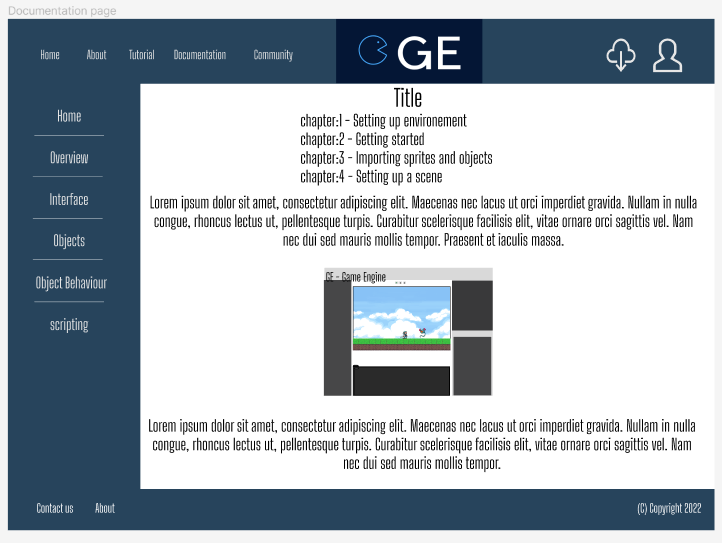
The next medium fidelity is the is the community chat board, this is an important prototype as it is considered one of the social good aspects of the group project. We can see the same page header applied; it will be used across the board. Next, is a segment that provides functionality for the user. As suggested within the UX guidelines a search bar has been provided, a “community posts” banner and “Add post” option has been added, along with some filtering functionality. Finally, the collection community posts have been displayed in a symmetrical fashion. Each post contains a header, content, like button and reply button.



Logo - (myfreelogomaker, 2022)

## Documentation page

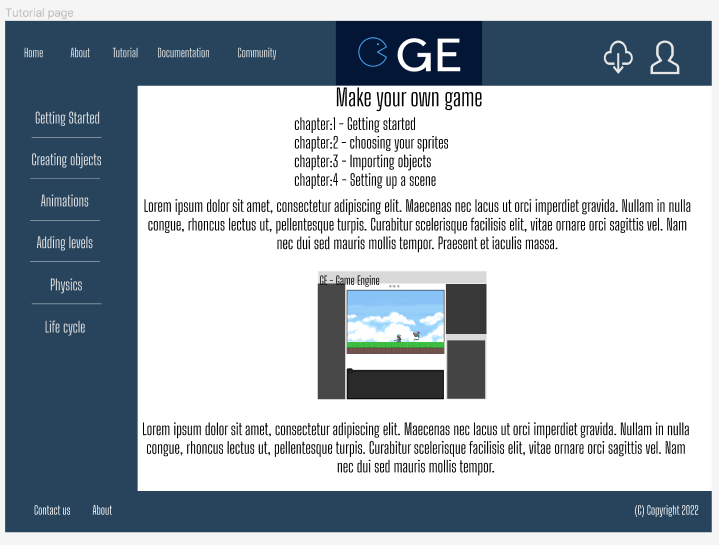
Below we can see the medium prototype for the documentation page of the HorizonGE website. This page will include the standard header as described in the above, a dynamic sub menu bar to the side of the prototype depicting the content available for reading, and further information on the game engine below. The main content to be read through will be displayed in the format – title, chapter and content supplemented with screenshots.



Logo - (myfreelogomaker, 2022)

## Tutorial page

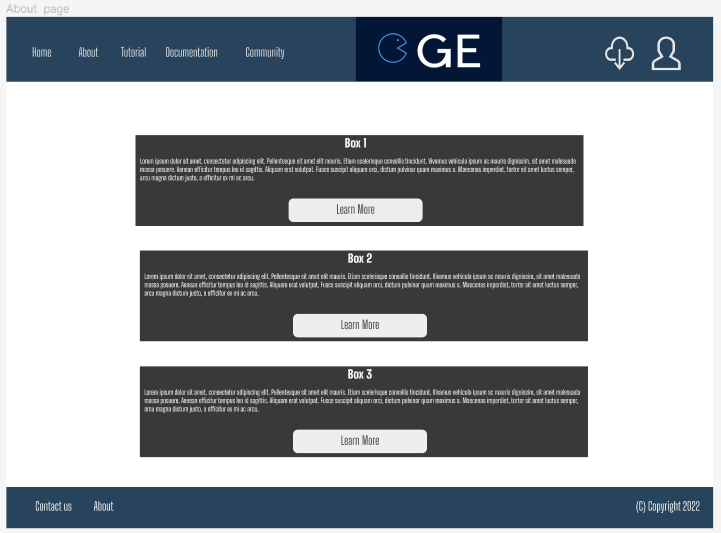
The tutorial page will follow a similar format to the documentation page, however, contain content pertaining to building a game using the game engine as opposed to the usage of the game engine features. We can see below that these pages both look quite similar.



Logo - (myfreelogomaker, 2022)

## About page

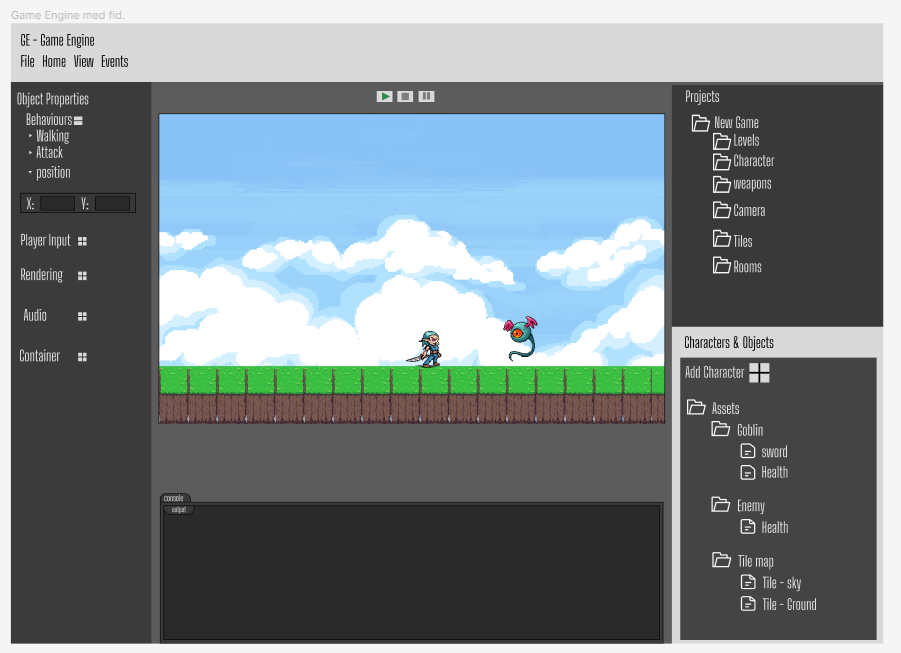
The about page will a simple website page that allows the user to find out more information about the HorizonGE game engine. It uses the same standard header and footer from across the website.



Logo - (myfreelogomaker, 2022)

## Game engine editor

The final medium fidelity prototype is the Game engine main editor user interface. We can see the main GE title with the user options toolbar underneath. The left-hand side panel provides and shows the implemented and available object properties that can be applied to a character or object in a hierarchical view. The main editor shows the scene view area containing each component element of the user's game, we can see how the sprite character, enemy and Tile map in this area. Below the main editor is the console to contain output. To the right of the image, we can see the project view where the user can manage all their game folders. Finally, in the bottom right of the image is the characters and object section where the user will be able to select their character sprites and tiles.



Goblin warrior & cyclopse monster – (Hyptosis & Zabin, 2012)

Sky - (Riva, 2016)

Ground - (ArMM1998, 2017)

**References**

References

aolko. (2022). *aolko/construct: Scirra Construct Classic*. GitHub. <https://github.com/aolko/construct>

ArMM1998. (2017, February 16). *Zelda-like tilesets and sprites*. OpenGameArt.org. <https://opengameart.org/content/zelda-like-tilesets-and-sprites>

Babich, N. (2017, May 2). *Best Practices for Search | UX Booth*. Uxbooth.com. <https://www.uxbooth.com/articles/best-practices-for-search/>

Carnes, B. (2022, August 25). *Create Games with LÖVE 2D and Lua*. FreeCodeCamp.org; freeCodeCamp.org. <https://www.freecodecamp.org/news/create-games-with-love-2d-and-lua/>

Castanié, P. (2018, November 10). *man in blue denim top inside room*. Unsplash.com; Unsplash. <https://unsplash.com/photos/fG_ud73U_ZM>

Chover, M., Marín, C., Rebollo, C., & Remolar, I. (2020). A game engine designed to simplify 2D video game development. *Multimedia Tools and Applications*, *79*(17). <https://doi.org/10.1007/s11042-019-08433-z>

coronalabs. (2022, August 30). *coronalabs/corona: Solar2D Game Engine main repository (ex Corona SDK)*. GitHub. <https://github.com/coronalabs/corona>

Doucet, L., & Pecorella, A. (2021, September 2). *Game engines on Steam: The definitive breakdown*. Game Developer. <https://www.gamedeveloper.com/business/game-engines-on-steam-the-definitive-breakdown>

Fitzgerald, A. (2022, July 27). *What Is the Law of Proximity & Why Does it Matter in UX Design?* Hubspot.com; HubSpot. <https://blog.hubspot.com/website/law-of-proximity>

Fogg, B. (2022, July 26). *Behavior Model*. Behaviormodel. <https://behaviormodel.org/>

Game Designing. (2022, August 25). *A Comprehensive Guide to GameMaker Language (2022)*. Video Game Design and Development. <https://www.gamedesigning.org/engines/gml/#:~:text=is%20king%20here.-,What%20Programming%20Language%20Does%20GameMaker%20Studio%20Use%3F,the%20commands%20you%20typed%20in>.

GameMaker. (2013). *Make A Game With GameMaker | Free Game Engine*. GameMaker. <https://gamemaker.io/en/gamemaker>

Gibbons, S. (2018, December 9). *Journey Mapping 101*. Nielsen Norman Group. <https://www.nngroup.com/articles/journey-mapping-101/>

Godot docs. (2022a). *Differences between GLES2 and GLES3*. Godot Engine Documentation. <https://docs.godotengine.org/en/stable/tutorials/rendering/gles2_gles3_differences.html>

Godot docs. (2022b). *Frequently asked questions*. Godot Engine Documentation. <https://docs.godotengine.org/en/stable/about/faq.html>

Godot Engine. (2021). *About Godot 4, Vulkan, GLES3 and GLES2*. Godot Engine. <https://godotengine.org/article/about-godot4-vulkan-gles3-and-gles2>

godotengine. (2022, September 9). *godotengine/godot: Godot Engine – Multi-platform 2D and 3D game engine*. GitHub. <https://github.com/godotengine/godot>

Gregory, J. (2019). *Game engine architecture*. Crc Press, Taylor & Francis Group.

Hyptosis, & Zabin. (2012, December 2). *Lots of free 2d tiles and sprites by Hyptosis*. OpenGameArt.org. <https://opengameart.org/content/lots-of-free-2d-tiles-and-sprites-by-hyptosis>

Lin, S. (2017, January 31). *The Rules for Modern Navigation | UX Booth*. Uxbooth.com. <https://www.uxbooth.com/articles/the-rules-for-modern-navigation/>

Linietsky, J. (2017). *Godot 3’s renderer design explained*. Godot Engine. <https://godotengine.org/article/godot-3-renderer-design-explained>

*LÖVE - Free 2D Game Engine*. (2022). Love2d.org. <https://love2d.org/>

love2d. (2022, August 26). *love2d/love: LÖVE is an awesome 2D game framework for Lua.* GitHub. <https://github.com/love2d/love>

madmedicsoft. (2022, September 20). *Pixel art cars from “Gop go go” project (russian, soviet an some inomarka)*. OpenGameArt.org. <https://opengameart.org/content/pixel-art-cars-from-gop-go-go-project-russian-soviet-an-some-inomarka>

Mads Soegaard. (2020, July 26). *Hick’s Law: Making the choice easier for users*. The Interaction Design Foundation; UX courses. <https://www.interaction-design.org/literature/article/hick-s-law-making-the-choice-easier-for-users>

Messaki, E. (2020, December 18). *Call-to-Action. The Foundations of Efficiency - Outcrowd - Medium*. Medium; Outcrowd. <https://medium.com/outcrowd/call-to-action-the-foundations-of-efficiency-f8218fa8e344>

myfreelogomaker. (2022). *My Free Logo Maker - Get a Free Logo Design and Download*. Myfreelogomaker.com; My Free Logo Maker. <https://myfreelogomaker.com/brandkit/97131777/logo-files>

Nielsen, J. (2000, July 22). *End of Web Design*. Nielsen Norman Group. <https://www.nngroup.com/articles/end-of-web-design/#:~:text=Jakob’s%20Law%20of%20the%20Internet%20User%20Experience&text=This%20means%20that%20users%20prefer,other%20sites%20they%20already%20know>.

Nielsen, J. (2016, July 9). *Tabs, Used Right*. Nielsen Norman Group. <https://www.nngroup.com/articles/tabs-used-right/>

Nielsen, J. (2020). *10 Usability Heuristics for User Interface Design*. Nielsen Norman Group. <https://www.nngroup.com/articles/ten-usability-heuristics/>

Ogunleke, S. (2022, January 25). *Why Is Lua Popular for Game Development and How Do You Use It?* MUO; MUO. <https://www.makeuseof.com/why-is-lua-popular-for-game-development-how-to-use/>

Riva, P. (2016, January 13). *Sky background*. OpenGameArt.org. <https://opengameart.org/content/sky-background>

Scratch. (2022). *Scratch - Imagine, Program, Share*. Mit.edu. <https://scratch.mit.edu/>

*Scratch*. (2022, October 11). GitHub. <https://github.com/LLK/>

Sherwin, K. (2018). *Card Sorting: Uncover Users’ Mental Models for Better Information Architecture*. Nielsen Norman Group. <https://www.nngroup.com/articles/card-sorting-definition/>

Soberanes, U. (2018, August 25). *man wearing Sony PlayStation VR*. Unsplash.com; Unsplash. <https://unsplash.com/photos/MxVkWPiJALs>

stackoverflow. (2011, February 19). *What is a good game engine that uses Lua?* Stack Overflow. <https://stackoverflow.com/questions/5053134/what-is-a-good-game-engine-that-uses-lua>

stockgiu. (2019a, June 20). *Arcade game world and pixel scene Free Vector*. Freepik. <https://www.freepik.com/free-vector/arcade-game-world-pixel-scene_4814952.htm#query=2d%20art&position=0&from_view=keyword>

stockgiu. (2019b, June 20). *Arcade game world and pixel scene Free Vector*. Freepik. <https://www.freepik.com/free-vector/arcade-game-world-pixel-scene_4814938.htm#query=2d%20art%20castle&position=3&from_view=search>

Unity Technologies. (2017). *OTOY OctaneRender*. Unity. <https://unity.com/products/otoy-octanerender>

Unity Technologies. (2022). *Unity*. Unity. <https://unity.com/>

Virdeee. (2022, September 20). *Main Heroes*. OpenGameArt.org. <https://opengameart.org/content/main-heroes>

Wikipedia Contributors. (2021, October 3). *Construct*. Wikipedia; Wikimedia Foundation. <https://en.wikipedia.org/wiki/Construct>

Wikipedia Contributors. (2022a, June 2). *Solar2D*. Wikipedia; Wikimedia Foundation. <https://en.wikipedia.org/wiki/Solar2D>

Wikipedia Contributors. (2022b, August 28). *OpenGL ES*. Wikipedia; Wikimedia Foundation. <https://en.wikipedia.org/wiki/OpenGL_ES>

Wikipedia Contributors. (2022c, September 16). *Scratch (programming language)*. Wikipedia; Wikimedia Foundation. <https://en.wikipedia.org/wiki/Scratch_(programming_language>)

Wikipedia Contributors. (2022d, October 4). *Godot (game engine)*. Wikipedia; Wikimedia Foundation. <https://en.wikipedia.org/wiki/Godot_(game_engine>)

Wikipedia Contributors. (2022e, October 8). *GameMaker*. Wikipedia; Wikimedia Foundation. <https://en.wikipedia.org/wiki/GameMaker#:~:text=The%20engine%20uses%20Direct3D%20on,and%20proprietary%20APIs%20on%20consoles>.

XXSS IS BACK. (2018, January 6). *Two Computer Flat Screen Monitors Turned On · Free Stock Photo*. Pexels.com; Pexels. <https://www.pexels.com/photo/two-computer-flat-screen-monitors-turned-on-777001/>

Yuriy Denisyuk. (2022, July 5). *Godot vs Unity 2022: Which Game Engine is Best for You?* Pinglestudio.com; Pinglestudio. <https://pinglestudio.com/blog/full-cycle-development/godot-vs-unity-2022#:~:text=speak%20for%20themselves.-,Godot%20vs%20Unity%3A%20for%20beginners,offers%20a%20smoother%20entry%20point>.