‘Save the Jungle’ - Tower Defense Game

- ASP Group 8, Mar 2022

Members:

Alex Chu
Dimitri Vlachos
Jeremy Matthews
Freda Xiaoyun Yu
(Absent: Sharif Khan)

Link for Our Game:

https://aspgrp8.z1.web.core.windows.net/

Github:

In order not to introducing confusion for our developers when cloning and pushing repositories, we have created two Github routes, in different members' pages.

- Github route for **codes**: https://github.com/matthewsja/asp8_TowerDefense
- Github route for all the **documents**: https://github.com/FredaXYu/ASP_Group8

User Guide: See here.

Contents

Background

Virus, fears, insulation. In 2022, the world is not yesterday. Nature has taken revenge back.

How we wish to turn the clock backwards, to the days when humans had not done such bad things to Nature! How we wish that the Nature is just like hundreds of years ago, when there were no greenhouse effects, loss of rainforests, and rise of sea levels. It's time to reflect what we humans have done to the Natural environment. We need to apologize to Nature, apologize to the reason that has made the COVID-19 spread.

What if WE are the animals, and OUR ENEMY is human being?

Previously we had the idea to make an **educational Tower Defense Game**. Since Miller in his book mentions the importance of 'a good story' (Miller, preface), now we wish to implement such a story into our game. We set our game setting to be **in the jungle or forest, where animals (and plants) build towers to defense against human beings' invasion to the jungle**.

Reasons we choose this story:

1. It is educational. Whoever plays the game, he or she would be largely educated by the environment-protection idea.

2. It is creative. The **novelty** for our main idea is that we think the partition settlement – animals as our side, and human beings as enemy – is innovative. Seldom has anyone thought of **standing in the shoes of others**, seldom has anyone considered he or she is the animal, and humans the group is another thing. Once we have changed our perspective to play the game, to defense, to know that the safeguard is tough, then we can realise that we humans have done lots of devastating behaviours to the forest, and perhaps we may stop these bad behaviours just from now on.

3. Characters are from real world. (1) It manifests Nielson's second usability principle 'match between system and the real world'. (2) It's easy to draw characters, since we don't need to create any new characters out of thin air.

4. It has ecological meaning. (1) We need to apply ecology concepts to set which animal as tower and which human behaviour as enemy. Perhaps in the future we will design more targeted animal-against-enemy pairs based on the animal's habits. (2) Reversely, once we have built our software with a large library of towers and enemies, ecologists might use our project to educate people, or might even learn from our settings. They perhaps will ponder about the old theories from an entirely new perspective – standing from the animals' side to see humans, or they may find some reasons for some certain animals to survive in the world.

To state why initially we choose to build an educaitonal Tower Defense game, here are the reasons:

(1) The pandemic has introduced a wide opportunity for remote working. We can work remotely in a team for a big project.

(2) We all are interested in making a game, since it is relaxing enough to make the process enjoyable.

(3) The pandemic era is also an 'isolation' era for everyone. It's time for the market of video games appeared on the Internet since it's a new world (though virtual) for people to get rid of isolation in real life. People can find friends through games. If we can utilise this feature, we may make our game spread by users' sharings. We may gain a lot of money.

(4) In this pandemic era, it's hard for people to reach education resources. Thus, an educational product is

a good meal for them to absorb the meanings they should understand outside of the school.

(5) We guess schools and universities are also using online tools to teach right now. So we may introduce our product to universities. Once they've accepted, our product would have a reliable platform. Till then, profit is not important, since we have found our meanings.

(6) We sincerely hope that our product would bring happiness, meanings, reflections, and some changes to the people who use it, and to the whole world we live in. People love games. If we can utilise this feature to make them realise even if a little knowledge or a little reflection, it is worthwhile for us to do the whole thing. The world doesn't need wars, and yet it is a war just through each plastic product we buy. It needs sober individuals instead of sleepwalkers who are addicted to games. Still we are making games, just because we want to wake people up.

Aims & Objectives

(Adjusted parts from midterm are shown in green.)

- Scope: On the whole, To build a **Tower Defense (TD) Game** which operates on computers **with an educational theme of environmental protection, by settling animals as towers and human bad behaviours as enemies. (Newly elaborated)**
- **This game should be educational so that users will understand: (1) humans' bad behaviours are enemies of the living creatures in Nature; (2) animals need to take much effort to protect their home; (3) there can exist a happy ending where humans and animals live in harmony; (4) although animals have done all they could to defense, the most powerful weapon is still the law settled by humans ourselves. (Newly elaborated)**
- To discover the process of making a video game when doing this **game project**.
- To involve all **team** members into this big task. To maximize the productivity of the team by using each **individual**'s advantages. Team members can learn from each other to improve their shortages. Task: draw a Gantt chart, ddl Nov. 26; **a new Gantt chart for the second half of work, ddl Jan 20**.
- For each team member, learn how to manage **time** and deadlines for each sub-task.
- Learn how to **cooperate** a task. Tasks: (1) settle down the platforms, ddl Nov. 30. (2) **handle members' leave requests and arrange works fluently (previously said 'meeting arrangement' can be canceled since our communication through Slack is clear enough);** (3) understand the full development process and important milestone tasks, ddl Dec. 31.
- Learn how to do **version-control**s for a project using git operations or platforms. Task: use Github, ddl middle Dec.
- Experience the process to do simiple **statistics**. Task: (1) make questionnaires, ddl Nov. 30. (2) spread the questionnaires, ddl Dec. 15. (3) data analysis, ddl Jan. 10.
- To fully understand the **users**' psychology, then make a project which benefits the users. Tasks: (1) extract user requirements from the data, ddl Dec. 31. (2) continously gather user opinions, ddl March 2022.
- To understand the basic elements and how they interact. Tasks: (1) write a **requirement specification**, ddl. Dec. 31. (2) draw **UML,** ddl. Dec. 31.
- To realise the requirement specification into **codes**. Tasks: (1) initial codes, ddl Dec. 31. (2) full codes, ddl

March 2022.

- To do software testings, ddl March 2022, including user testing, usability testing, accessibility testing, etc.
- **Understand the importance of telling a good main story through book reading / user preferences. (Newly added)**
- **To experience how an agile project works in team, to understand what should we do to get users involved, to experience the process of responding quickly, to explore how to take all the changing records timely and tidily. (Newly added)**

Stakeholders:

Not everyone can appreciate the meanings of our product. Thus, we should target precisely, so that our product and our energy won't be wasted.

Since it's educational, our target groups are **students**. Since it's a game, our target groups are **young people** who are energized. Also, it could be utilised by any **environment-protection groups or famous Nature exploration organisations** to spread their ideas (or to advertise their groups). It could also be widely used in **universities** as a case in class, so we should consider **teachers** within. (Once we have communications with any **game platform**, it will also be our first-place key stakeholder.)

Key stakeholders	Age	Job	Importance
Students	6-30	Primary/secondary/university student	Main
Teachers/educators	40-60	Lecturer/organiser/officials in university	Main
Organisations	-	Non-profit/profit environmental protection group	Expectation
Game companies/platforms	-	Leading platforms in game industry	Expectation
Players for the above platforms	Depends	Game users	Expectation

In the process of questionnaire gathering, we have settled 'age' question and 'job' question in order to identify whether the person is key stakeholders that we should focus on.

According to one interview to an experienced customer manager done by Xiaoyun: (1) People tend to give us some information through phone calls, but not through questionnaires, especially when there are open questions that need them to write down. So we should construct as many **multiple-choice questions** as possible in our questionnaire. (2) For key stakeholders, we should take priority for them, like making phone calls periodically, and visiting their companies in person, etc.

Planning & Research

Research & novelty

Game collections:

We have searched all through the Internet about similar environment-protection or animal games.

(1) Widely known **Pokemon** is similar to our settings which includes many game types. However, its idea is always to separate Pokemons (animals) into different teams and fight each other, or fight against the evil organisation Team Rocket. It's not like our idea to let humans and animals to fight. So we have novelty. In our thoughts, Pokemon provides little meaning other than to realise the surviving rule in Nature.

Callahan et al. also investigated Pokemon, and finally have created a new board game '**Phylo**'. The new game aims at biodiversity conservation, but surely Pokemon itself is rarely educational. We think our work converges with Callahan's, since we both aims at a ecological game with animals as characters. However, our games advances in **human behaviours** involved.

(2) Study of ecological games:

Game Name	Type	Features	Comments
Eco	Open World Survival	Learn through play, w/t documents; 3D with vivid scene	
Animalia Survival	Simulation; Survival	Play as the animal to survive	Similar view point – stand on the side of animals.
A New Beginning - Final Cut	Adventure	Save the world from the climate cataclysm	Save the world actually don't need big actions. Just need awareness.
Botanicula	Casual; Exploration	Whole story happens on one tree.	Learn from the funny parts: sounds, design, story.
Flood protection game - Tsai	Casual; educational	Cartoon; simple drawing	
Liver defense biological game - Brich	Educational; cartoon; casual	Consistent style; few colour choices; simple shapes; meaningful panels; implemented clinical data	A game need to be straightforward.

Paper collections:

Research before midterm is here:
https://github.com/FredaXYu/ASP_Group8/blob/main/before_midterm/bg_research/Research_Results/before_midterm_research.doc

We have searched in many academic websites after midterm.

IEEE only includes limited amount of articles about tower defense game, and **very few about educational tower defense game**. That means, our project is rarely done by computer science area peers.

However in Nature website, we have found much evidences that puch us forward in making **educational games**:

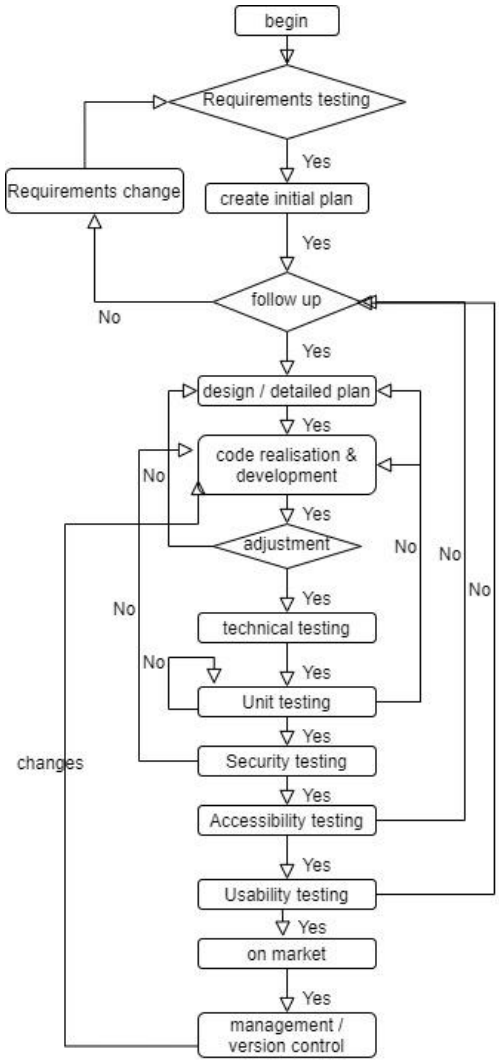
(1) In this isolation era, people are tend to catch mental illnesses. According to Peters et al., playing video games has a significant **efficacy for dyslexia** (Peters 'Dyslexia' paper). We deem that our game could also help users not only to cure them from already existed mental illness, but can also prevent them from getting down, by immersing themselves in a co-related world that we build. Just as what Krotoski says, there are evidence showing that students playing games often are better at learning than those who don't (Krotoski, pp. 695).

(2) Games are a creative method in education. 'games serve as very fundamental and powerful learning tools.' (Koster, pp.35). Krotoski has also stated in his finding that, games is a very valuable way for kids to learn, and surprisingly, **players may have learned** things '**even when** the educational material underlying the game **has flaws**', since the game itself has a wide range of market, and it will encourage its audience to debate about the scientific theory (Krotoski, pp. 695). Therefore, we don't need to be perfect but still our educational aim would be achieved.

(3) Boulton et al. have found that **Amazon jungle has reached a breakthrough point** where Amazon will transform from rainforest to tropical savannah, due to human activities such as tree cutting down. Here we have another support evidence for us to design the enemy 'saw'. We hope our game would teach people to protect rainforests to **save Amazon** and others.

(4) Koster has said that, 'The only real difference between games and reality is that the stakes are lower with games.' (Koster, pp.34) His point is that through games, players can experience a new world that is similar to the real world. This is inspiring. **Experiences are deep** in mind, and, games are immersing, so they are just fit for conveying ideas, if we can create a place. We hope through our game, users **will experience in person from the standing point of view of animals**, and they will understand human damaging behaviours should be stopped, and then we will embrace a better world.

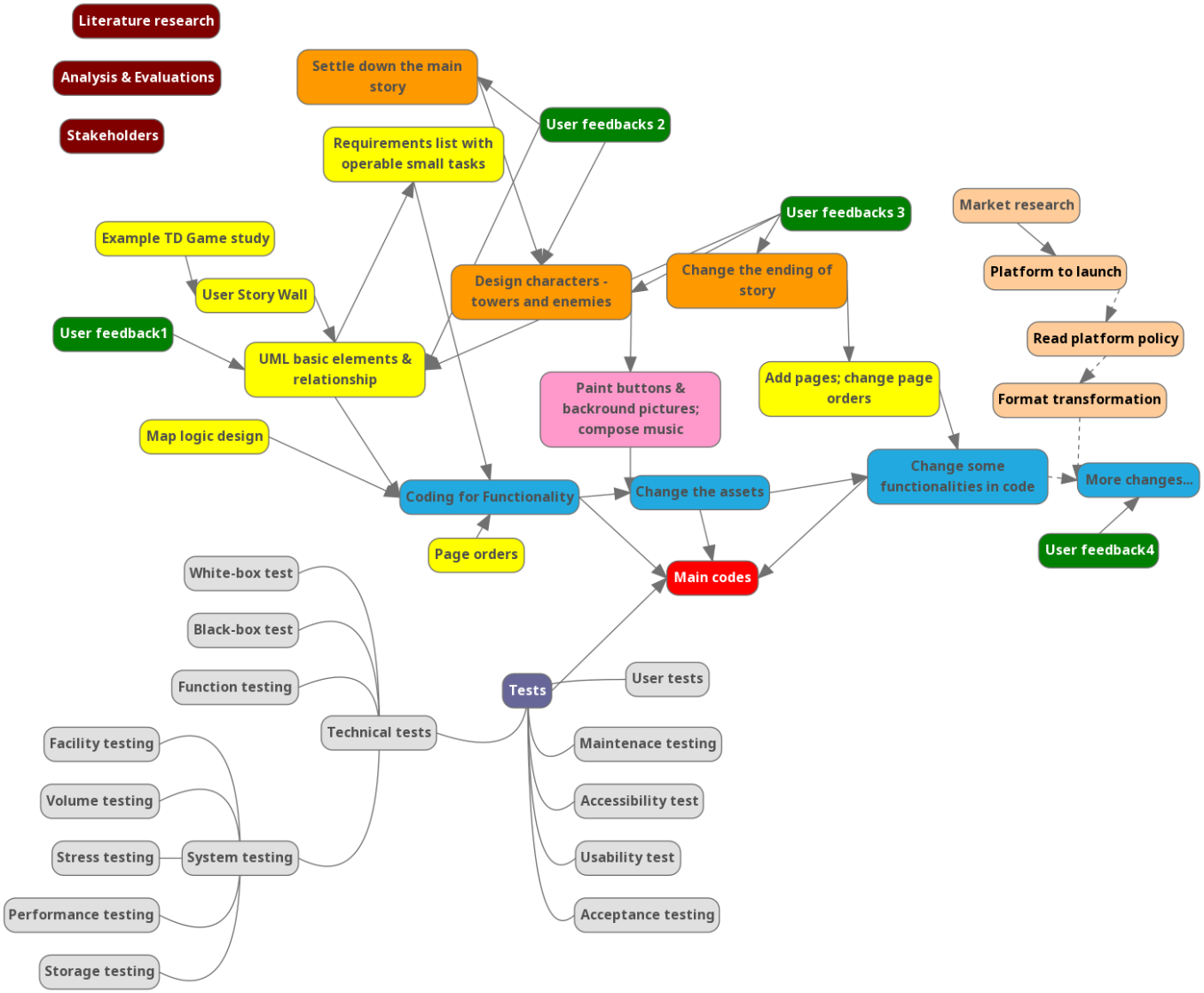
Time management & process

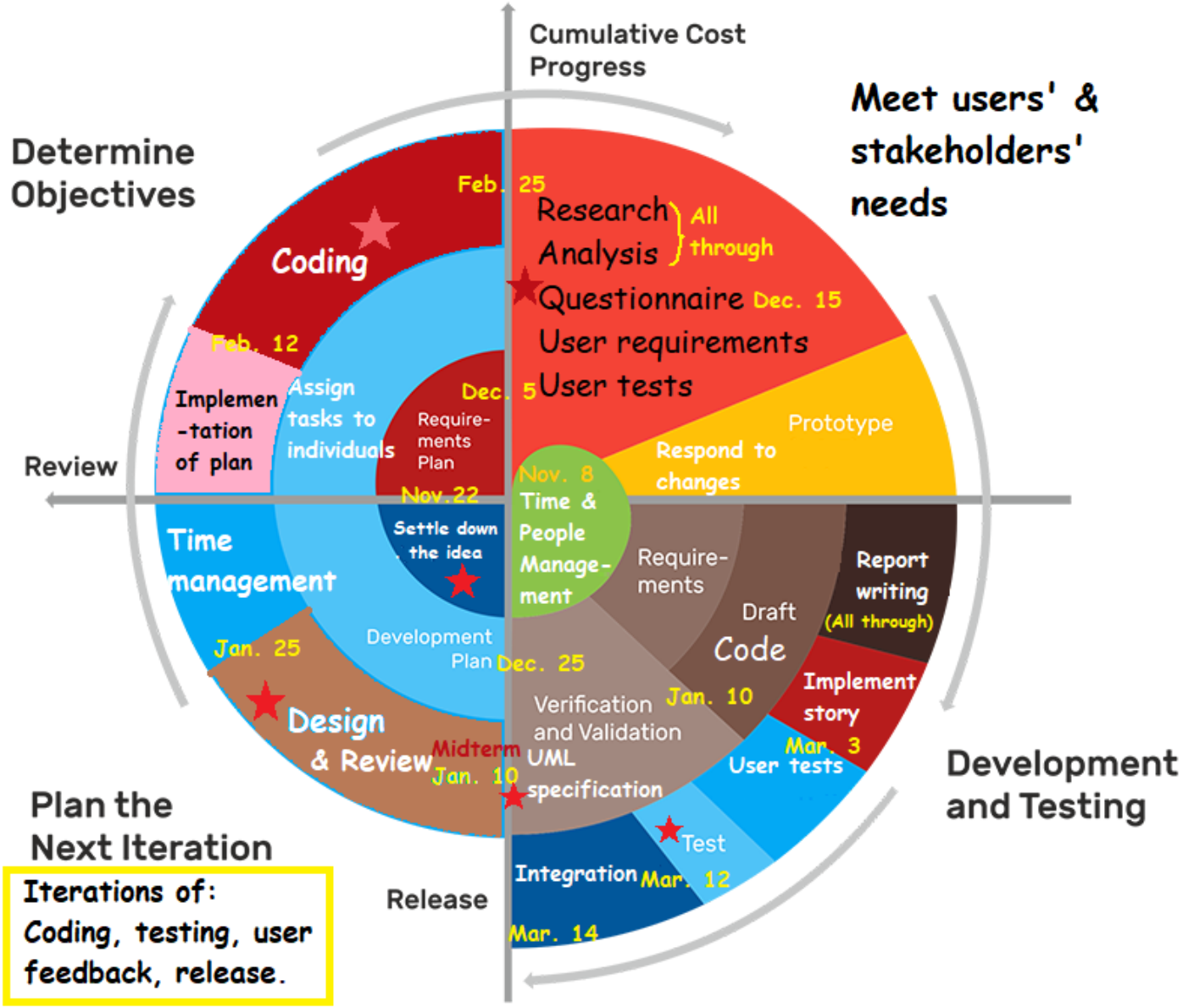


Left image shows our previously set working flow chart. There is a main branch, and several loops that we may go repeatedly. In fact, 'Code realisation' step costs longer time than others.

Below the mind map shows steps and their orders.

Below spiral image shows iterations that we have experienced. Milestones and obstacles are marked with red stars, and deadline dates are marked in yellow colour.


Literature research
Analysis & Evaluations
Stakeholders
Settle down the main story
User feedbacks 2
Requirements list with operable small tasks
Example TD Game study
User Story Wall
User feedback1
UML basic elements & relationship
Design characters - towers and enemies
User feedbacks 3
Change the ending of story
Market research
Platform to launch
Read platform policy
Format transformation
Add pages; change page orders
Paint buttons & backround pictures; compose music
Map logic design
Coding for Functionality
Change the assets
Change some functionalities in code
More changes...
User feedback4
Page orders
Main codes
White-box test
Black-box test
Function testing
Technical tests
Facility testing
Volume testing
Stress testing
System testing
Performance testing
Storage testing
Tests
User tests
Maintenace testing
Accessibility test
Usability test
Acceptance testing

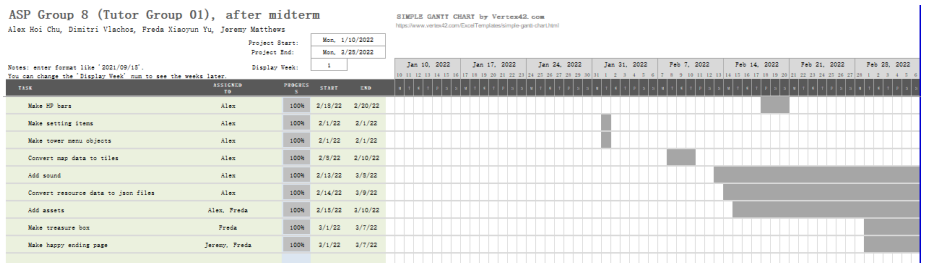


The spiral image just shows broad tasks. We have also written detailed Gantt Charts to allocate subtasks that could be delivered and tracked easily.

Gantt charts:

https://github.com/FredaXYu/ASP_Group8/tree/main/after_midterm/Gantt_chart_finalterm

Snapshot:



People management

Name	Roles	Tasks
Alex	Programmer, Organiser, Software engineer	Coding; UML; specification;
Xiaoyun	Designer, Data analyst, Report writer, Time manager	User tests; accessibility test; usability test; research; data analysis
Jeremy	Programmer, Software tester	Initial coding; prototypes;
Dimitri	Programmer, Report writer, Software tester	Coding; report
Sharif	-	-

There are difficulties from our members. Originally there were 5 members, but from the start to the end, Sharif Khan has never appeared. We four members endeavour all of our energy, but there are inevitable obstacles: Xiaoyun had her family issue from late January to middle February; Jeremy needed to look after his family members who were caught with COVID-19, and he always met storms that caused interruption of traffic and electricity; Dimitri caught COVID-19 himself.

To adjust the original plan, Alex has done almost all of the coding, with the help of Jeremy and Dimitri. Xiaoyun would write papers in February and March. We tailored the original aim – abandon the peripheral settings, and to leave questionnaires and user tests rather later to be conducted. Also, we didn't expect our product could be launched to game platforms before deadline.

Elements break-down

1. User Story Wall

To write codes efficiently, we listed all of the user operations and gave each a weight – to work on basic operations firstly and to postpone the peripherals. To assure the quality of our product to our users, we must complete all the operations listed in 'Release 1 (basic release)'.

User Story Wall:

User activities	**Know the environment**		**Set defense base and play**		**Count scores**	**Report prob**	**Make choices and freedom**		**Purchase**
Tasks	**Read the story**	**Check map levels**	**Set towers**	**Let enemies in**	**See scores and costs**	**Feedback to us**	**Control decorations**	**Navigate fluently**	**Buy things & share**
Release 1 (basic release)	Can open the game	Check 'map' page	Set one tower	Click 'start'	See stats when playing	See 'feedback'		See 'menu' in all pages	Can buy tower w/t an account

	Read intro page	Choose from levels	Upgrade it	See one enemy	See summary after completion	Know who made the game		'Menu' after intro page	Can buy tower based on scores won
		From selection to game	See more tower types	See another enemy				Can resume	
		Know where can put towers	Can sell any tower	Know enemies' paths				Can replay	
			Know towers' property					Can go to the intro	
Release 2	Skip intro	Save previous progress	Treasure box	Can speed up waves		Write email directly	Control music or not	Can see treasure box	Share Sc shots to social media
	Open the game easily	More map bg (grasslands etc.)		Treasure box		See pop-up for errors & quit	Control volume		
Release 3	See the game in game website	Challenging maps for strategy		Based on purchasing, skip waves		Can feedback the errors	Change skins		Create account
	See this game's propagation						Quit game		Tie up a bank card
	See more educational info								Developers receive money
									Watch other games'

								ads
								Time limit for minors

2. UML

The UML is adapted from that in the midterm paper. See here.

3. Specification

Please see Appendix for adjusted specification here.

After settling down the operations we should make, we had a check to the previously written UML and specification document. Finally we started coding, just to implement the functionalities, without aesthetic design implemented.

Analysis

Feasibility analysis

1. Technological feasibility:

Among our 4 members, two are experienced programmers, while the other two have little coding experience, so our knowledge range for technology is limited. However, we guess that there is a big possibility that we can find well-made libraries that are especially for making games. Indeed we have found Phaser library which is discussed in other parts.

2. Operational feasibility:

It's not too promising talking about each member's time and energy. Need to consider special cases which prevents our member(s) from working for approximately 1 month. To inject flexibility, we plan that each person should complete his or her allocated jobs, whether it's early or late. And, others have the ability to continously work on anyone's jobs. We should: (1) backup all the works to Github; (2) notify clearly in Slack group chat which job he has done and where we can find it; (3) ask for any leave in advance.

We anticipate that there would be difficulties when lauching in game platforms – we need to read platform documents, contact with their leader, visit their company since they are our key stakeholder, etc.

3. Financial feasibility:

Unless we have to release our game to game platform, we don't need to cost a lot, since most are coding jobs. If we only make questionnaires in mainland China, then it's free; but in western countries which

we care more about, questionnaire websites may charge us fees. But it would be small amount, since our project won't last long.

4. Legal feasibility:

(1) Game platform contract:

Before we get to know the platforms, we will not have any clue about this. Perhaps it doesn't allow scenes with violence or politics. We are safe since our game is cartoon. We need to be cautious whether our game would be tagged with 'antihuman' since it's a game to prevent human invasion. But we have our reasons: the theme is anti-bad-behaviours, instead of anti-human; behaviours are not equal to the entity; once bad behaviours are corrected, humans would get improved. We are not killing any people. We should not let any human entity to be the enemy. This is what we want to convey.

Potential platforms: TapTap, Apple Game, Google Play, Steam, etc.

(2) Laws in different countries:

Many countries have isolated Internet scopes. Since one of our members is in mainland China, we hope to attract Chinese users. We notice that game platforms are separated as 'mainland China version' and 'others', which means we need to (1) do translations for our game; (2) launch two versions in each platform; (3) use phone numbers and emails in different countries for different accounts. All should be assigned to a professionally organised team to release. If we do the releasing job, we need to plan more.

In addition, many countries have laws to restrict game playing time for minors. According to that, we must plan for: (1) account system – log in page, database; (2) time alerts poping up every 30 min; (4) forced to terminate when certain amount of time reaches.

Risk analysis

Throughout our progress, we are doing risk analysis.

Risks	Solutions
Specification risk: (1) might understand wrongly, so the implementations are in wrong diretion; (2) in testing, we might omit some specifications.	Track specification often.
Test cases might be incomplete (exception handling, margins, etc.)	Think roundly.
Some errors are occationally to occur, which might be untracked.	Screen shot often.
Loss of previous versions.	Use Github.
Users might get lost before we give them tutorials.	Tutorials can be planned in the future.
No time to summarise user test (feedback) result.	State the important feedbacks.

Design

Technological design

Since our time is limited by this programme, we cannot complete all the tasks listed in previous sections.

We must complete the tasks listed in 'Release 1' in 'User Story Wall', before Mar 14, 2022.

Dimitri has investigated potential engines that we could use, including Box2D (2D), Toxiclibs (2D + 3D),

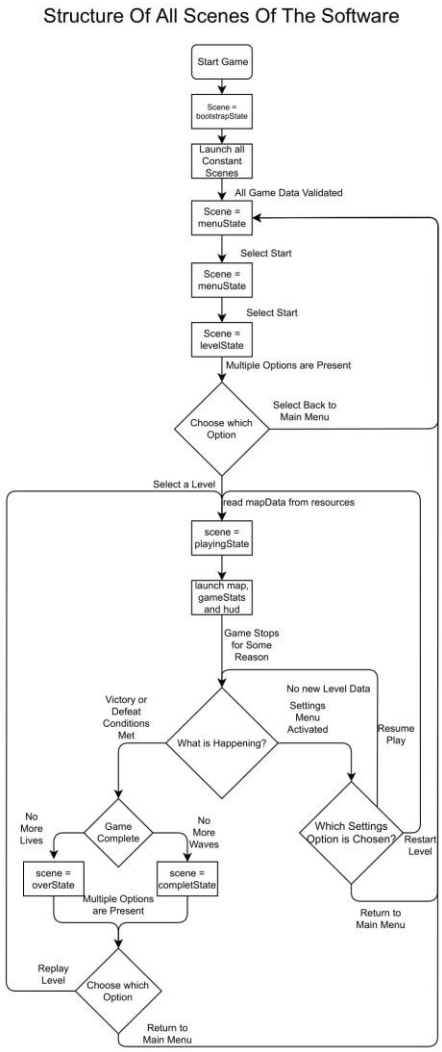
We choose **Phaser library** https://phaser.io/ to aid our development. Phaser is a fast, free and open source HTML5 game framework that supports both WebGL and Canvas rendering modes and can run in any web browser environment.

We choose it because: It's easy to handle tile maps; can implement physics (characters similar to real world); can live show in browsers; it has embedded modules to handle fires and bullets; etc.

We design our code structure to be **in hierachy**, with each folder aiming at a common goal, and each file specialising in one functionality. E.g. We should have: (1) a folder or file to handle all of the importing resources; (2) a folder to store all of the assets; (3) **the main folder 'src'** containing main codes – i. All of the pages users will see; ii. All of the elements in the game playing map; iii. All of the storage elements that are not arranged based on user interface but are based on map/HUD logic or based on specifications, for example a file handling enemies and their speeds, a file handling towers and upgrades and bullets; (4) a test file for debugging; (5) an index.html. The hierachy is gradually perfected during development.

The files contained inside 'src' should be created based on the newly updated **Specifications**: https://github.com/FredaXYu/ASP_Group8/blob/main/after_midterm/finalterm_proposal/Requirements_Table_updated_Alex.doc.

We plan to use **Chai, Mocha, and Sinon** to do **technical testings**. (1) Sinon.js is a tool useful when writing Node software testings. 'spy' in Sinon can monitor how a function is called. 'stub' in Sinon is flexible when testing one function which relies on another function. 'mock' in Sinon can monitor the activities of an object. (2) Mocha is a frame of JavaScript unit testing, and Chai is a library of assertions working along with Mocha.



Based on the User Story Wall, We have slightly changed the previously drawn system developing flow chart to adapt to the new design.

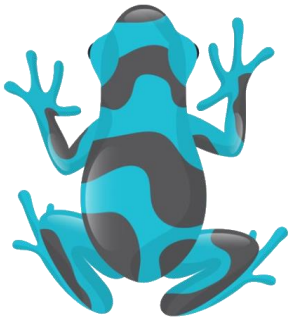
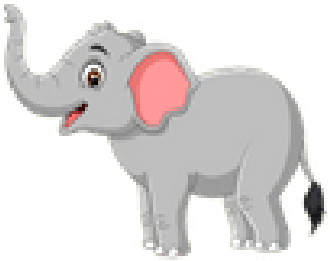
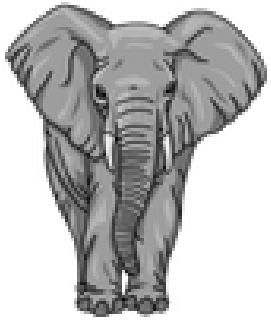
We have adjusted previously made 'Components' file here. The 'Components' file contains all of the objects we need to create, and their descriptions and relationships. This is what we based on to write prototype codes.

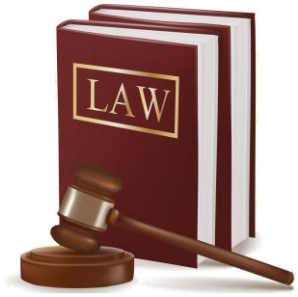
Aesthetic design

We use cartoons and stick figures because they look cute and casual. Just simply draw a button in Paint software, and adopt the same style to other buttons. According to testing users here, they like the cartoon theme very much, so we have confirmed our idea. However some users are not satisfied with our aesthetic design here, with opinions that it should be simple with colours and shapes.

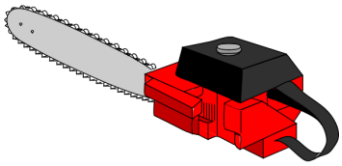
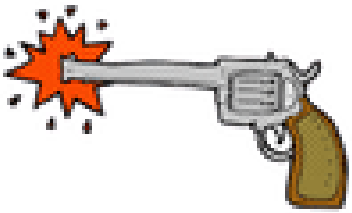
Character design

Tower types	Evolution 1	Evolution 2

Poison frog		
Tiger		
Piranha		
Elephant		
King cobra		

Law (the most powerful and triggers the happy ending)		

Since there are fierce animals shown in the character list, we have collected users' feedbacks. More than half of the key stakeholders support our characters (even some users give them 5 stars – full score), so we don't need to change them.

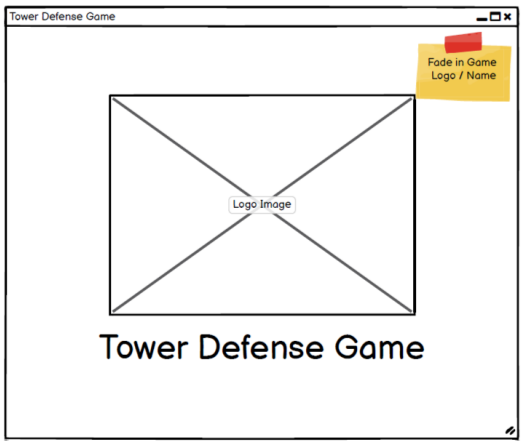
Enemy types	Evolution 1	Evolution 2
Plastic garbage		
Saw		
Hunting gun		

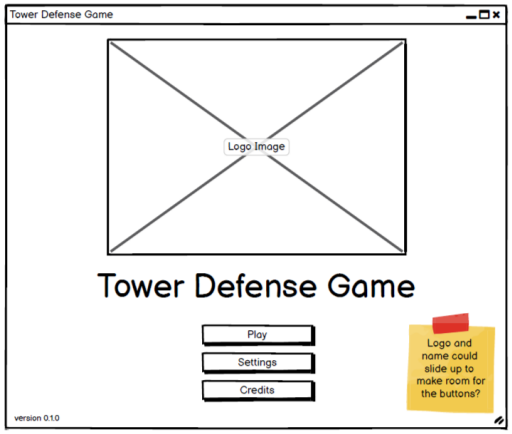
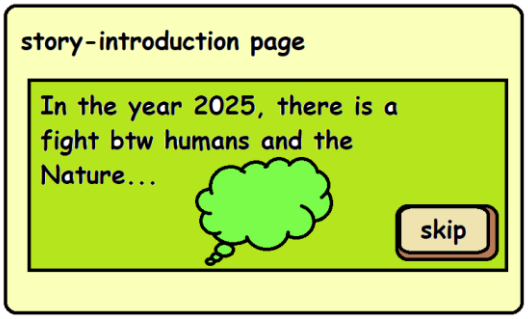
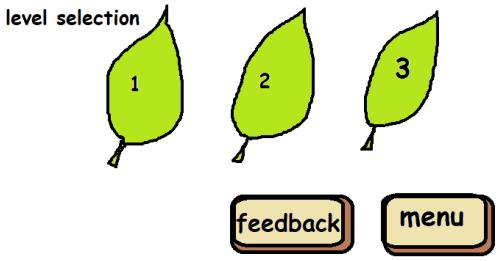
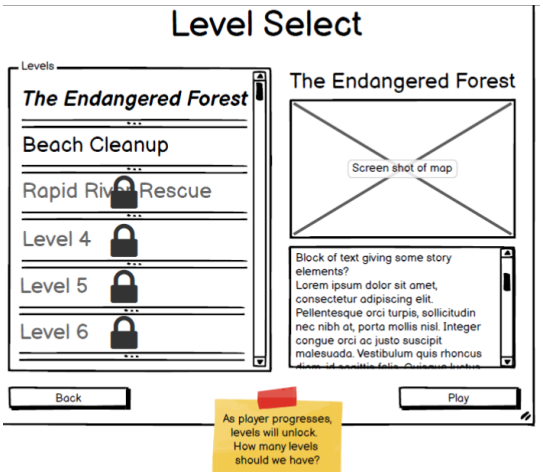
In order to get a broad idea about what our project should contain, we firstly gestated a general structure that could be used. The main idea is to make an **environment-protection game** to teach users to live

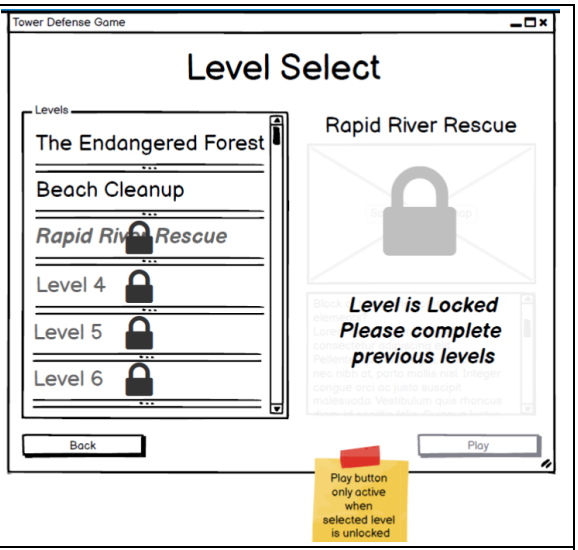
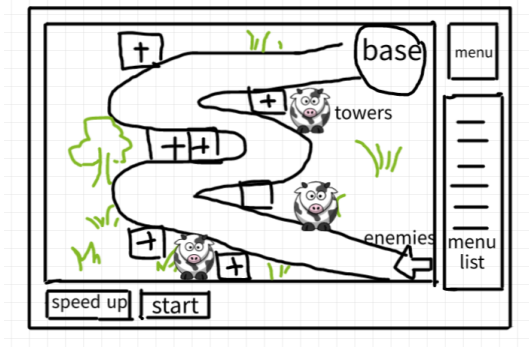
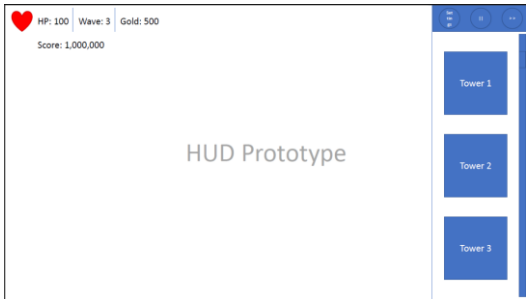
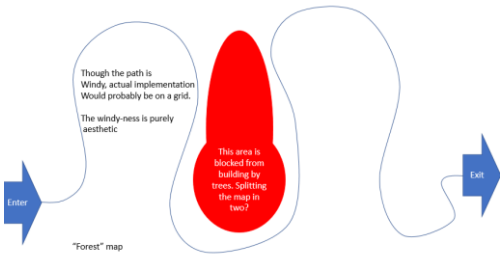
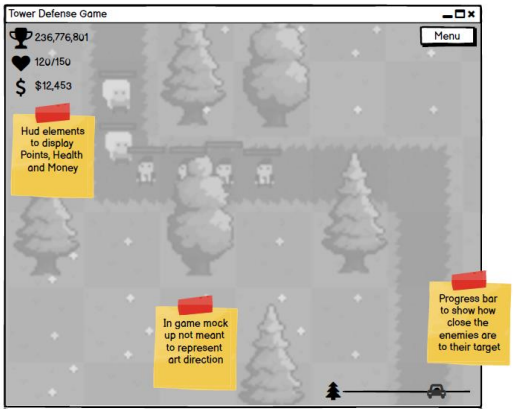
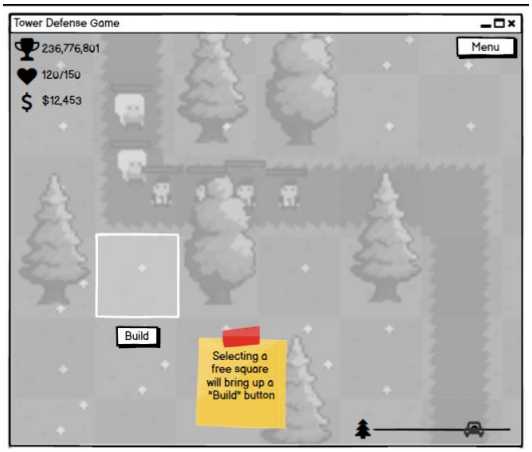
harmoniously with Nature.

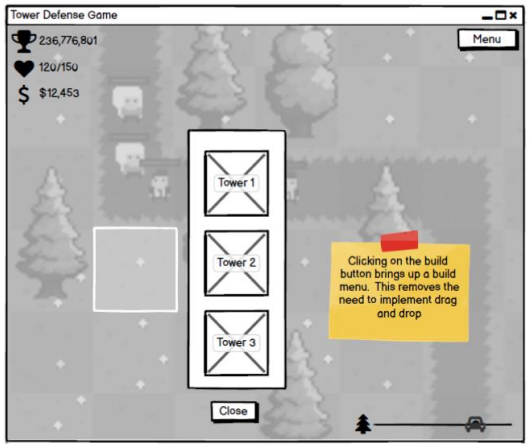
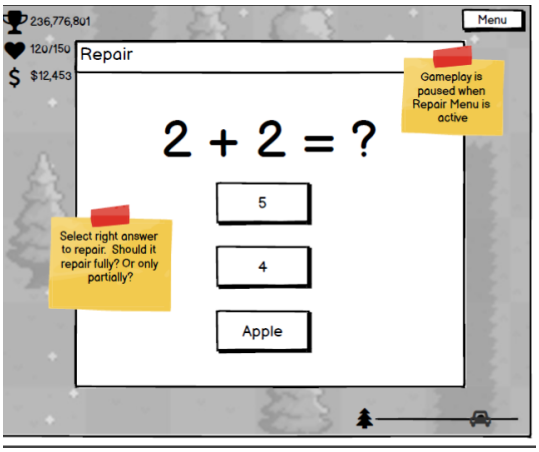
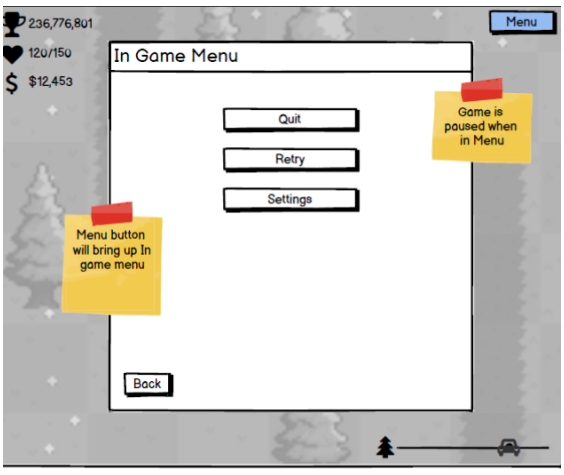
User Interface

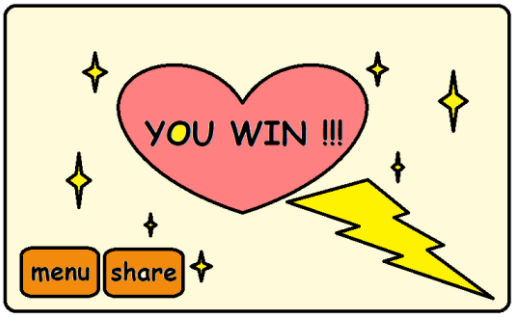
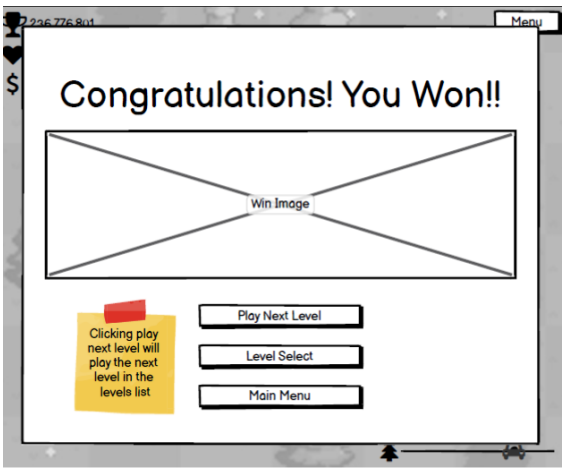
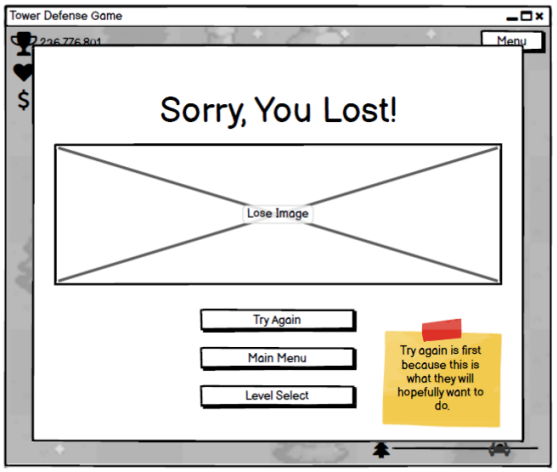
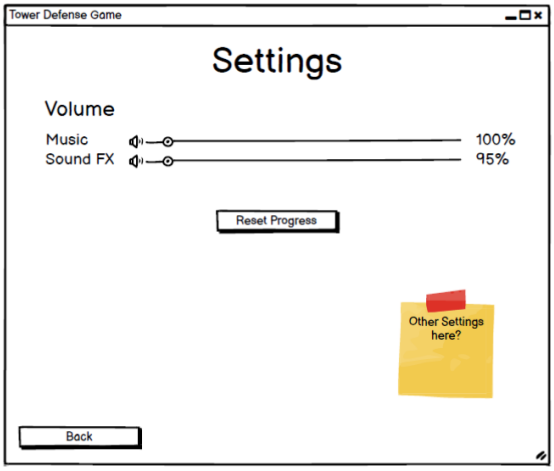
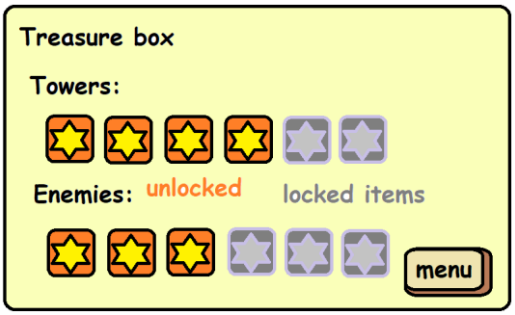
We then drew some simple design **ideas** for each page users may see:

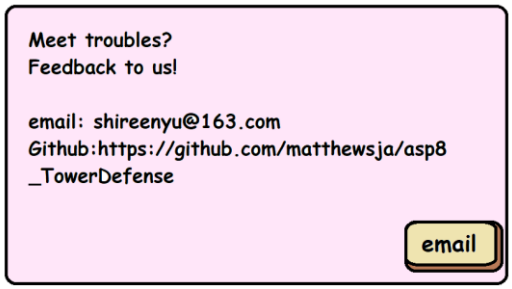
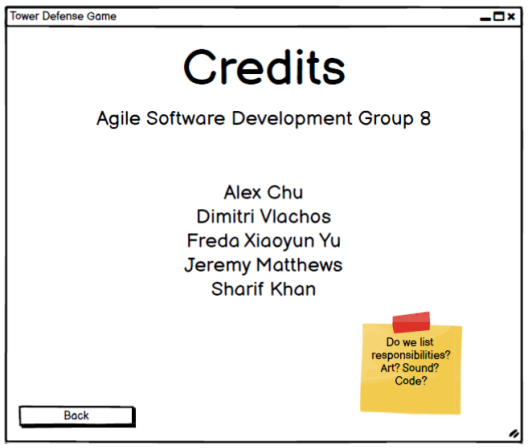
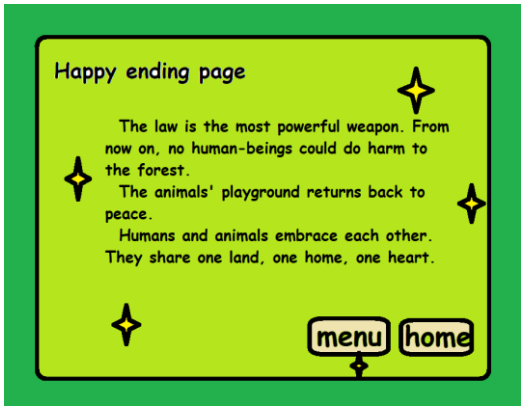
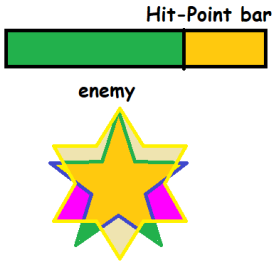
<table>
<tr>
<td>Logo page.</td>
<td> 


</td>
</tr>
<tr>
<td>Story Intro page, should let users know the backgroud of our story setting.</td>
<td> 
</td>
<td></td>
</tr>
<tr>
<td>Level Selection page. We should add jungle elements.</td>
<td> 
</td>
<td> 
</td>
</tr>
</table>

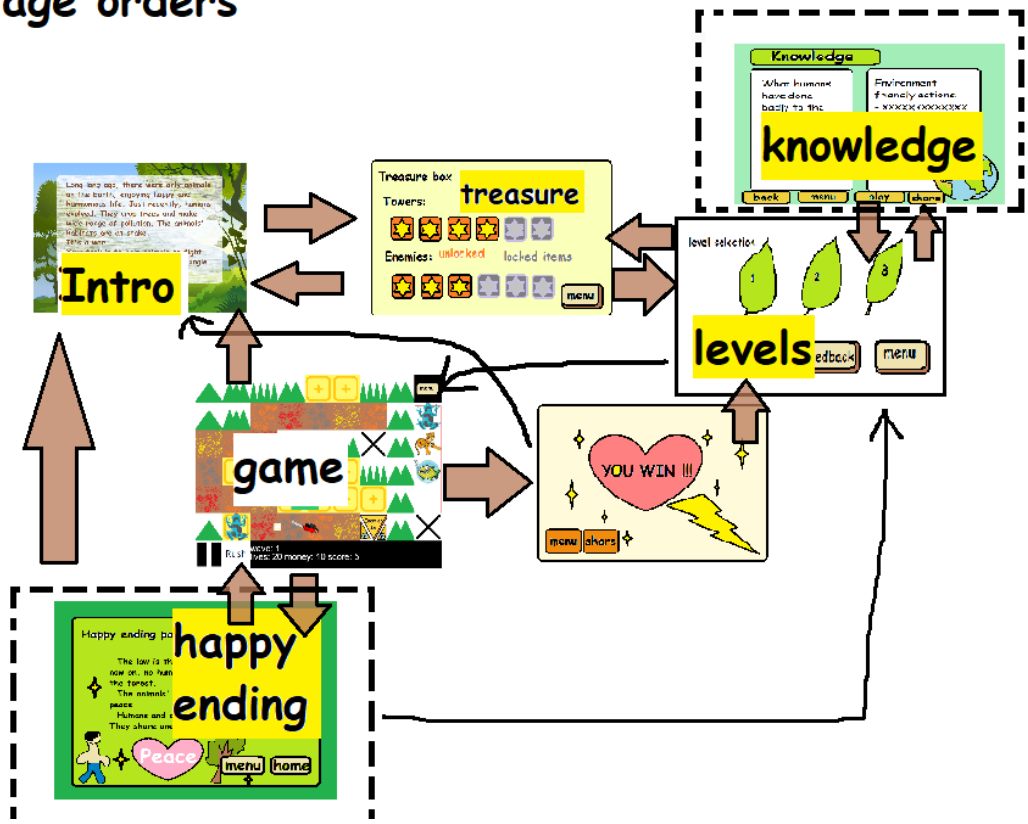
		
In the game playing page that we focus on, we plan to put the main map in the center, and leave the HUD to the side. HUD should contain: game stats (money, wave of enemies, lives, enemy HP, score); tower selections; speed changing buttons; start button to let enemies in; menu button. We have designed some ideas for the map.	  	 

	 menu Rush wave: 1 lives: 10 money: 11 score: 3	 Tower Defense Game 236,776,801 120/150 $12,453 Menu Tower 1 Tower 2 Tower 3 Close Clicking on the build button brings up a build menu. This removes the need to implement drag and drop  Repair would bring up the Educational part? Ask Math question and answer right to repair Repair Sell Sell will instantly sell / destroy the tower in this square.  236,776,801 120/150 $12,453 Menu Repair Gameplay is paused when Repair Menu is active 2 + 2 = ? 5 4 Apple Select right answer to repair. Should it repair fully? Or only partially?
menu		 236,776,801 120/150 $12,453 Menu In Game Menu Quit Retry Settings Game is paused when in Menu Menu button will bring up In game menu Back

Level completion page: should be rewarding and encouraging for the players. It should also display the navigation buttons. Can add game stats.		
Game over page.		
Settings.		
Treasure Box page. Should show all levels including locked levels (perhaps in developing and users never know but it's		

promising).		
Feedback Page. Steps are: (1) our information; (2) info + 'email' button which pops up email box; (3) info + feedback form which can submit to us.		
Ending page, triggered by placing 'law' the tower. Humans give up immediately. Our idea is to embrace each other and to live harmoniously in the end. This is the best result.		
Enemies should have a Hit-Point bar.		

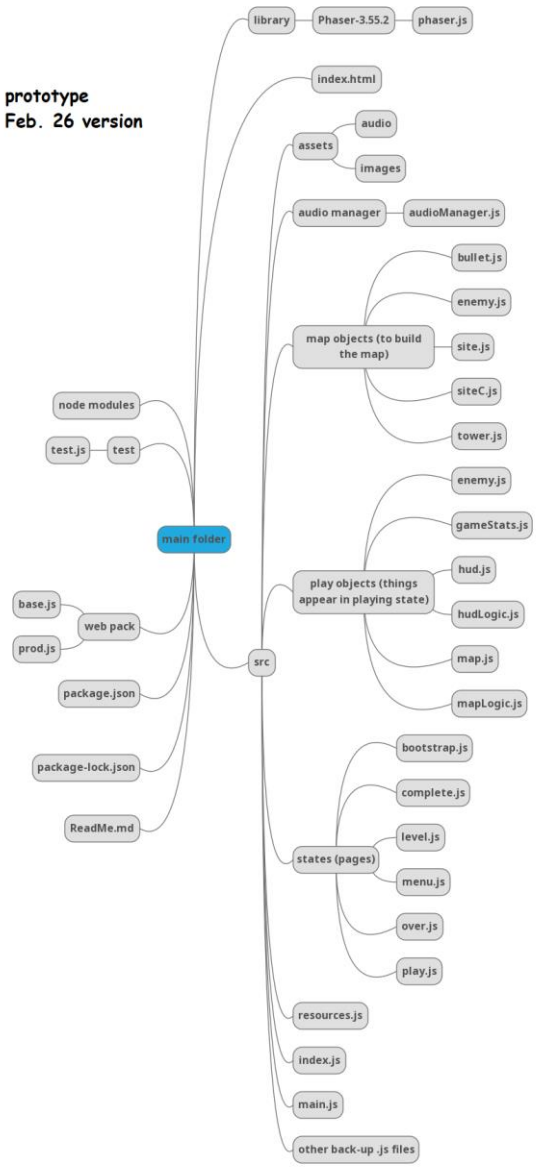
Page orders



Prototyping & Iteration

Based on Jeremy's initial codes, Alex has in mid Feb. developed a prototype that has realised all the basic functionalities of a tower defense game. We may later insert stories and adjust assets to rich the content.

The Github link for our codes of this software: https://github.com/matthewsja/asp8_TowerDefense/



Here is the structure of the **prototype**:

All the files are classified into folders so that their arrangement is meaningful and logically consistent.

'index.html' is used to open up the webpage in the browser.

'library' folder contains the Phaser library.

'node modules' lists all of the necessary functionalities of the node.

'test' folder contains the test files that will simplify the debugging process and future technical testings.

'web pack' folder manipulates web-based functionalities.

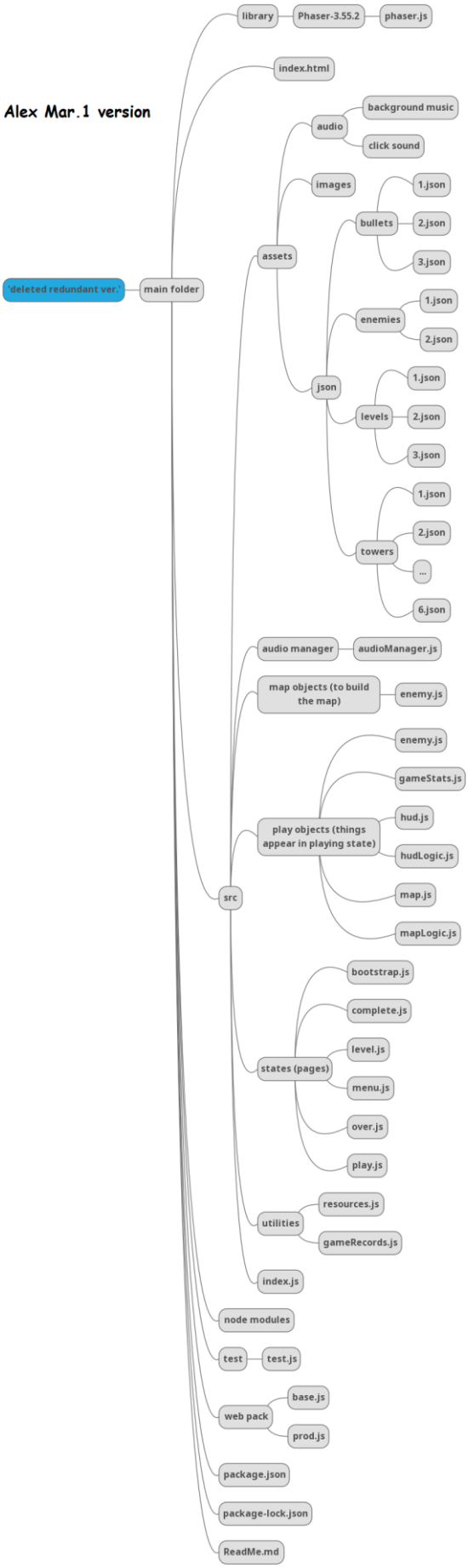
'src' folder contains all of the crucial codes. 'assets' folder contains all of the images and audios that we use. '**map objects**' folder contains all elements that needed to construct the map in the main play screen – there are: tower, bullet, enemy, and the site. '**play objects**' folder mainly manipulates interactions that users may trigger and the outcomes to display in the play screen. '**states**' folder handles the pages that display to users; whenever we need a new page, we add a new file here. 'resources.js' file is used to import all of the assets and other .js files. 'index.js' and 'main.js' files now are rather overlapping.

There are also other back-up .js files contained in the 'src' folder that need to be deleted.

There are also package.json files which is used to open this project in Node.js.

Features: can put towers to sites; towers can emit bullets; enemies come in correct path; enemies can be killed; can buy and sell towers; game stats are correct; several waves of different enemies; can select levels but logic is confused; can display game stats and menus but is mixed with the map; 2 types of tower; can change speed but button is not meaningful; has a level-completion page; has a game-over page; use a tile set to construct the map; bg music plays well.

The prototype is perfected by steps: On Feb. 6, Alex updated a change to the HUD to **let users change speed (since users mentioned this before midterm)**; and added some buttons that spawn the enemies in waves instead of individually. On Feb. 7, Jeremy wrote the **audioManager file** to handle audio plays. On Feb. 11, Alex updated a version in which the **maps are constructed by tiles** to be manipulated more easily; also imported an outside library to **handle the enemies' path**. On Feb. 15, Alex add **level-selection** functionality. On Feb. 18, Alex added AOE towers; also restructured the resource class so all the importing is done in the preload, and has met a problem when writing JSON files to include a dictionary to manipulate the resources. On Feb. 20, Alex added **Hit-Point bars** for each enemy. Some changes are done in a branch, not merged to main branch yet.



The Mar.1 version file updates:

Added JSON files inside 'assets' to make configurations for each object (levels, bullets, enemies, etc.), so that whenever mouse hovers on any character, it shows information. This corresponds to the 'Recognition Rather Than Recall' rule mentioned by Nielson's 10 Usability Principles.

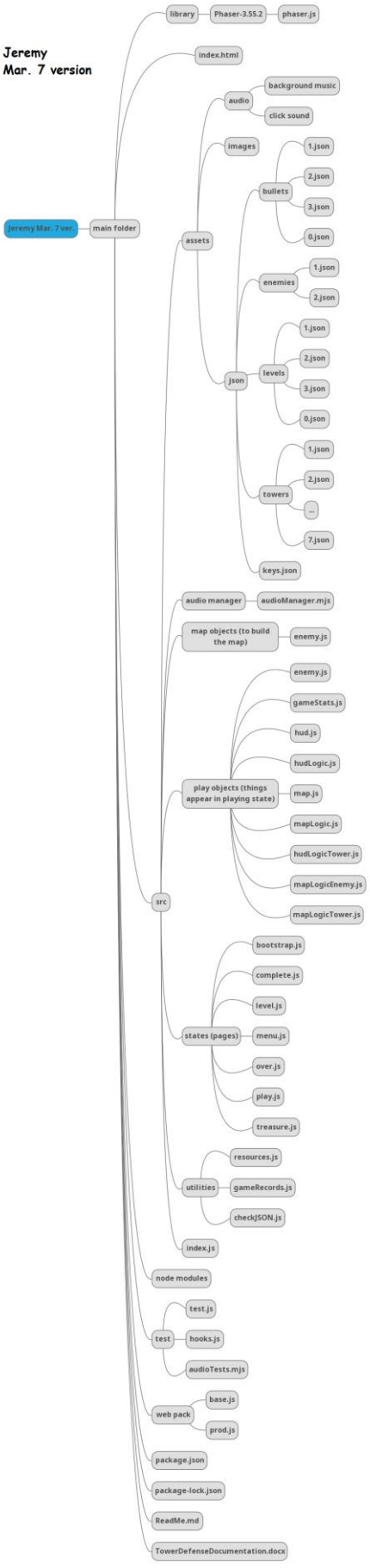
Shortened the 'map objects' folder.

Refined the map and HUD logic.

Resource file is restructured.

'resouce.js' file and 'gameRecords' files are classified into 'utilities' folder.

Features: the menu and other buttons and game stats are separated from the map in the main playing screen; added HP bars; level-selection page is well-displayed; clicking sound; 'start' button in the playing page is obvious; 3 types of tower; speed-change buttons are meaningful to new users; level-completion page is meaningul with game stats; game-over page is meaningful; 3 levels are in different difficulties; each object can show description whenever mouse hovers on it.



Mar. 7 version updates:

'utilities' folder newly added 'checkJSON.js' file which realised the dictionary functionality for resources mentioned by Alex previously.

Xiaoyun has implemented the environment-protection story by changing the assets, since users need a good story.

Has added 'treasure' state to summarise all of the towers and enemies for users, since users need hints, instead of memorisation.

Jeremy has added the happy ending page which is triggered by placing the LAW tower, since we want it to be educational and peaceful.

Features: story has been implemented – updated asset pictures; the 'Law' tower is added; 'happy-ending' screen is made by Jeremy and Xiaoyun, which will be triggered whenever user puts 'Law' onto the map; Jeremy also added clicking sounds to all buttons.

On Mar. 4, Jeremy has **generated a link** for our game to be open easily in any browser. Later he has updated this link based on new versions.

Version progress in Github see here.

System Development

User-centered

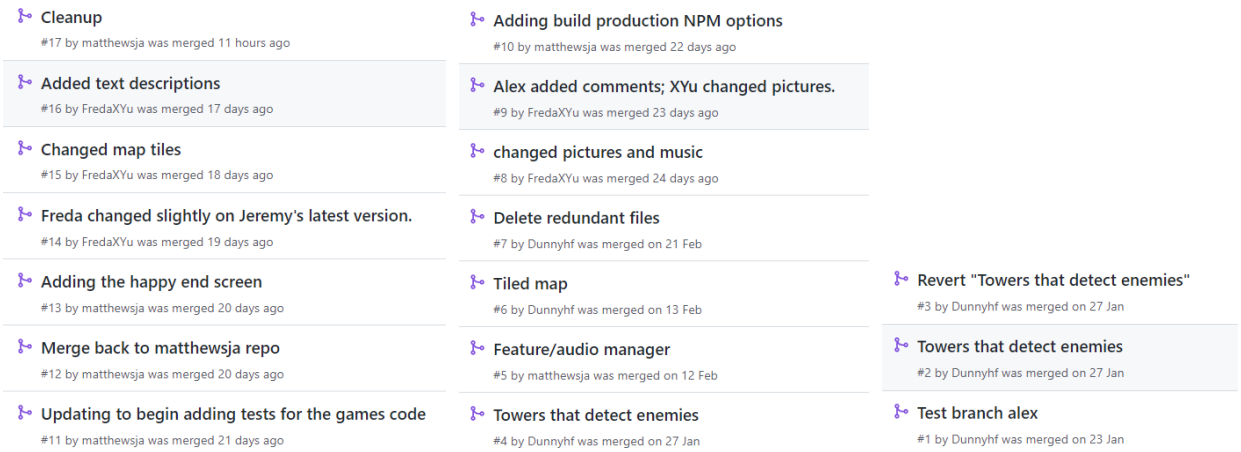
Based on user opinions in Dec. 2021 and the User Story Wall that we proposed, we have written and updated our Requirements for the project. The **User Story Wall** contains all the short events (functionalities) that users will experience, **most of which are compulsory** without the need to ask user opinions. Based on **Dec. 2021 user opinions**, we have: (1) designed several levels with different difficulties; (2) added some guidance for users who are not familiar with TD games; (3) designed the characters as **cute and war-related** according to the user preferences; (4) the ability to speed up attacks; (5) 'more skills and magic' – developed 'the Law' tower which can trigger the happy ending in any cases; (6) smiplified characters to let users memorise less things; (7) added 'plants (on map), animals (towers)' elements; (8) beautiful and bright scene with the settings in a jungle; (9) still point of view; (10) reward mechanism after each level; (11) some sense of design with a united style of doodling.

Besides, for each turn of code updates, we invite users to experience and give us feedbacks, including several 'permanent testing users'.

Github Usage

The complete codes are here: https://github.com/matthewsja/asp8_TowerDefense which is a different repository from the one we have created before midterm. The reasons we create a new repo: (1) A new repository is portable for members to handle using git operations, since here are all the codes without big files; (2) previous repository's owner Xiaoyun has had some familiy issues that she couldn't manage any updates temporarily; (3) the new repository's owner Matthews Jeremy is an experienced programmer, so he will update and manage the code repo well.

The 'main' branch is for Jeremy to update any key versions and for others to merge their branches to. All members have contributions to this branch.



The 'Dunnyhf-patch-1' branch is created by Alex to write any updates.

The 'testingSetup' branch is created by Jeremy to set up technological testings.

Documentation of Codes

Now we introduce the codes of the latest version.
Hierachy:

- Mar. 27 ver.
    - main folder
        - library
            - Phaser-3.55.2
                - phaser.js
        - index.html
        - src
            - assets
                - audio
                    - background music
                    - click sound
                - images
                - json
                    - bullets
                        - 0.json
                        - 1.json
                        - 2.json
                        - 3.json
                    - enemies
                        - 1.json
                        - 2.json
                    - levels
                        - 0.json
                        - 1.json
                        - 2.json
                        - 3.json
                    - towers
                        - 1.json
                        - 2.json
                        - ...
                        - 7.json
                    - keys.json
            - audio manager
                - audioManager.mjs
            - map objects (to build the map)
                - enemy.js
            - play objects (things appear in playing state)
                - enemy.js
                - gameStats.js

            - hud.js
            - hudLogic.js
            - map.js
            - mapLogic.js
            - hudLogicTower.js
            - mapLogicEnemy.js
            - mapLogicTower.js
        - states (pages)
            - bootstrap.js
            - complete.js
            - level.js
            - menu.js
            - over.js
            - play.js
            - treasure.js
        - utilities
            - resources.js
            - gameRecords.js
            - checkJSON.js
        - index.js
    - node modules
    - test
        - test.js
        - hooks.js
        - audioTests.mjs
        - enemyTests.mjs
        - stateTestUtils.mjs
        - bootstrapStateTests.mjs
    - web pack
        - base.js
        - prod.js
    - package.json
    - package-lock.json
    - ReadMe.md
    - TowerDefenseDocumentation.docx

See code description file **here**, or download it from:
https://github.com/FredaXYu/ASP_Group8/blob/main/after_midterm/finalterm_proposal/peripheral_files/final_files_submit/ASP8_Tower_Defence_Code_Documentation_Mar28.pdf

We have done **technical tests** when developing codes, by creating a testing branch in Github: https://github.com/matthewsja/asp8_TowerDefense/tree/testingSetup The details see here.

Our project is **Agile**:

Individuals and interactions	We value individual value over the routines. Alex can do broad-scale software plan; Xiaoyun is good at data analysis and customers; Jeremy and Dimitri are professional in game development. Each individual may have issues, but when one is down, others are working.
Working software	After we have settled down using JavaScript and Node to build the game, we all started working. Github is useful to look after each member's needs. Documentations are later written, since when working, we put 100% energy in writing codes.
Customer collaboration	We don't force stakeholders to sign any contract, but instead, we invite them to be our permanent guest to provide feedbacks. We work towards the expectations of these feedbacks.
Responding to change	Time schedule is flexible. Customers' ideas may change. We may have new ideas. Individuals are happy to know new requirements and build new functionalities, e.g. the 'treasure box' and 'happy-ending page'.

Coupling & Cohesion

Coupling:

We have separated our task into several .js files, with each one emphasizing one functionality. To do so, we create separated classes all derived from Phaser.scene class. Unless inside the class we need to use picture assets (background, button design, etc.), there are rare elements calling data in other .js files.

Indirect Coupling:

There are no direct calling among modules. Example:

20 //this allows usage of attributes and functions from another scene 21 var gameRecords = this.scene.get('gameRecords') 22 **In over.js** 20 //this allows usage of attributes and functions from another scene 21 var gameRecords = this.scene.get('gameRecords' 22 **In complete.js**	In src\utilities\gameRecords.js file, we can find no calls for elements in other modules, although in other files, we can see they call this gameRecords class data. gameRecords.js is a rather independent module.

Common-environment Coupling:

Use the same data input. Examples:

129 'start': { 130 fileType: 'image', 131 path:'src/assets/images/start.png' 132 }, 39 //create an image that does something when clicked on 40 var start = this.add.image(450, 620, 'start').setInteractive() 41 start.displayWidth = 200 42 start.displayHeight = 160 **In menu.js file**	The .\src\utilities\resources.js file is created in order to load picture, sound, and other assets apart from functionality operations. Several classes (.js files) all use the same picture

`//create an image that does something when clicked on`
`var startButton = this.add.image(600, 620, 'start').setInteractive()`
`startButton.displayWidth = 150`
`startButton.displayHeight = 120`
In treasure.js file
	which is loaded in it.

Data Coupling (Feature Coupling):

Data is transferred from one module to another. To call it 'Feature Coupling', since the data type transferred is class or file, instead of a simple data type like int, float, etc. Examples:

`restart.displayWidth = 200`
`restart.displayHeight = 100`
 	
`//when these images are clicked on, the scene of the game changes depending on`
`//this takes the game to the main menu scene`
`menu.on('pointerdown', function () {`
`this.scene.scene.start('menuState')`
`})`
`//this takes the game to the level select scene`
`level.on('pointerdown', function () {`
`this.scene.scene.start('levelState')`
`})`
`//this takes the game to the play scene with the previously set level data`
`restart.on('pointerdown', function () {`
`this.scene.scene.start('playingState')`
`})`
 	
`class MenuState extends Phaser.Scene`
`{`
`constructor ()`
`{`
In menu.js file
	./src/states/complete.js, we create three buttons, and insert contents when they are clicked. The pages (states) we will jump to, are nother already created classes 'menuState', 'levelState', 'playingState'.
	In fact those states are **classes** in other .js files, which represent all the pages we will see in game.
	In fact, each page contains some page directions to other pages. These directions must call classes written in other modules.

`import HUDLogic from './play objects/hudLogic.js'`
`import HUDLogicTower from './play objects/hudLogicTower.js'`
`import GameStats from './play objects/gameStats.js'`
 	
`//set different configurations for the game`
`const config = {`
`type: Phaser.AUTO,`
`parent: 'phaser-game',`
In index.js file
	The index.js file functions as the leader. It imports all the other sub-files, and sets configurations.

Cohesion

Our project uses **Functional Cohesion** most often, which means inside each module, all the elements contribute to one functionality.

Take .\src\states\level.js file as an example. All the elements contribute to the 'level selection' page.

`import AudioManager, { SFX } from '../audiomanager/audioManager.mjs';`
 	
`class LevelState extends Phaser.Scene`
`{`
`constructor ()`
	The whole file is one class.

`create ()`
`{`
`//these prevent the active scenes used in playing the game from covering the`
`this.scene.stop('gameStats')`
`this.scene.stop('hud')`
`this.scene.stop('map')`
 	
`// The title, added by XYu Mar5:`
`this.bar = this.add.image(450, 100, 'level_selection_bar')`
`//////////////////////////////`
	Inside 'create' method, firstly stop the functionalities in previous scenes. Then, load the picture.

`//these allow usage of attributes and functions from other scenes`
`var resources = this.scene.get('resources')`
`var gameRecords = this.scene.get('gameRecords')`
`var levelSelect = this.scene.get('level')`
	Get the elements in other scenes to be used in this scene.

`//creates an image that could be clicked on`
`var back = this.add.image(50, 50, 'left').setInteractive()`
`//when the image is clicked on the game would be the menu state`
`back.on('pointerdown', function () {`
`this.scene.scene.start('treasureState');`
`AudioManager.playEffect(SFX.BUTTON_CLICK);`
`})`
	Create the 'back' button at the top left. Set a link to where it leads to. Add sound effect.

`//this function positions the buttons which when clicked on would take the player`
`this.levelButtons = function(){`
`//iterates over all the levels that could be played`
`for(var i = 0; i < resources.levelList.length; i++){`
`var x`
`//the y position is preset to this as there are not enough levels to worry about`
`var y = 250`
`//the x position of the button depends on the index in the for loop`
`//so far, this only accommodates five levels`
`switch (i){ // XYu has changed these values Mar 5`
`case 0:`
`x = 100`
`break;`
`//call the function to actually make the button with position data based on the p`
`this.makeButton(resources.levelList[i], x, y)`
`//this function is used to make a button for each level that could be played`
`//takes in as parameters 1)the key of the level and the 2)x and 3)y coordinates of where t`
`this.makeButton = function(level, x, y){`
`//makes an image and makes it interactive`
`//the image is the image chosen by the level from the loaded images`
`var levelButton = this.add.image(x, y, resources.levels[level]['selection']['p`
`//takes the description of the level and saves it to the image`
`levelButton.description = resources.levels[level]['selection']['description']`
	Create level buttons with each leads to a specific level. Then load images for each button and add descriptions when mouse hovers.

`//function to handle what happens when a level button is clicked on`
`//takes in as a parameter the name of the level`
`this.clickButton = function(level) {`
`//sets the value of the selected level in gameRecords to the name of this level`
`gameRecords.levelSelect = level.toString();`
`//finds the level in resources based on the name and sets the current level to it`
`resources.mapData = resources.levels[gameRecords.levelSelect];`
`//changes the scene to the playingStats scene`
`this.scene.start('playingState');`
`AudioManager.playEffect(SFX.BUTTON_CLICK);`
`}`
	When clicked one level button, load the corresponding game stats and the map data. Add sound effect.

Testing

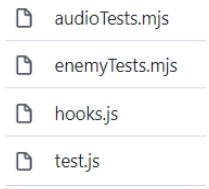
Technological testing

The test files can be seen here: https://github.com/matthewsja/asp8_TowerDefense/tree/main/test

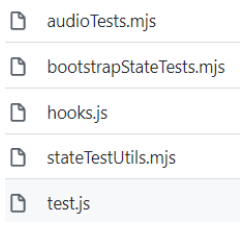
On Mar. 5, we have built 2 tests. In the end, Jeremy has added 2 more testing NPM packages so we will need to run **npm install** after pulling the latest version. The new packages are **Chai**, and **Sinon**. These 2 packages work with **MochaJS** to add more functionality to the tests. Chai is an assertion library which makes

checking the output of tests really easy. Sinon is a mocking library which allows for adding in fake functions and objects. In Jeremy's initial tests for the audio manager, he uses Sinon to pretend to be the Phaser objects that the manager expects. Then he is able to make sure that our code is calling the Phaser code as expected. Since we did not write Phaser, we do not need to be too worried about testing Phaser and only need to make sure our code is using it as expected. Then on Mar. 10, Dimitri has updated the enemyTests.mjs file.

Before Mar. 27, the testing files in our main branch:



Before Mar. 27, the testing files in our testing branch:



Snapshot for each testing file:

audioTests.mjs

```
import { assert } from 'chai';
import sinon from 'sinon';
import AudioManager from '../src/audiomanager/audioManager.mjs';

// Generates a mock Phaser Sound Manager object
function createPhaserSoundManagerMock() {
  let phaserSoundManager = {
    context: {
      state: 'suspended',
      resume: function() {}
    }
  };
  sinon.stub(phaserSoundManager.context, "resume").callsFake(() => 'mock');
  return phaserSoundManager;
}

// Generates a mock Phaser Loader object.
function createMockPhaserLoader() {
```

enemyTests.mjs

```
import { assert } from 'chai';
import sinon from 'sinon';
import Enemy from '../src/map objects/enemy.js';

// Generates a mock Enemy objecy
function createEnemyMock() {
  let testEnemy = {};
  sinon.stub(testEnemy.context, "resume").callsFake(() => 'mock');
  return testEnemy;
}

//Testing structure
describe('Enemy', function() {
    it('Enemy should exist', function() {
        assert.
    });
});
```

hooks.js

```
const sinon = require("sinon");

// Restores the default sandbox after every test
exports.mochaHooks = {
    afterEach() {
      sinon.restore();
    },
  };
```

test.js

```
var assert = require('chai').assert;



describe('Array', function() {


  describe('#indexOf()', function() {
    it('should return -1 when the value is not present', function() {
      assert.equal([1, 2, 3].indexOf(4), -1);
    });
  });

});
```

User testing (summative)

Stage 1:

Throughout the whole process, we are involving key stakeholders in. In mid December, we have spread out questionnaire about the theme and main opinions for our game. The detailed documents can be seen in midterm report, or here: https://github.com/FredaXYu/ASP_Group8/tree/main/Questionnaire

Stage 2:

After midterm, we have make a prototype, then implemented a simple design for the story, and created a link that's very easy to open and play on PC with any browser here: https://aspgrp8.z1.web.core.windows.net . So **before Mar. 7**, we have another user test wave. This wave includes many platforms, all targeting at our key stakeholders which are students or university teachers or game companies. In **Slack** chat, since all of our classmates are IT-related or students, they all belongs to our stakeholders. So we just simply created some votes, and then created a feedback form using **Google form**. And in mainland China, Freda also spread another version of questionnaire through **Tencent document,** mainly about the broad theme. We have set age and job questions to quickly identify whether the answerer is our key stakeholder or not.

We have found that, the **key stakeholders** who belong to education realm generally are **positive and supportive** for our main theme (animal defense), and they are very delighted to expect such a product.

Google form feedbacks (users from Slack):

- Colours could have been better and less random.
- It may load slightly longer with low bandwidth.
- The Design of the game is not well integrated with its visuals.
- Little personal freedom.
- When you run the game at a faster rate, the towers don't speed up equally with the enemies. I used an aoe tower on the first corner, and at normal speed it would kill the first wave by itself, but when I sped it up the enemies got through.
- Tower designs are good.
- Enemies (saw & gun) are just fine.
- Most prefer big game rather than small one.
- 1/4 of them feel it is educational, while 3/4 of them are either not sure, or not educated at all.

Tencent document feedbacks (users from mainland China):

- Half of them firmly 'like' the theme (animals defense).
- Almost of them (all of our key stakeholders) accept the cartoon versions of fiece animals to be placed in game.
- They feel the characters and game designs are just so-so.
- Stakeholders give a better score for characters.

Evaluations:

- There is bias when spreading questionnaires. We cannot target our key stakeholder groups very precisely, so we might miss large amount of key stakeholders. For example, we should enter into secondary schools and universities in person, then spread questionnaire to teachers, students, and officials.
- The **sample size is too small** that samples may not be presentive for the population (stakeholder group).
- Since we are limited by time and pandemic, we cannot spread questionnaires easily. Thus, Xiaoyun just spread surveys to people surrounding her (father, uncles, cousin). There may be **bias** since people would not willing to say the truth to hurt our motivations. However, there is a good thing - most of her acquaintances

are education-related.
- We should have previous answerers involved, but it's hard to find them. Still we **have several old users** to provide continuous feedbacks. **Permanent users are**: Xiaoyun's father; Emma; 送快递的狐狸喵, Roberto, Hena(?).

Stage 3:

Starting from Mar 9, we have made some improvements: files are refined; stories are clearly explained; added a treasure box that shows all characteres; added an 'instantly win' page triggered by placing 'the Law' into the map, just to tell people that there is hope for peace. We updated the link for our game here (same URL): https://aspgrp8.z1.web.core.windows.net/

Usability testing

We have listed the items listed in midterm for usability test into **one chart**, for all the versions to check on. Please see the appendix for the chart here.

Accessibility testing

We have listed the items listed in midterm for accessibility test into **one chart**. Please see the appendix for the chart here.

Evaluation

Processes:

ADVANTAGES	DISADVANTAGES
We have allocated tasks to individuals, so that our work won't overlap. Although members are not in full time, we have arranged all of the elements to be done. Members are active and willing to help each other and share any burden.	Our work is not effecient. Members sometimes get vacuum in schedule since our main focus in that period might not be them. We must think about another plan to adjust to the inevitable members' leaves.
We have conducted necessary basic tests.	We should avoid testing our own software. Instead, we should invite another team to test.
We have users to participate in our whole process, so our product emphasizes the weight of users.	There are lack of long-term users that may reflect their opinions step by step along with our development.
We are willing to do any changes based on new ideas or new user requests, and Github is timely for	We have only finished limited amount of tests. We should do more tests, including more types of tests.

everyone to see the changes.	
	We didn't discuss too much about the process of product-launching. Marketing is not our advantages, but we should have a plan or ask professionals.

FUTURE EXPECTATIONS

Once the product is on market, it is not done. Software tests can reveal errors, but it cannot replace the maintenance step to assure the correctness, completeness, and consistency of the software. We should build a **software maintenance schedule** in order to follow up the changes to our product.

Should let a **professional testing team** to handle the testing part, instead of our own, since we might have embedded understanding bias that may cause our product to be easilly understood by us, but hardly understood by users.

We might **recruit long-term user representatives** to be involved in our complete process of development to get more 'user recall', which is a point that Myers emphasizes (Myers, pp. 147).

We must conduct **more types of tests**: Black-box test; White-box test; Function testing; System testing (Facility testing; Volume testing; Stress testing; Performance testing; Storage testing; Configuration testing; Compatibililty testing; Installation testing; Reliability testing; Recoverty testing; Maintenace testing; Documentation testing; Procedure testing); Acceptance testing.

According to Myers:' Examining a program to see if it does not do what it is supposed to do is only half the battle; the other half is seeing whether the program does what **it is not supposed to do** (Myers, pp. 13). ' We must take off redundant parts to make our game straigtforward.

Product:

ADVANTAGES	DISADVANTAGES
Although we are not experts, we learn from zero and finally have built basic functionalities for a typical tower defense game successfully.	There are more to be modified and added to build a complete tower defense game.
We have made our software rather big and well-developed. It is well-classified into folders. It has low module coupling level, and high module cohesion level. It has proper .html and .json files. Files have meaningful names.	Aesthetic design could be better. We should learn from professional designers about the software they use and the templates they refer to. Should make design simple, modern (although we want it to be cartoon) and consistent.
We have successfully implemented Phaser library.	Phaser library is not fully explored. Not all of the members are familiar with this library.
The environment-protection story is adopted brilliantly. Characters are designed well (pictures are	No volume control; no tutorials; no feedback methods; no pop-ups for potential errors; no

found from the Internet).	sharable buttons; no account system; no purchasing method; no protection for teenagers (time control etc.).
The software we wrote is flexible to implement any story or to do any extensions. We have a well-written code documentation here.	Haven't yet found a proper platform to launch.

FUTURE EXPECTATIONS

Should adjust our game to **different screens** (phone, webpage), and can change the size of canvas.

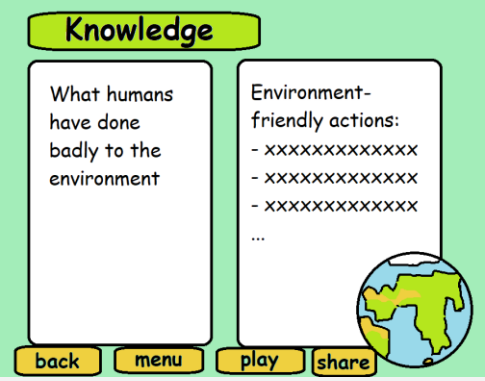
Should fully explore **Phaser features** and adopt them.

We can design towers (with weapons) and enemies **more targeted**, just like in Pokemon, Grass type creatures are afraid of Fire type ones. For example, our Poison Frog can escape from the Hunting Gun, so once put the Poison Frog, it would damage more to the Gun's HP. Perhaps after adding to more and more towers (creatures), our game can **build a universe** just like the Pokemon world (also like the Phylo world).

It is also meaningful once we have discovered how each creature can defense against some certain human beings' weapons. It would be helpful **for ecologists** to understand how the Earth is resilient to human destructions. These trails would also have positive effects to the **education meaning** which we emphasize most.

More to be done can refer to 'User Story Wall' the **'Release 2' and 'Release 3' parts**. To perfect our game, we need to learn from big games (or industries) and build more peripheral functionalities.

We may also design a whole new part called '**knowledge part**', including knowledge for human damaging to the Nature, and what should we do to be environment-friendly. Once we have added this page, we feel it's necessary to add a 'main menu' page which displays all the buttons to all the pages.



We should build more **challenging maps**, explore the best solution, in order to increase the difficulty of our game. Users will find it intellectually enjoyable.

Self-rating based on McCall software quality criteria:

Criteria	Self-marking	Comments
1. Auditability	★★★☆☆	The software itself is easy to build docs, but we should write better documents.
2. Accuracy	★★★★★	API, controls are accurate.
3. Communication Commonality	★★★★☆	We use standards.
4. Completeness	★★☆☆☆	Need to develop.

5. Conciseness	★★★★★	Each file is concise.
6. Consistency	★★★★☆	Requirements, design, codes, tests, management are rather consistent but should be more synchronised.
7. Data Commonality	★★★★☆	We didn't use one thing to represent several concepts.
8. Error Tolerance	★★★☆☆	Should write more exception handling sentences.
9. Execution Efficiency	★★★☆☆	Rather slow when loading images in phones.
10. Expandability	★★★★★	Can expand it based on the prototype.
11. Generality	★★★★☆	Use Phaser library.
12. Hardware Independence	★★★★☆	The link is adaptable in: PC, some phones (with high Internet speed), with different OS.
13. Instrumentation	★★★☆☆	Can hardly point to the place of error.
14. Modularity	★★★★★	Independent modules.
15. Operability	★★★☆☆	Lack of tutorials.
16. Security	★★★☆☆	Once we set Github as private, it will be secure.
17. Self-documentation	★★★☆☆	Not enough comments.
18. Simplicity	★★★★☆	Most codes are understandable.
19. Software System Independence	★★☆☆☆	The link is easy to reach. But the codes are hard to run in Node.
20. Tracebility	★★★★☆	Since files are short, libraries are simple, and we use Github, it's traceable for errors.
21. Training	★★★☆☆	It's easy to open the link. Hard to install Node and run the software using it.

Summary

- We have successfully built basic functionalities of an **educational Tower Defense game** with the **story of jungle animals defense against human invasion**. Our game is portable, cartoon, casual, original, and meaningful. We have implemented a certain level of aesthetical design and an attracting story. There are future tasks to expand peripheral functionalities and to launch on platforms.
- This game is in certain level **educational**, since after our introduction, many users have understood: (1) humans' **bad behaviours are enemies** of Nature; (2) animals need to take much **effort** to protect their home; (3) if we try, humans and animals **can live in harmony**; (4) although animals have done all they could to defense, the most powerful weapon is still **the law** settled by humans ourselves. We realise that **it is difficult for our users to fell it as 'educational' enough**, since most of the feedbacks say they are not educated. Reflect: (1) Perhaps we didn't leave enough time for users to read introduction and treasure box pages; (2) it is possible that users cannot trigger the final 'happy ending' since they don't know how to set the Law; (3) we **should add other information pages to teach people** what to do for an environment-friendly life; (4) make a survey. We still need to do a lot here.
- New programmers in our team have experienced the process of making a video game when doing this

game project. Experienced programmers have given their contributions to the codes.

- All members are involved. We have maximized the productivity of the team by using each **individual**'s advantages (please see individual's reflection). Have built 2 Gantt charts to manage time and tasks.
- Each team member has learned how to manage **time** and deadlines for each sub-task by writting individual task lists.
- Have learned how to **cooperate**. (1) settled down the platforms: Github, Miro, Tencent document, Google document, Typeform, etc. (2) **have adjusted in some extent based on members' leave requests and have lessons learned;** (3) have allocated tasks based on user stories. We should in the future to allocate tasks timely so that all people would have things to do at the same time.
- Everyone uses **version-control**s for our project using Github.
- Some members have done **statistics**. (1) Questionnaires are continuously made to track user opinion changes. (2) spread the questionnaires. (3) user feedback forms, invite users face to face. (4) data analysis using Python Jupyter Notebook and Excel. Expectations: (1) expand sample size; (2) spread out surveys to more people to avoid bias.
- Have explored **users**' psychology. Have **targeted key stakeholders** and gathered their opinions. We should maintain good relationships with them and keep them along with us thoughout the process.
- To understand the basic elements and how they interact, we have written a **requirement specification**, and drawn **UML**. We have made **changes** to them.
- Have realised the requirement specification into **codes**. (1) initial codes were done in mid Dec. (2) **prototype** is fully written in mid Feb. (3) **developed versions** are settled in Mar. See all codes here: https://github.com/matthewsja/asp8_TowerDefense/
- Have done software testings, including technical testing, user testing, usability testing, accessibility testing, etc.
- Have understood the importance of telling a good main story through book reading, user preferences, and paper reading. Have **implemented this story** into prototype in early Mar. Still we should track key stakeholders' opinions.
- Our product is well-written. **It has low module coupling extent, and high module cohesion extent. It is accessible, consistent, concise, usable, robust, understandable, rather safe, effective, expandable, re-usable, maintanable, tracable, and in some extent error-tolerant.** It has lots of **comments.** Besides, we have a good **documentation.**
- Have experienced how an agile project works in team. Have understood what should we do to get users involved. Have experienced the process of responding quickly, **taking all the changing records timely and tidily**. You can see **all** the versions of files and documents here: https://github.com/FredaXYu/ASP_Group8

Individual reflection

Name: Freda Xiaoyun Yu
Tasks I have done:
Research – Literature review; Market analysis; SWOT writing;

Questionnaires making; Questionnaires spreading; Data analysis; User testing documents making; User feedback tracking; From user feedback to user story to epics break down;
Gantt charts making; Mind map making; procedure diagram making;
Main story proposing; Page order proposing;
Assets making – composing background music, drawing pictures;
Documents in Github arrangement;
Accessibility testing; Usability testing
Midterm paper writing (>80%); Final-term paper writing (>90%); Documentation of Codes (>60%).

Freda Xiaoyun Yu's Personal Timetable

Tasks done | Planning period | Crucial things done | Leave for personal issues

Missions Done	Nov. 2021				Dec. 2021				Jan. 2022				Feb. 2022				Mar. 2022			
Gantt chart making																				
SWOT																				
(Miro) discussions																				
Paper research, books download																				
Questionnaires making (western)																				
Questionnaires making (China)																				
Questionnaire spreaded																				
Data analysis by Python																				
Story proposing																				
Answer to Alex's questions																				
User stories																				
Midterm proposal writing																				
All files gathered in Github																				
(Leave for family issues)																				
Compose bg music																				
Paint menu icons, bg pic.																				
Implement picture assets																				
User feedback forms																				
Write 'Treasure box' codes																				
Refine codes																				
User guide																				
Accessibility, usability forms																				
Intro, summary, evaluation, analy.																				
Final proposal writing																				

Team Working Evaluation

I deem myself to be capable of completing a big task, but previously I was used to working all the tasks **individually**. This time I meet such a wonderful opportunity to know the experienced group members and work with them. As I don't have much experience in computer scale, I **should learn from them**.

My Strengths	My Weaknesses
- My teammates mainly have programming skills, but I have the ability to **arrange tasks** from a big scale. Previously when I firstly joined this group, I thougt they would be more experienced than me in task arrangement especially for game project, but after I asked moderately in group, no one responded me. So then I acted as the peripheral-things-manager to **make all the plans and documents**. - They don't have as much ability as I have in **data analysis**. I have done 3 projects in high school, all	- I am a **stubborn** person who will work according to only one route, and I don't like others to change my work. It's lucky for me that teammates are all lenient and they support my ideas, so I don't need to change a lot. It's time for me to be regretful at the beginning that I was not adaptable to this new group. Otherwise I could have learned more from others. - I **misundestood some technical terms** 'Prototype', 'Requirements', etc., which leads to a low score for our midterm proposal. I should **gather teammates to proofread** the final.

using statistics. And I have attended data science courses. Therefore, I am experienced in data handling. - I have the good habit for backing up files very often. Now it's the first time that I **use Github**, and I enjoy using it. - I'm **good at user psychology**. I can use emoji to raise the attention of teammates in group chat. I can phrase a good message in order to collect more users to fill in our survey. My questionnaire questions are not too long so **users can quickly understand** without confusion. - I am active in group, and I am **able to undertake heavy responsibilities**.	- My teammates indeed have a much higher ability in **programming codes** than me. I have learned for the following: -- Search online for **related libraries** and study how to use them. There are specific functions and classes for us to reuse. -- **UML and Specification**. I have known the format. -- From the specification, we should **separate the files to different folders** to form a good hierachy and to maintain a low module coupling level. -- We can even only write one class in one file to maintain a **high module cohesion** level. -- Use **exception handling** codes to avoid vague bugs. -- The whole structure of a game software should include **many types of files**: index.html, index.js, other .js files, a test.js file for debugging, etc. -- I still need to learn **how to build a node.js software,** though now I am fluent in using node to run the codes.

Evaluation of own work referencing original proposal

- I have helped our team to successfully built an educational TD game. (1) Educational: the story of environment-protection is raised by me; I proposed the idea of the happy ending page to teach people to give up weapons; I have read lots and lots of research papers for related realm; I have contacted familiar friends who work in education area for suggestions; (2) Game development: I have no experience, but I can understand the initial prototype codes and then make updates to them; I can use Github; I am the art designer; I wrote the bg music; I can do some of the testings; (3) Data analysis: my skills are important for our team to get user feedbacks timely and continuously.

- Agile project: (1) I should plan more when I leave for personal issues; (2) I have notified members whenever I've written a new file; (3) I should allocate (mid/) final proposal sub-sections to each individual, so my tasks wouldn't be heavy; (4) our project goes rather straightforward without too much changes, but I should be flexible in mind for any sudden change.

Appendix

A. References

1. References for this paper:

Boulton, C.A., Lenton, T.M. & Boers, N. *Pronounced loss of Amazon rainforest resilience since the early 2000s*. Nature Climate Chang. (2022). https://doi.org/10.1038/s41558-022-01287-8

Brich, Julia et al. 'LiverDefense: Using a Tower Defense Game as a Customisable Research Tool'. 2015 7th International Conference on Games and Virtual Worlds for Serious Applications (VS-Games), 2015, pp. 1-8, doi: 10.1109/VS-GAMES.2015.7295779.

Callahan, M.M., Echeverri, A., Ng, D. et al. *Using the Phylo Card Game to advance biodiversity conservation in an era of Pokémon*. Nature Palgrave Commun 5, 79 (2019). https://doi.org/10.1057/s41599-019-0287-9. https://phylogame.org/

Koster, Raph. *A Theory of Fun for Game Design*. Paraglyph press, USA, 2005.

Krotoski, A. *Serious fun with computer games*. Nature 466, 695 (2010). https://doi.org/10.1038/466695a

Miller, Donald. *Building a StoryBrand Clarify Your Message So Customers Will Listen*. HarperCollins Leadership, 2017.

Myers, Glenford J., Corey Sandler, Tom Badgett. *The Art of Software Testing*. 3rd ed. John Wiley & Sons, Inc., New Jersey, 2012.

Peters, J.L., Crewther, S.G., Murphy, M.J. et al. *Action video game training improves text reading accuracy, rate and comprehension in children with dyslexia: a randomized controlled trial*. Nature Sci Rep 11, 18584 (2021). https://doi.org/10.1038/s41598-021-98146-x

Tsai, Meng-Han, Yu-Lien Chang, et al. 'The effectiveness of a flood protection computer game for disaster education'. Springer, 15 March 2015. https://link.springer.com/article/10.1186/s40327-015-0021-7

2. References for our game product:

Picture resources:

https://github.com/FredaXYu/ASP_Group8/blob/main/after_midterm/asset_pictures/ASP_pictures/compressed_icons/picture_resources.txt

Library resources: https://phaserjs.com/

Code resources that helped with turning the ideas into Phaser code are:

1) https://www.phaser.io/examples,
2) https://blog.ourcade.co/posts/2020/organize-phaser-3-code-game-object-factory-methods/,
3) https://photonstorm.github.io/phaser3-docs/Phaser.GameObjects.Image.html,
4) https://stackoverflow.com/questions/54076702/problem-with-changing-the-scenes-in-phaser-3,
5) https://www.youtube.com/watch?v=S1VSKkL_ePM,
6) https://phaser.io/phaser3/devlog/99
7) https://www.phaser.io/news/2018/03/pathfinding-and-phaser-3

B. Accessibility testing

		Feb 19 ver.	Mar 1 ver.	Mar 7 ver.			
Perceivable	Platform accessible	?	?	Y			
	Can load pictures / summary	Y	Y	Phones sometimes are slow; PC ok.			
	Can play sound & music	Y	Y	Y			
	Speed runs normally	Y	Y	Y			
	No dizzy-causing scene	Y	Y	Y			
	Locally legal	?	?	Maybe. It's a link, non-profitable			
Operable	Operations and keys are correctly related	Y	Y	Y			
	Less likely to stuck; show notices	Y	Y	Y			
	Reasonable operations	Y	Y	Y			
Understandable	Understandable main story	?	?	Y			
	Understandable buttons	Y	Y	Y			
	Tutorials for any new thing	?	?	Parts - 'Treasure box'			
Robust	Exception handling codes for potential errors	?	Y	Y			
	Comprehensive control flows	Y	Y	Y			
	Meaningful error messages	?	?	Y			
	Keep running facing errors	?	?	Y			
	Termination routes (return to home / pop up message)	?	?	?			
Types of disability	Vision disability (blind / colour blind)	?					
	Physical disability (difficult to use keyboard / mouse)	? Only need mouse clicking/touching					
	Cognitive disability (poor memory)	?		Need to add info for			

				towers			
	Literacy disability	?		? Future could add text reading			
	Hearing disability	? Music can be shut off					

C. Usability testing

Nielson's 10 usability principles		Feb 19 ver.	Mar 1 ver.	Mar 7 ver.			
1. Visibility of system status	Menu in all pages	Y	Y	Y			
	Users know where they are	Y	Y	Y			
	Home in all pages	?	?	?			
	All levels in 'map'	Y	Y	Y			
	Treasure box	N	N	Y			
2. Match between system and the real world	No abstract characters	?	?	Y			
	Good story	?	?	Y			
	Meaningful main characters	?	?	Y			
	Vivid background	?	?	Y			
	Shapes of buttons / icons	?	?	Y			
3. User control and freedom	Volume control	N	N	N			
	Skin costumization	N	N	N			
	Choice of towers	Y	Y	Y			
	Choice of levels	Y	Y	Y			
4. Consistency and standards	Same word for characters	Y	Y	Y			
	Buttons in the same type	?	?	Y			
	One character one shape	Y	Y	Y			
5. Error prevention	Delete error-prone conditions	Y	Y	Y			
	Show notices before error	?	Maybe	Maybe			
6. Recognition rather than recall	Always show names of tower	Y	Y	Y			
	Name the weapon	Y	Y	Y			
	Enemies look evil	?	?	Y			
	Towers & characters look kind	?	?	Maybe			
	Emphasize the main troop	N	N	N			
	Always show treasure box	N	N	N			
	Always show the menu	Maybe	Maybe	Maybe			
7. Flexibility and efficiency of use	Tutorials for inexperienced	N	N	N			
	No tutorials for experienced	?	?	N			
	Choice of fighting speed	Y	Y	Y			
	Account system	N	N	N			
8. Aesthetic and minimalist design	Intro animation show relevant	N	N	Y			
	Minimize # of characters	Y	Y	Y			
	Minimize # of buttons	Y	Y	Y			
	Background is not too obvious	Y	Y	Maybe			
	Emphasize the main info	N	Maybe	Maybe			
9. Help users to recognise,	Give our email	N	N	N			

diagnose, and recover from errors	Pop-up info for any error with our contact info	N	N				
10. Help and documentation	'Help' button in each page	N	N				
	The help document	N	N				
	Add a search functionality in the help document	N	N				

D. User testing (formative)

D1. Questionnaire and data analysis for settling down the theme, mid Dec.

Full documents please see here: https://github.com/FredaXYu/ASP_Group8/tree/main/Questionnaire
Typeform questionnaire link: https://8xecctbe3tl.typeform.com/to/rjWAd3Qm Response number: 10
Tencent document survey link: https://docs.qq.com/form/page/DYWlZdWlleEZJdlpB Response number: 7

D2. Questionnaires and feedback forms for the prototype with few aesthetic pictures imbedded, Mar 7.

Place: Slack #general group
Deadline date: Mar 7
Response number: 28
Feedbacks:
- 4/28 people don't like the cartoon versions of fierce animals as the main characters at all (1 star).
- **17/28 people are very delighted** to see them.

How do you like this idea: Set the cartoon versions of fierce animals as the main characters (towers) in our Tower Defense Game? eg. tiger  , poison frog  , piranha  , cobra  , the law  ?

1 score	4
2 scores	1
3 scores	4
4 scores	2
5 scores	17

Place: Slack #random group
Deadline date: Mar 7
Response number: 18
Feedbacks:
- 13/18 people like our main theme very much.

We are making a Tower Defense game. How do you like this idea: "Set jungle animals as towers to defense against human invasion to the jungle. " Please score it. Thank you~!

5 scores 13
4 scores 2
3 scores 1
2 scores 1
1 score 1

Place: Google questionnaire
Deadline date: Mar 7
Response numbers: 4
Link: https://forms.gle/UxNpjDTWP9d8qcAT7

Time stamp	3/4/2022 13:38:27	3/4/2022 14:59:32	3/4/2022 16:09:24	3/4/2022 18:31:43
Is there any system errors / bugs / stops / functionality errors for the above link that you open?	The animals don't appear many times. Can't put defense as a result.	The link works well	No, It may load slightly longer with low bandwidth.	No
Our idea is to let jungle animals to defense against human invasion. Please rate this idea.	⭑ ⭑ ⭑ ⭑ ⭑	⭑ ⭑ ⭑ ⭑	⭑ ⭑ ⭑	⭑ ⭑ ⭑ ⭑
Please rate this picture (aesthetic design, conveniency, information clarity, etc.) which is the game screenshot.	⭑ ⭑	⭑ ⭑ ⭑	⭑ ⭑	⭑ ⭑ ⭑ ⭑
About aesthetic design, where should be improved?	Graphics could be improved. Transparent PNGs instead of white bg PNGs would have been better. **Colours** could have been better and less random.	The aesthetics are fine for a small MVP, obviously there could be more polish in a final release version of something.	The Design of the game is not well integrated with its visuals.	

About navigation and information clarity, where should be improved?	Provide instructions in the menu.	There are a few weird things. When you run the game at a faster rate, the towers are less effective. They **don't speed up equally with the enemies**. I used an aoe tower on the first corner, and at normal speed it would kill the first wave by itself, but when I sped it up the enemies got through.	It needs to be integrated well visually, so make use of examples from other tower games eg.buttons and features at the bottom and information at the top. Features should be easily distinquishable and not blend in with the game. Some good examples can be found in mobile.	
Please rate the following main characters (towers).	⭑ ⭑ ⭑ ⭑	⭑ ⭑ ⭑ ⭑	⭑ ⭑ ⭑ ⭑	⭑ ⭑ ⭑ ⭑ ⭑
Please rate the following enemies.	⭑ ⭑ ⭑ ⭑ ⭑	⭑ ⭑ ⭑	⭑ ⭑ ⭑	⭑ ⭑ ⭑ ⭑
About the characters (towers & enemies), how to improve?	They are amazing, keep it up!		Use more relatable characters, e.g. Plants vs zombies has zombies, for your game you would have a lumber jack, a cowboy, dump truck.	
Do you think you are free for personal adjustment in this game?	⭑	⭑ ⭑ ⭑ ⭑	⭑ ⭑ ⭑ ⭑ ⭑	⭑ ⭑ ⭑ ⭑
Do you like portable small game OR well-developed big game? Which one do you willing to play most often?	well-developed big game	portable small game	well-developed big game	well-developed big game

Do you think you are educated and have understood more after playing this game?	No	No	Maybe	Yes
How relaxing and fun is this game?	⭐⭐⭐	⭐⭐⭐	⭐⭐⭐	⭐⭐⭐⭐
In which platform do you want to see our game (if you know any)?	Common mini-game websites	Online	PC	
Is your job related to education (primary / middle school / university students, teachers, etc.)?	**Yes, related.**	No.	No.	**Yes, related.**
Your age?	18-30	30-45	18-30	18-30

Place: Tencent document (mainland China)

Deadline date: Mar 7

Response number: 8

Link: https://docs.qq.com/form/page/DYUFySVJHbVJUZFNC

提交者（自动）	刘军	Lu	送快递的狐狸喵	七柱🏛篝火🔥	白🐾	白🐾friend	liuliang	云中客
提交时间（自动）	##########	##########	##########	##########	##########	##########	##########	##########
喜不喜欢以“丛林、热带雨林、动物”保护作为游戏的题材？🌲🌸🦡 🍀🐯🐸（必填）	喜欢	凑合	凑合	喜欢	喜欢	凑合	喜欢	喜欢
主题：使用雨林中的动物作为塔防，设置好它们的位置，让它们发挥本领以阻止人类入	喜欢	凑合还行	不知道，选不出来	喜欢	凑合还行	凑合还行	喜欢	喜欢

侵雨林大本营。 您喜欢这个主题吗？（必填）								
喜不喜欢把卡通化的猛兽作为主角？（主角形象见下一题的图片）（必填）	喜欢，因为它们有力量	喜欢，因为它们有力量	喜欢，因为它们有力量	喜欢，因为它们有力量	喜欢，因为它们有力量	喜欢，因为它们有力量	喜欢，因为它们有力量	喜欢，因为它们有力量
给下列角色打分。（5 分是最喜欢，1 分是最不喜欢）（必填）	4 分	3 分	3 分	3 分	4 分	3 分	4 分	4 分
给下列游戏界面打分。（5 分是最喜欢，1 分是最不喜欢）（必填）	4 分	4 分	3 分	4 分	3 分	3 分	3 分	4 分
您的性别？（必填）	男	女	男	女	女	女	男	男
您的年龄层？（必填）	45-65	18-30	18-30	45-65	18-30	18-30	45-65	45-65
您的职业与教育有关吗？例：中小学生、大学生、正在读书、教师、学校工作者（必填）	无关	**有关**	无关	无关	无关	无关	**有关**	**有关**

User test forms collected:

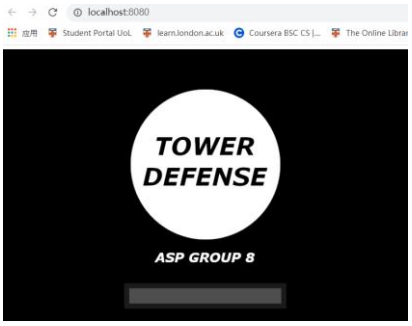
Name	Freda Xiaoyun Yu	Gender	female	Age	27
Job	Computer Science student	Date	Feb. 19, 2022	Version	Alex early Feb.

Have seen this game before?	N	Is one of the developers before?	N
Platform working fine?	Difficult	Whole speed runs normally?	Y
Operations and keys are correctly related?	Y	Main story is understandable?	?
Button are understandable?	Y	If you have disability, how do you feel about our game?	-
Do you think you are always clear with the help of navigation buttons ?	Y	Volume control / skin customization?	N
Is it fluent for choice of towers or levels?	Y	Do you think buttons are in consistent style?	Y
Do you think characters are in consistent style?	?	You don't need to remember any names. Is it true?	Y
See tutorials?	N	See our contact info?	N
Help document?	N		

Likes

- Wow! The music is melodic. It's continuous, without too much fluctuation, so that it allows me to concentrate on the game and how I should operate.

- I like the icon in the first page.



- It's a surprise that the bullets work well!

- I can see the enemies coming in.

- It's good to see there is a menu. It's handy.

Dislikes

How to Open The Game

Firstly I downloaded the zip file from our Github page. Then unzip it.

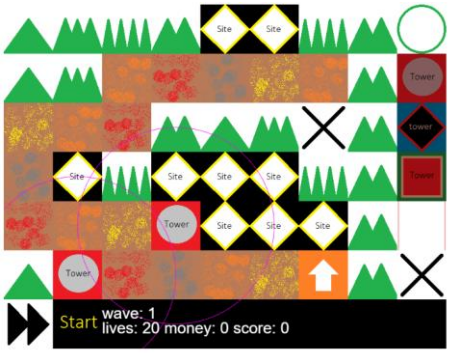
1. I tried to open it with Brackets, but it cannot show the live window.

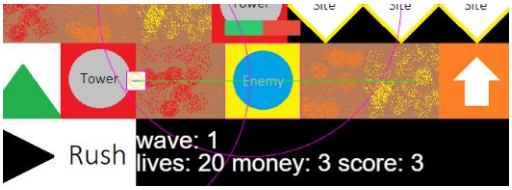
2. I tried to open the folder with VSCode, then right click the 'index.html' file -> 'Open With Live Server'. It cannot show anything in the browser.

Then I asked Alex, and he told me I should install Node.js, and type 'npm install' in the unzipped folder's route. Then type 'npm start'. I spent a while to install these things. As a developer, I know it's my responsibility to learn how to operate it with Node.js, but as a user, I would expect a more convenient method to open the game. I would expect that, once I open some web page, it will show a link. I click on the link, then it will pop up a window for the game in the browser.

- I cannot identify where the enemies will come. Well, finally I noticed that I should click on the upper left button to let the enemies in. There should some tutorials.

- I do not know where the menu is. It's just an empty circle.
- I'm not quite sure where I can put towers – places other than the roads and the mountains?

Name	Freda Xiaoyun Yu	Gender	female	Age	27
Job	Computer Science student	Date	Feb. 26	Version	Alex Chu Feb. 19 version
Have seen this game before?		Y	Is one of the developers before?		N
Platform working fine?		Difficulty	Whole speed runs normally?		Y
Operations and keys are correctly related?		Y	Main story is understandable?		?
Button are understandable?		Y	If you have disability, how do you feel about our game?		-
Do you think you are always clear with the help of navigation buttons ?		N	Volume control / skin customization?		N
Is it fluent for choice of towers or levels?		Y	Do you think buttons are in consistent style?		Y
Do you think characters are in consistent style?		?	You don't need to remember any names. Is it true?		Y
See tutorials?		N	See our contact info?		N
Help document?		N			
Likes					
- I can see where to call the enemies now!					
- The lower left button can also arrange the enemies' speed, and the icon is meaningful that everyone can understand. 					
- There is a level completion page. 					
- I think the basic functionalities are complete!					

- Now all of the words and the pictures don't overlap each other.
- It's brilliant to see the life Hit-Point bar. 
Dislikes
- The 'Site' picture should be more meaningful that everyone can know that there is an empty position and towers can be put on there. Perhaps there needs a tutorial.
- The level-completion page should award more satisfaction to the game player. Should change the background.
- There are two 'menu' buttons (the 'circle' and the 'menu'). Should change the first to 'menu', and the second to the 'main panel' or something similar.
- the 'site' should be more meaningful to users that they can put towers there. Change the picture. 
- Shoud add meaningful characters.

D3. Questionnaires and feedback forms for the prototype with aesthetic pictures imbedded, Mar 10.

User feedback forms collected:

Name	jun liu	Gender		Age	55
Job	it engineer	Date	2022 – 03 – 09	Version	Mar. 7
Have seen this game before?		no	Is one of the developers before?		no
Platform working fine?		yes	Whole speed runs normally?		yes
Operations and keys are correctly related?		yes	Main story is understandable?		yes
Button are understandable?		yes	If you have disability, how do you feel about our game?		

Do you think you are always clear with the help of navigation buttons ?	no	Volume control / skin customization?	yes
Is it fluent for choice of towers or levels?	almost	Do you think buttons are in consistent style?	yes
Do you think characters are in consistent style?	almost	You don't need to remember any names. Is it true?	no
See tutorials?	no	See our contact info?	no
Help document?	no	Do you think you are educated after playing this game?	not exactly
Likes			
character style			
Dislikes			

Name	Emma	Gender	female	Age	25
Job	finance	Date	22.03.09	Version	Mar. 7
Have seen this game before?		no	Is one of the developers before?		yes
Platform working fine?		yes	Whole speed runs normally?		yes
Operations and keys are correctly related?		yes	Main story is understandable?		yes
Button are understandable?		no	If you have disability, how do you feel about our game?		normal
Do you think you are always clear with the help of navigation buttons ?		no	Volume control / skin customization?		yes
Is it fluent for choice of towers or levels?		yes	Do you think buttons are in consistent style?		yes
Do you think characters are in consistent style?		yes	You don't need to remember any names. Is it true?		yes
See tutorials?		no	See our contact info?		yes
Help document?		yes	Do you think you are educated after playing this game?		no
Likes					
Dislikes					

Google form feedbacks, shared in Slack:

Deadline date: Mar 11

Response number: 3

Link: https://forms.gle/n4hvKYXt9WXsVZ8i7

timestamp	3/9/2022 15:09:35	3/9/2022 16:12:58	3/10/2022 11:19:55
Name (or nickname)	mihail	Roberto	Hena
Gender	male	male	female
Age (can provide a range)	38	22	20
Job	freelancer	3D Designer & Programmer	Researcher
Date of filling this form	2022/3/9	2022/9/3	2022/3/1
Have seen this game before?	No	Yes	Yes
Platform working fine?	Yes	Yes	Yes
Whole speed runs normally?	Yes	Yes	Yes
Operations and keys are correctly related?	Yes	Yes	Yes
Main story is understandable?	Yes	Yes	Yes
Button are understandable?	Yes	Yes	Yes
If you have disability, how do you feel about our game?		N/a	
Do you think you are always clear with the help of navigation buttons ?	Yes	No	Yes
Volume control / skin customization?	Yes	No	Yes
Is it fluent for choice of towers or levels?	Yes	No	Yes
Do you think buttons are in consistent style?	Yes	No	Yes
Do you think characters are in consistent style?	Yes	No	Yes
You don't need to remember any names. Is it true?	Yes	Yes	Yes
Do you think you are educated after playing this game?	No	No	Yes
How relaxing and fun is this game?	⭑ ⭑ ⭑ ⭑	⭑ ⭑ ⭑	⭑ ⭑ ⭑ ⭑

Likes:	images are funny	I like some visual graphics more than the previous version, The game is more easy to navigate **than the previous version**.	Bravo! **You took into account all the feedback** and created an amazingly improved version of your adorable game. **I especially like the message you provide at the end which says that humans are not our enemies but it is just there actions that are sometimes wrong.** Love it! I like how there is a little introduction to the game now too, introducing the characters. Graphics are a lot better too. Basically, you improved on everything I found not so good in the last version, kudos! All the best!
Dislikes:	it would be great if the levels would have **different layouts**, there was some sort of **animation** for the towers and animies, and it seemed to me that **animies moved too fast**. I would decrease their speed and increase their health.	The features and functionalities of the game **don't look structured** and don't contain more content e.g.Tutorial Screen with backstory and how to play with controls, Varied Levels, Level descriptions or layouts,Game Settings, Skin customization or animal skin unlocks in main menu etc.	I feel like there still a **lack of tutorial** on how to play the game. I had to figure out that you have to click the plus sign and select the animal etc. I wish it was added too. But otherwise, it is perfect!

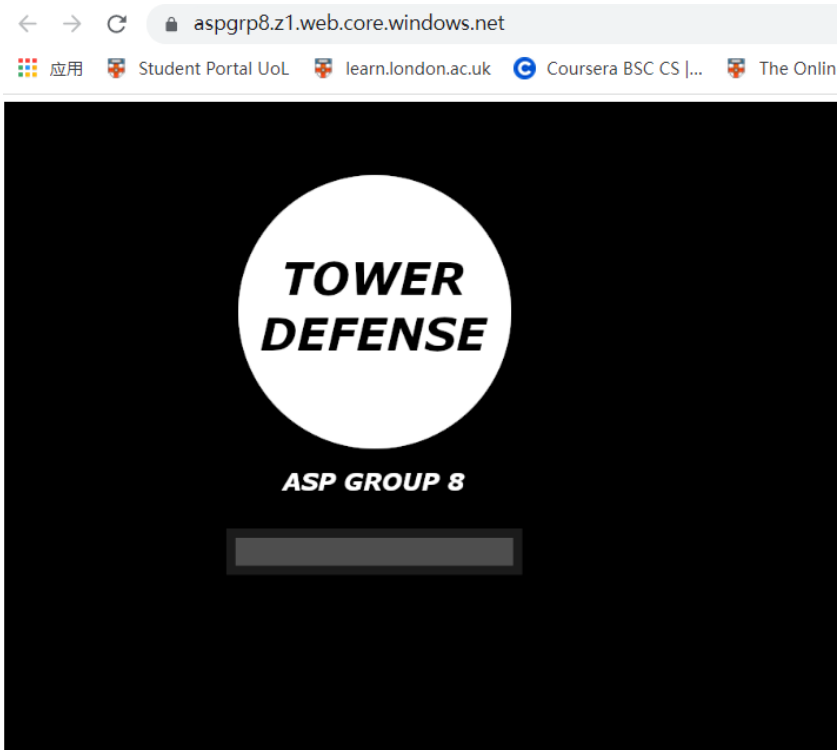
E. User guide for our software

Game name	Save the Jungle
Link	https://aspgrp8.z1.web.core.windows.net/
Feedback address	shireenyu@163.com,
Release date	Mar. 2022
Type	Tower Defense Game
Hardware needed	Any PC with any operating system/ phones with high Internet speed.
Software needed	Must install a web browser.

Hi! A warm welcome for you to play our game! As developers, we would like to guide you though. Have fun!

Firstly, you need to **paste this link to the browser**: https://aspgrp8.z1.web.core.windows.net/

Wait for a while, it will show you our logo. If there are nothing shown in the center, then please wait a minute or refresh the page. You will immediately hear the background music now. The music is composed by us.



After some seconds, it will automatically direct you to the below page with a forest background. Please read the introduction carefully, since it's the settings for our game.

Click on 

button at the bottom of this page, let's get started!

You will see the treasure box displaying all of the characters you will meet.

In the center, there are two types of characters: towers, and enemies. Your task, is to use money to buy towers then settle them onto the map, to defense against enemy waves. Of course, you will play in the later pages.

There are several different **towers: Poison Frog, Tiger, Piranha, Elephant, and the Law**. There are two lines of them. Yes! The first line and the second line are the same species, but the second line is the evolved version. The evolved versions are more costive than basic versions.

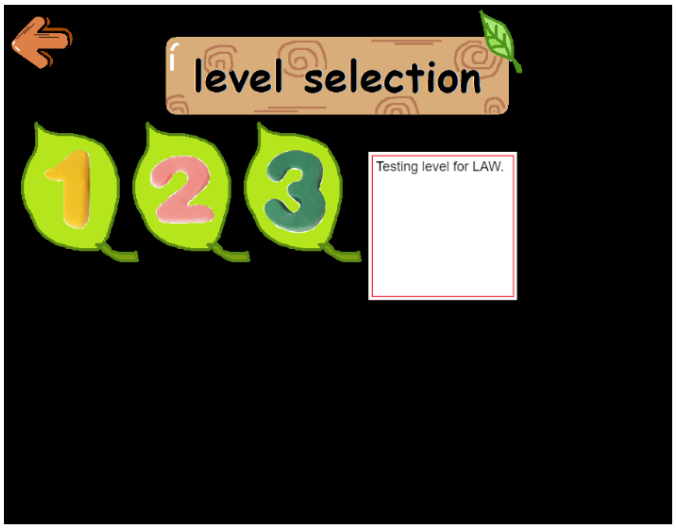
Let's meet the enemies! In the above image, enemies are shown at the bottom. You will firstly see **Plastic Garbage**, which is the most common damage humans do to the Nature. Then there is the **Saw.** Lumberjacks use electric saws to cut down trees. This is one of the main reasons for forest loss. Then there is the **Hunting**

Gun. Illegal hunting has deprived many lives of endangered species. Sometimes, over-hunting will do harm to local biodiversity.

(If you want to read the introduction story again, press  button at the upper left corner to go back.)

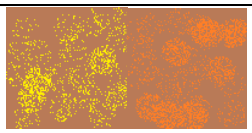
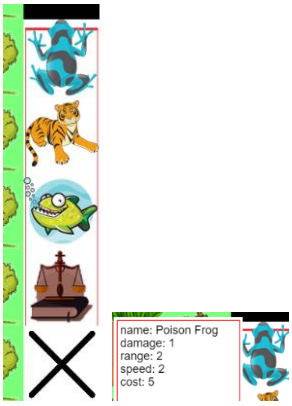
Alright, now that you are familiar with these characters, click  at the bottom.

This is the Level Selection page. You can **click on the '1' '2' '3' leaf-shaped buttons**  to start the game, they represent for three different levels. If you want to know more about each level, hover your mouse at each leaf, then you will see a description.



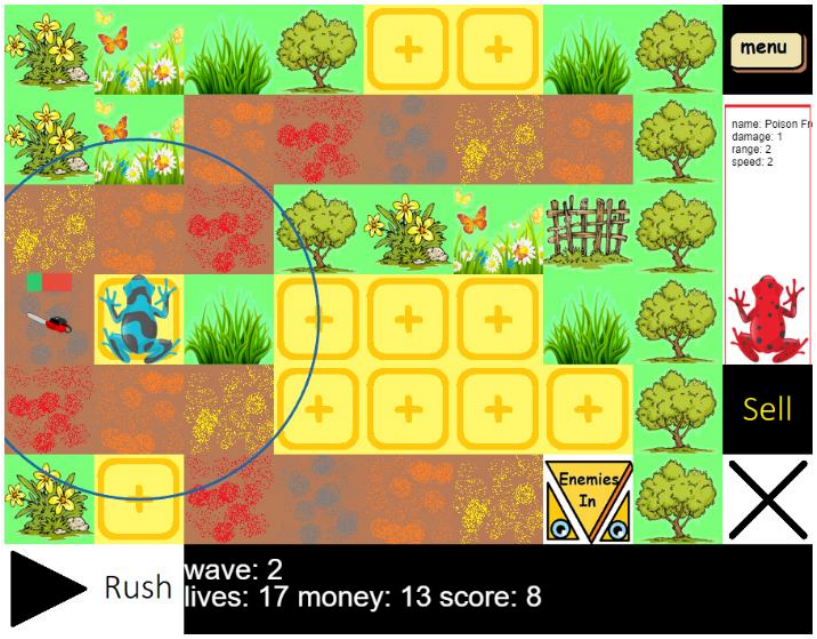
Below is the game playing page. Apart from the right and bottom sides (buttons and game statistics), the colourful main part is what you should notice.

Brown zigzag bricks are the roads. This is where the enemies will follow.

	This image shows the place where enemies will come in.		Brown zigzag bricks forms the path. This is where the enemies will follow.
	These are grass and trees where you cannot pur towers to.		This picture means you cannot put towers here.
	This is where you can put towers. Click on this yellow brick, then you will see towers listed on the right hand side.		Click the yellow button to see this list displayed on the right. The colourful characters are those you can afford to; the shaded ones are those you cannot afford to. You can hover your mouse on each of them to see its descriptions. Click the black 'X' to close this list.
	Already set your towers? Click on this triangle buttonat bottom left to adjust the speed of fighting.  means double speed.  means triple speed.		Speed settled down? Let's start the fighting! Click the 'start' button at bottom left.

Enemies are coming in. You can still click on the triangle button at the bottom left to adjust the fighting speed. 'Rush' means it is rushing, it is not clickable.

Enemies are marked with Hit-Point bar at the top. If this bar is all red, then it will die.



When enemies are coming in, you still can change towers. Click on the tower, then it will show its **upgraded version** on the right-hand side. If you want to delete this tower, click **'Sell'** to exchange for money.

You may have noticed the big blue circle around the Poison Frog. It is its attacking range.

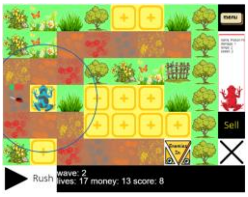
Keep on noticing the game statistics on the bottom. There are second waves of enemies in the 1st level. After all enemies of the first wave are killed, there will show a time counting down. Be prepared.

Then, the second wave of enemies is coming!

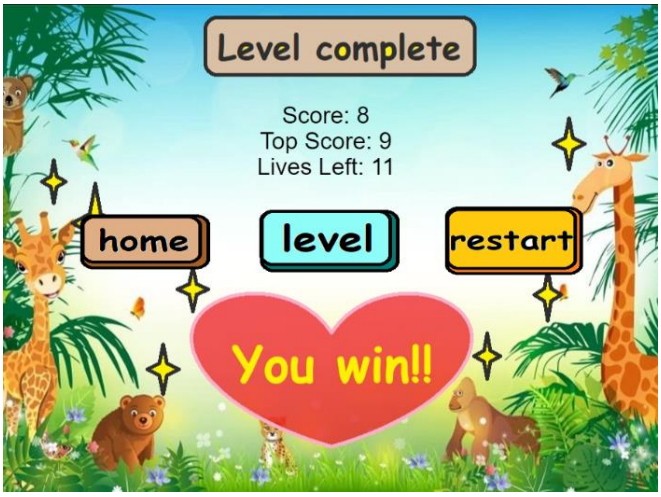
If your lives are counting to 0, then game over. Click on one of the three buttons to get you restarted.

home will go to: 

level will go to: 

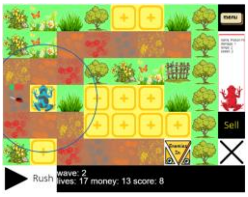
restart will go to: 

If your lives are still a positive number after all enemies have stridden (or you have killed them), then congratulations, you've passed this level! The following page will appear:



home will go to: 

level will go to: 

restart will go to: 

There is indeed a **hidden bonus**. Try to play Level 3 and see what will happen! This time enemies are dense. What should you do?

Hint: Try the Law  .

Once you clicked the Law, a happy ending will appear:

home will go to:

level will go to:

restart will go to:

The Law is the utmost weapon for human bad behaviours. Once we have settled the Law, humans will give up their attacking, and embrace the Nature. Humans finally realised that, we humans and animals are friends from the very beginning. **We are not here to fight, but to embrace**.

Thank you take time to listen to our story.

To protect the enviroment, this is what we want to convey to you. Keep on doing the good. Cheers!

F. Adjusted Requirements

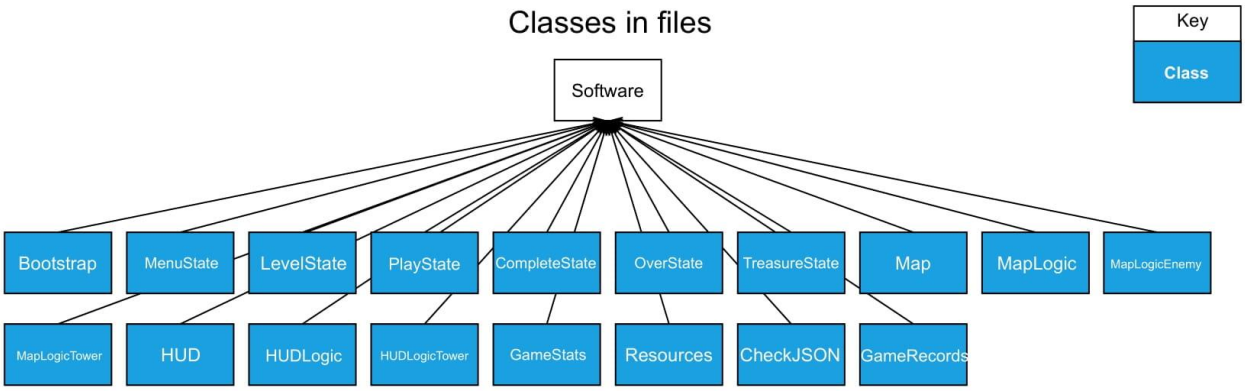
Full document:
https://github.com/FredaXYu/ASP_Group8/blob/main/after_midterm/finalterm_proposal/Requirements_Table_updated_Alex.doc **This is a 57-paged large file which is very crucial – all the system development is built based on it. You may also find it in the files uploaded with this proposal.**

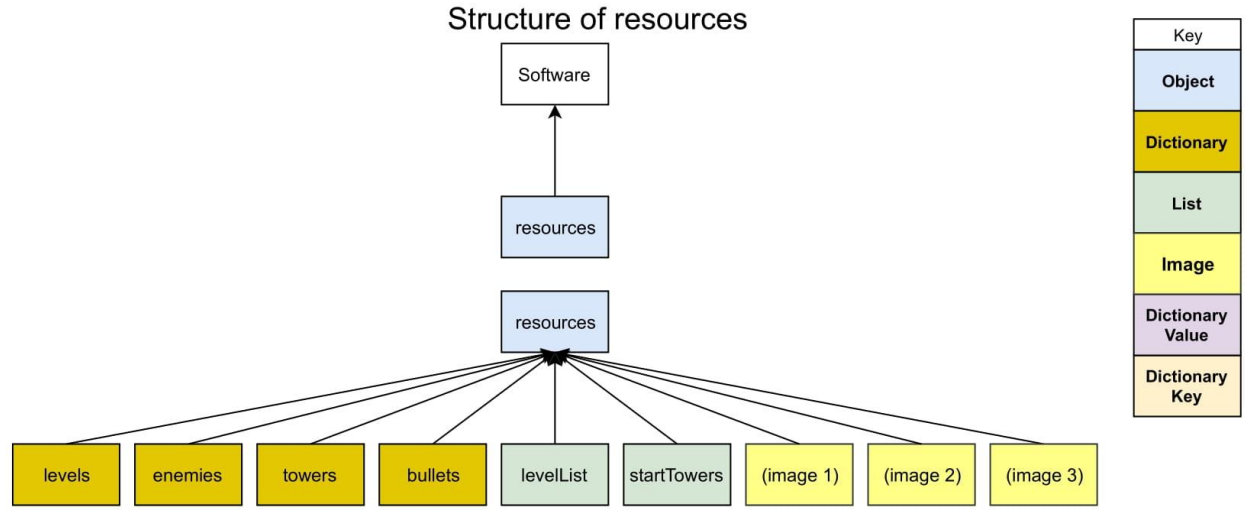
Snapshot:

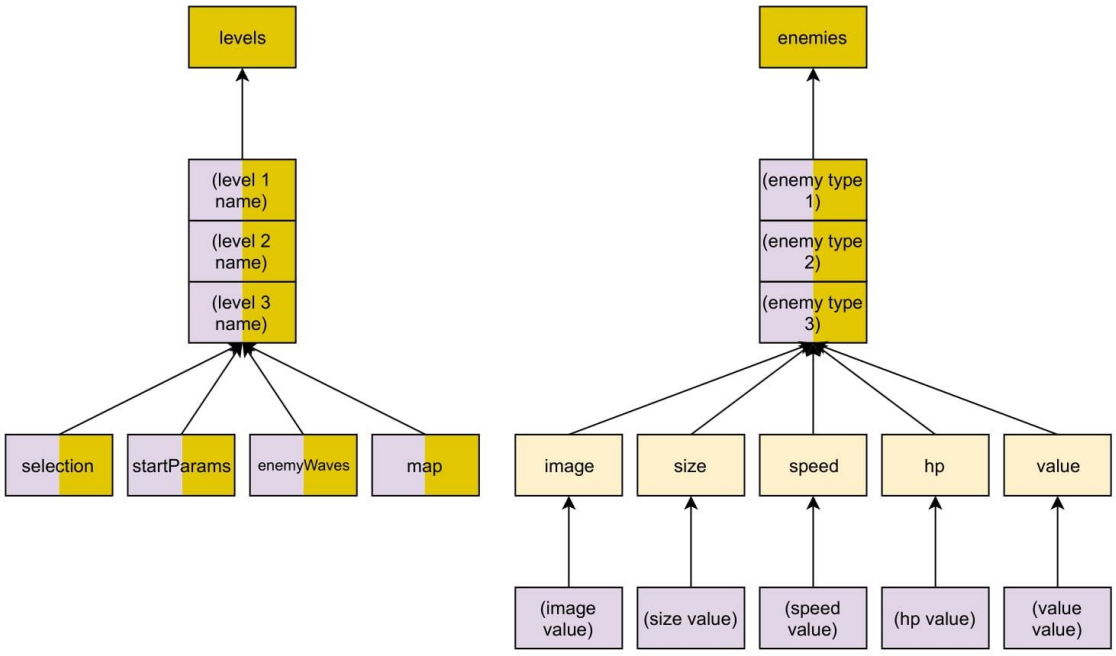
0: Requirements for when the Software starts

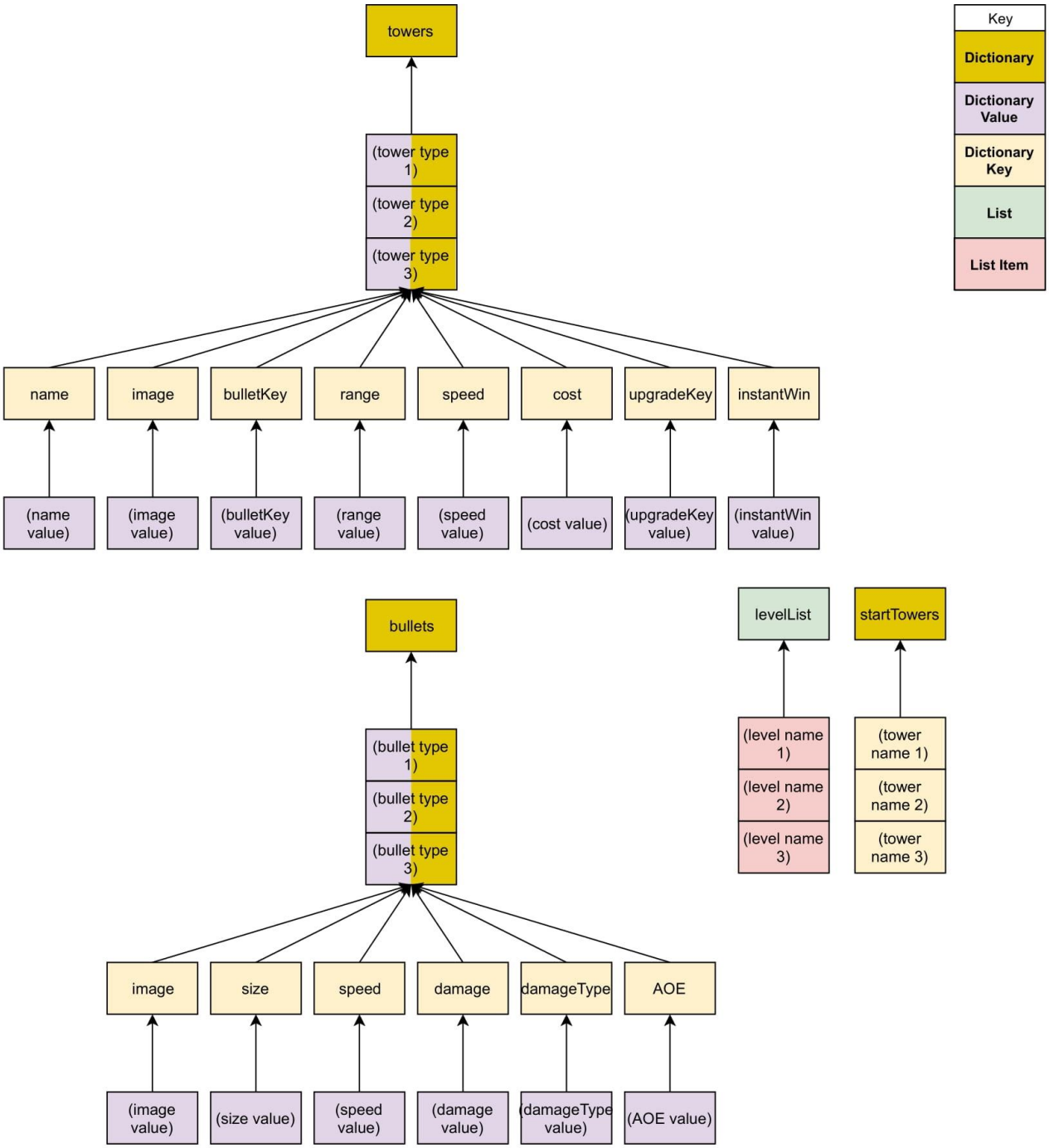
ID	Requirement	Howit was completed
0.0: Resource class		
0.0.0	The software shall have a ResourceObj class	This class is now called Resources and is a Phaser scene class ofresources.js.
0.1: Gameplay classes		
0.1.0	The software shall have a MapObj class	This class is now called Map and is a Phaser scene class ofmap.js.
0.1.1	The software shall have a GameStatsObj class	This class is now called GameStatsand is a Phaser scene class ofthe file gameStats.js.
0.1.2	The software shall have a HUDObj class	This class is now called HUDand is a Phaser scene class ofhud.js.
0.2: Map classes		
0.2.0	The software shall have a OriginObj class	There is no longer anOriginObj class. The origin object is made through the makePath() method ofmapLogic.js.

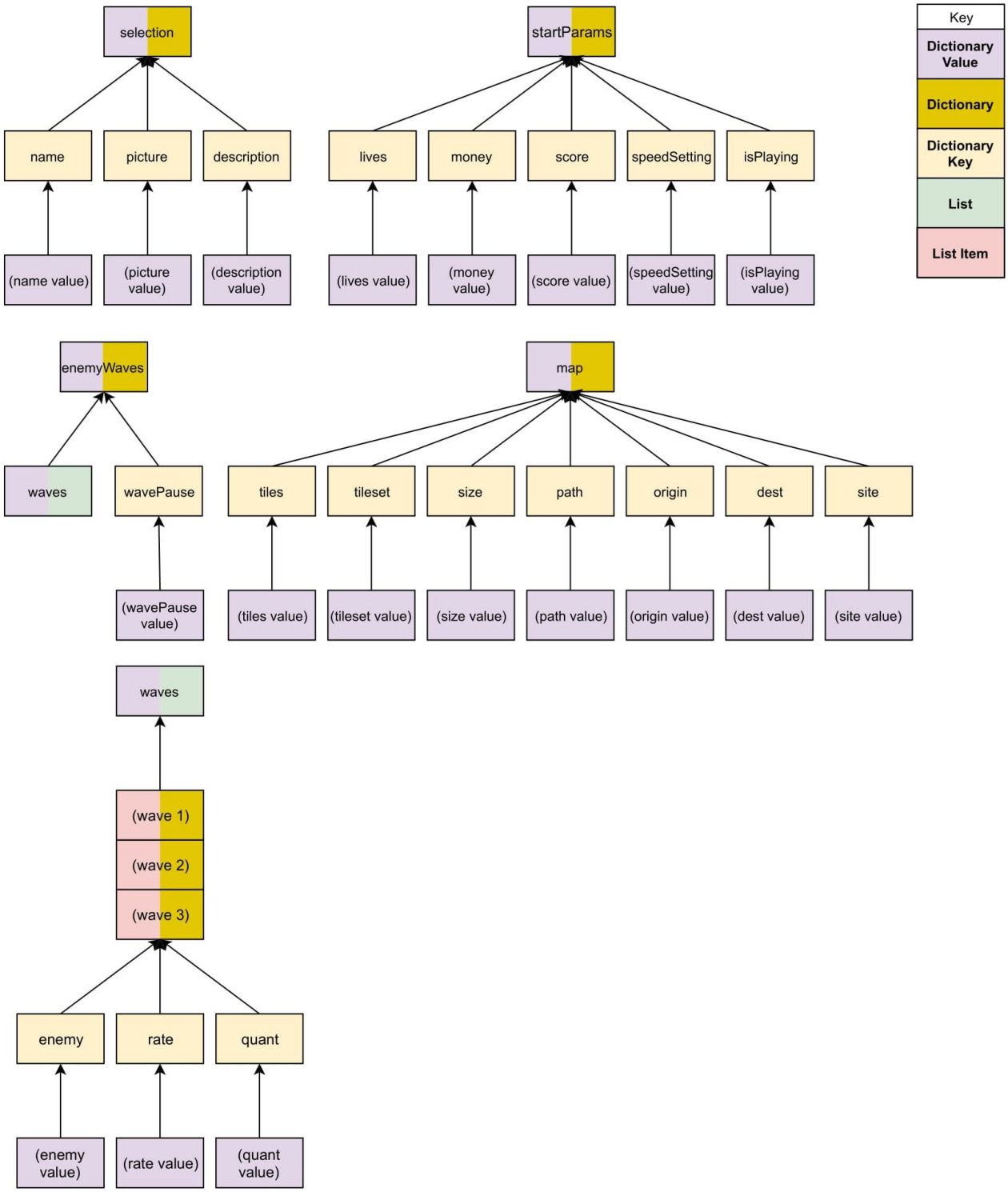
G. Adjusted Components



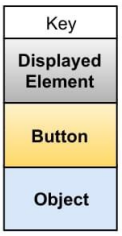




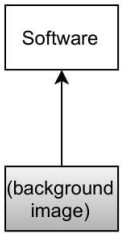

towers
(tower type 1)
(tower type 2)
(tower type 3)
name
image
bulletKey
range
speed
cost
upgradeKey
instantWin
(name value)
(image value)
(bulletKey value)
(range value)
(speed value)
(cost value)
(upgradeKey value)
(instantWin value)
Key
Dictionary
Dictionary Value
Dictionary Key
List
List Item
bullets
(bullet type 1)
(bullet type 2)
(bullet type 3)
image
size
speed
damage
damageType
AOE
(image value)
(size value)
(speed value)
(damage value)
(damageType value)
(AOE value)
levelList
(level name 1)
(level name 2)
(level name 3)
startTowers
(tower name 1)
(tower name 2)
(tower name 3)


selection
name
picture
description
(name value)
(picture value)
(description value)
startParams
lives
money
score
speedSetting
isPlaying
(lives value)
(money value)
(score value)
(speedSetting value)
(isPlaying value)
Key
Dictionary Value
Dictionary
Dictionary Key
List
List Item
enemyWaves
waves
wavePause
(wavePause value)
map
tiles
tileset
size
path
origin
dest
site
(tiles value)
(tileset value)
(size value)
(path value)
(origin value)
(dest value)
(site value)
waves
(wave 1)
(wave 2)
(wave 3)
enemy
rate
quant
(enemy value)
(rate value)
(quant value)

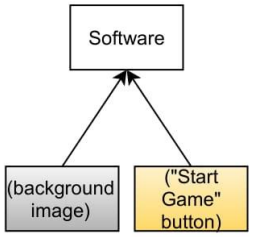
Software When at Different Scenes



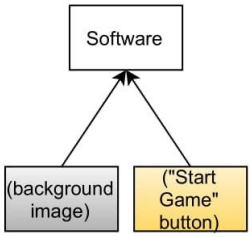
Scene = bootstrapState



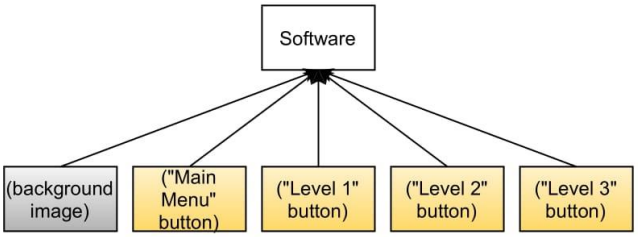
Scene = menuState



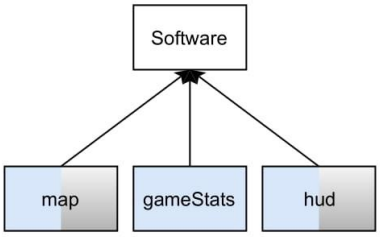
Scene = treasureState



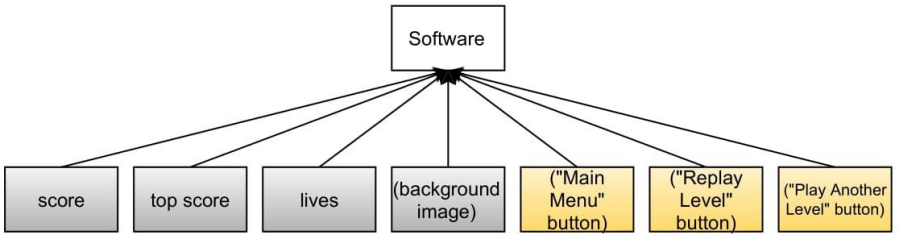
Scene = levelState



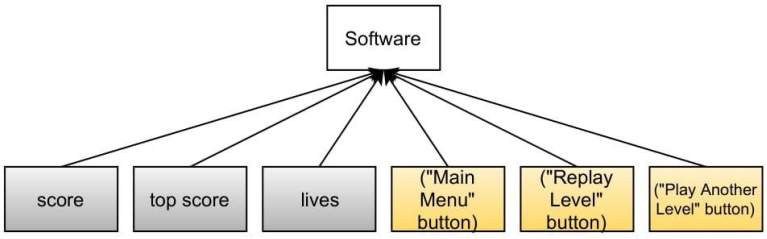
Scene = playingState



Scene = completeState

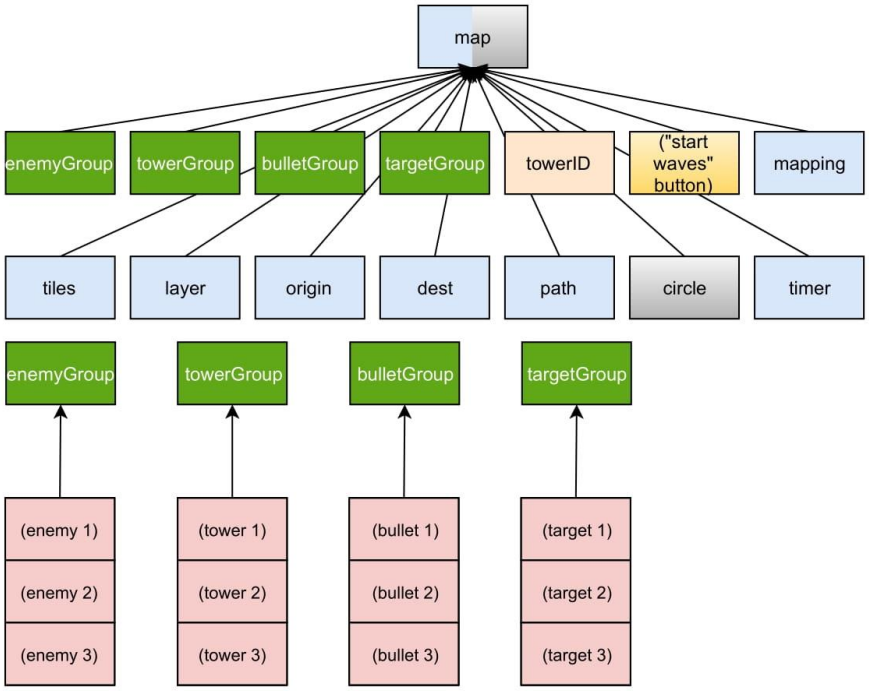


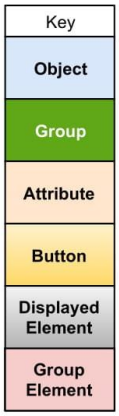
Scene = overState



Note: These do not include constantly running objects like resources.

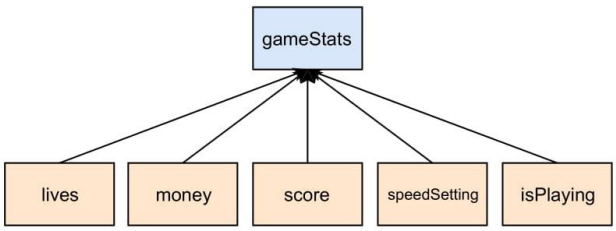
Structure of map

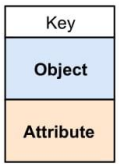




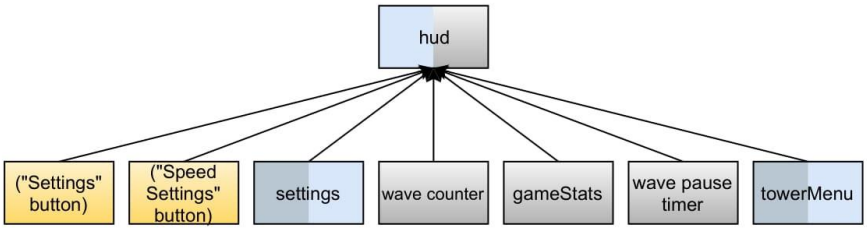
Note: The category "Group" refers to Phaser group object.

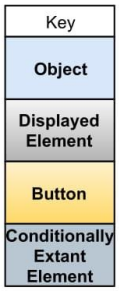
Structure of GameStats



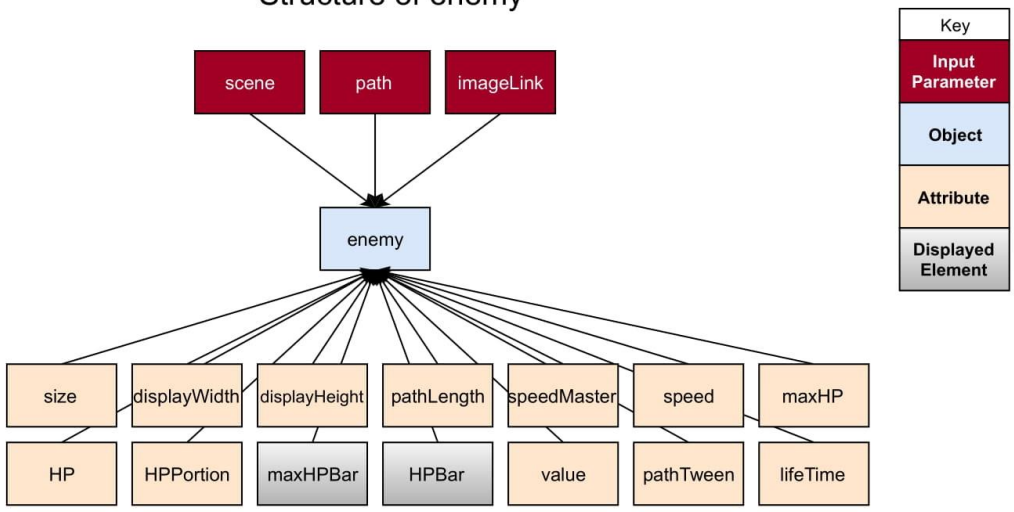


Structure of hud

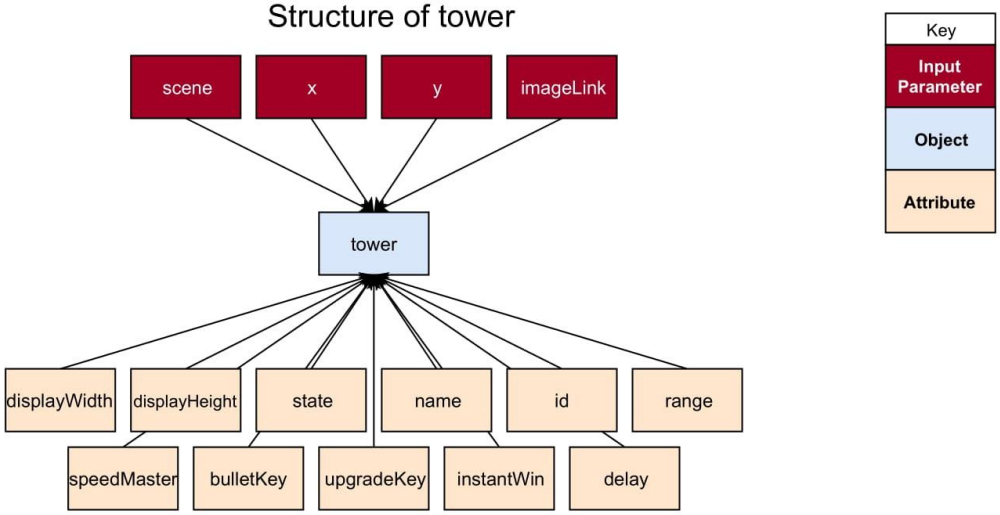




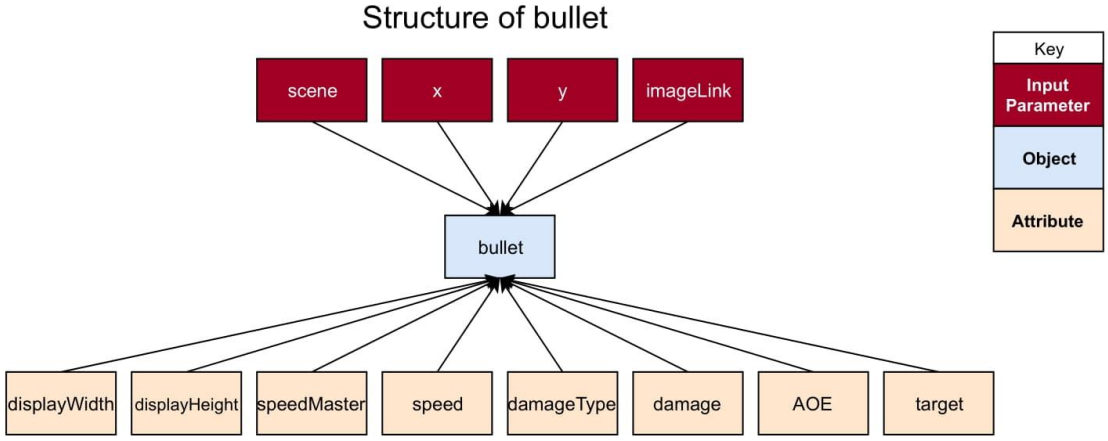
Structure of enemy



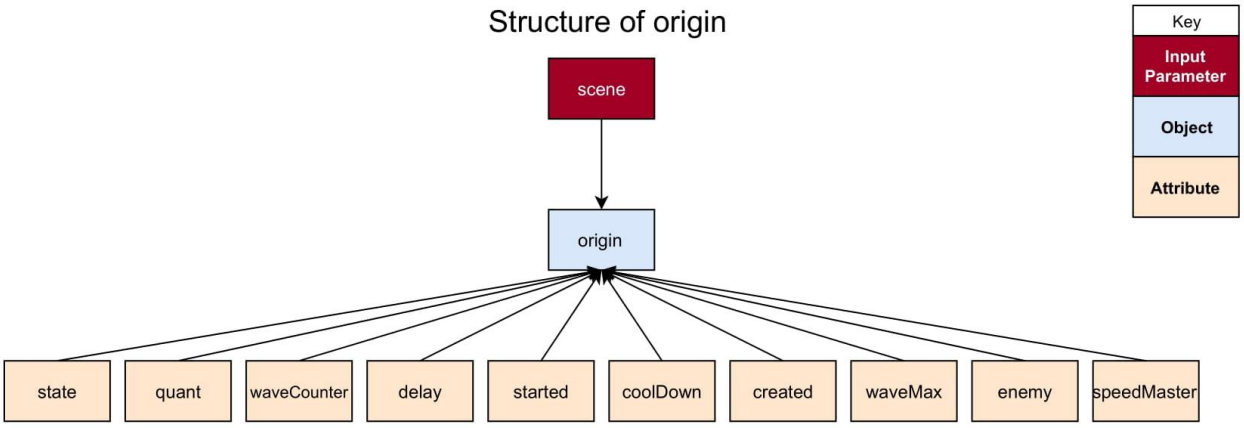
Structure of tower



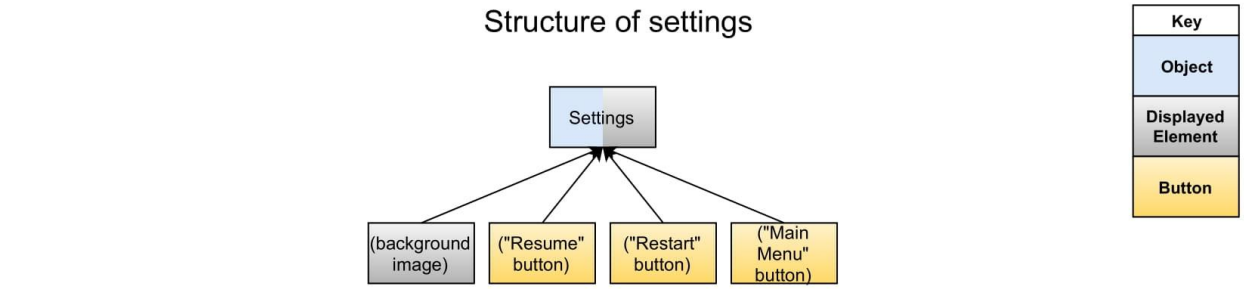
Structure of bullet



Structure of origin



Structure of settings



Structure of towerMenu

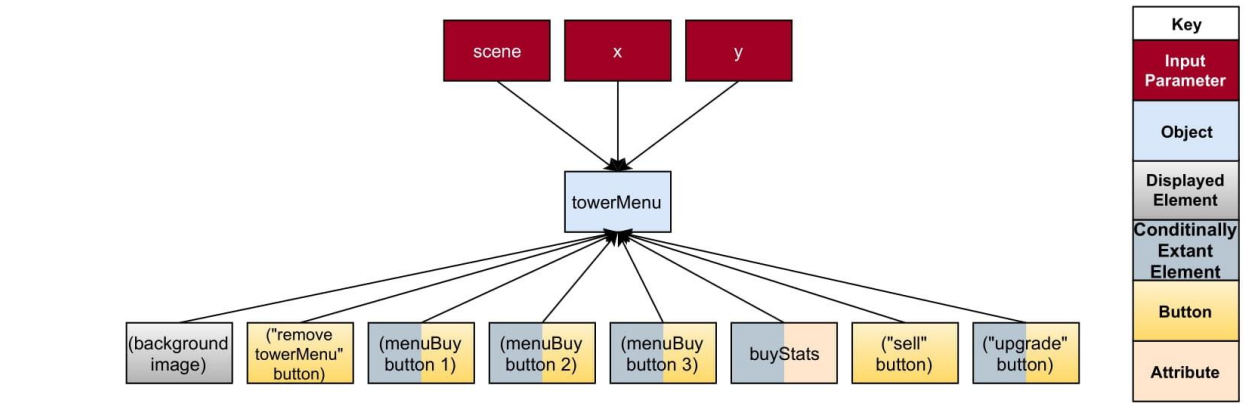
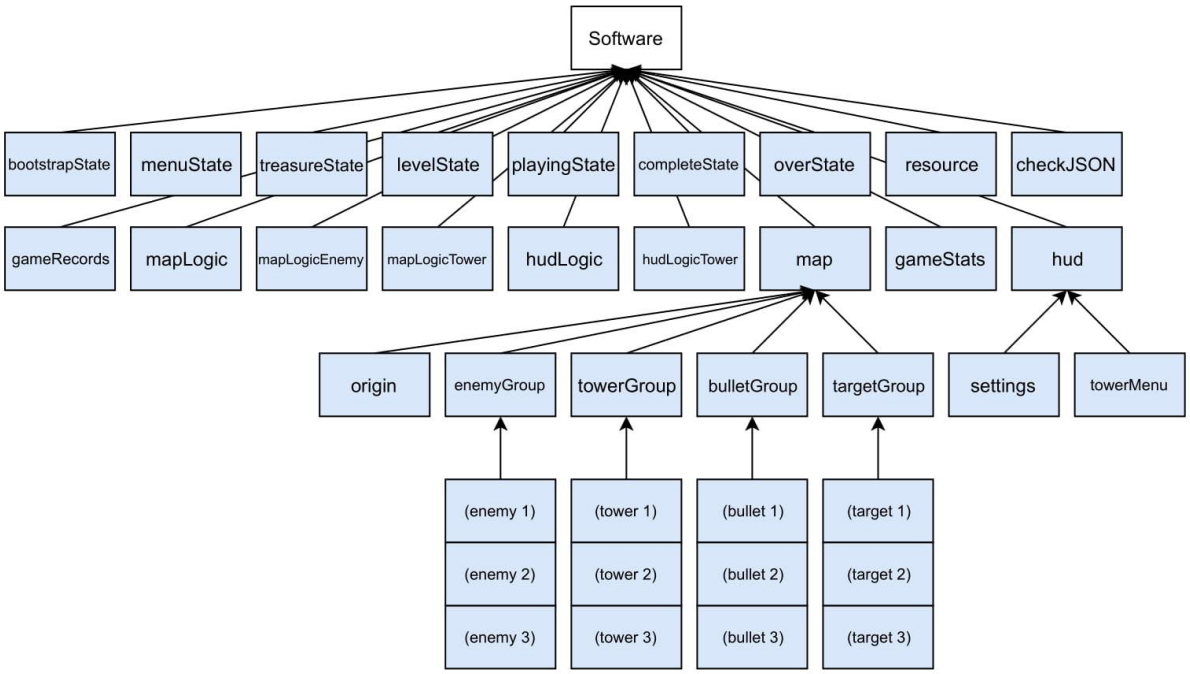
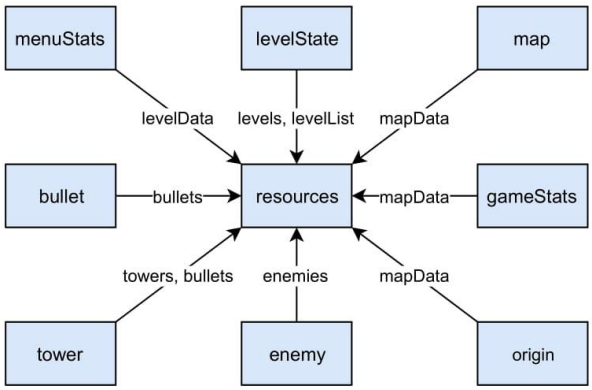


Diagram Of Where Different Objects Would Be Found In The Software



*Note: With Phaser, everything,including images are technically objects, this is just shows objects significantly impacted through the code. Not all of of these objects would exist at the same time. This just demonstrates where these objects would be found when they do exist.

Diagram Of Objects That Read Data From Resource



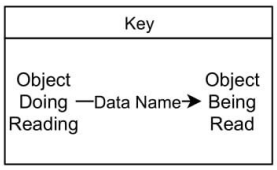


Diagram Of Objects That Read Data From gameStats

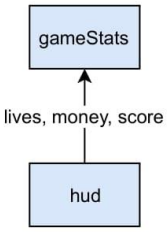
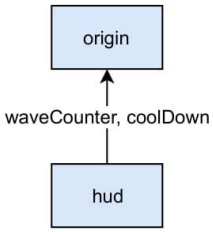


Diagram Of Objects That Read Data From origin



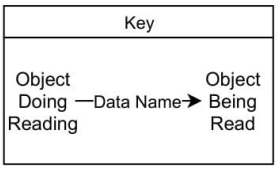


Diagram Of Objects That Read Data From enemy

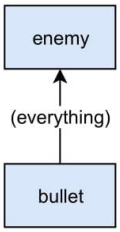


Diagram Of Objects That Read Data From tower

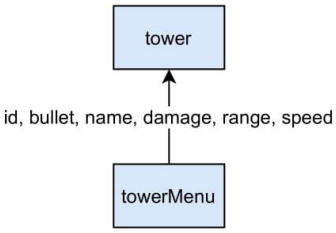
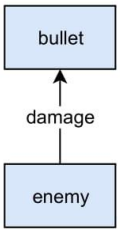


Diagram Of Objects That Read Data From bullet



H. Documentation of Codes

audioManager

audioManager.mjs

The audio manager is used to manage the audio of the game. The music, SFX and so on.

Parameters

- *_soundManager* Phaser's default soundManager for the audioManager to manage.
- *_sfxEvents* A list of sfx 'events' to be called by the *playEffect()* function.
- *_musicEvents* A list of music 'events' to be called by the *playMusic()* function.
- *_musicSounds* A list of music 'sounds' to be called by the *playMusic()* function and looped.
- *_volume* (default: 1) The audible volume of all sounds.
- *_musicVolume* (default: 1) The audible volume of all music.
- *_sfxVolume* (default: 1) The audible volume of all sfx.
- *_currentMusic* (default: undefined) Music to be playing currently.

Functions

pause()

Pauses all sounds being played by the audioManager.

resume()

Resumes all sounds being played by the audioManager.

stopAll()

Stops all sounds being played by the audioManager.

*playMusic(*musicEvent*)*

Plays the *musicEvent*, adding it to the list *_musicEvents* if it is not already in it.

stopMusic()

Stops music from playing.

*playEffect(*sfxEvent*)*

Plays the *sfxEvent*, adding it to the list *_sfxEvents* if it is not already in it.

getVolume()

Returns the value of *_volume*.

*setVolume(*volume*)*

Sets the value of *_volume*.

getMusicVolume()

Returns the value of *_musicVolume*.

*setMusicVolume(*musicVolume*)*

Sets the value of *_musicVolume*.

getSfxVolume()

Returns the value of *_sfxVolume*.

*setSfxVolume(*sfxVolume*)*

Returns the value of *_sfxVolume*.

_getInstance()

Returns the *_instance*. Unless no instance exists, in which case it returns null.

Map Objects

enemy.js

This class extends the Phaser class Phaser.GameObjects.PathFollower, since enemies need to follow a settled path.

Functions

constructor (*config*)

Sets configurations for the enemy.

follow(*self*)

Let enemies follow the specified path.

destroyEnemy(*self*)

Remove the enemy when its Hit-Point is 0.

Play Objects

enemy.js

Same as mentioned above.

Below files, each is a class extended from Phaser.Scene, and each is set a keyword in the constructor with the format: `super('mapLogicTower');`

gameStats.js

It handles game statistics.

Create() method documentation:
https://newdocs.phaser.io/docs/3.55.2/Phaser.Types.Scenes.SceneCreateCallback

Within it, firstly reuse data from other scenes:

```
//these allow usage of attributes and functions from other scenes
        var gameStats = this.scene.get('gameStats')
        var resources = this.scene.get('resources')
```

Then import map data from resources scene:

```
//take the starting parameters from the resources scene and feed them to
//the code is a bit verbose, this shortens it for later use
        var params = resources.mapData['startParams']

        this.lives = params['lives']
        this.money = params['money']
        this.score = params['score']
        this.speedSetting = params['speedSetting']
        this.isPlaying = params['isPlaying']
```

Update() method documentatation: https://newdocs.phaser.io/docs/3.55.2/Phaser.Scene#update

Within it, use attributes and functions from other scenes:

```
//these allow usage of attributes and functions from other scenes
        var map = this.scene.get('map')
        var gameStats = this.scene.get('gameStats')
        var gameRecords = this.scene.get('gameRecords')
```

Set the game over state when lives go to 0.

```
//in the case that the value of lives reaches 0, the game will change t
            if(this.lives <= 0){
                console.log('game over')
//record the current score to be displayed later
                gameRecords.score = gameStats.score
//call the function to update the high score if needed
                gameRecords.updateTopScore()
//record the number of waves survived to be displayed later
                gameRecords.wavesSurvived = map.origin.waveCounter
// this changes the scene
                map.timer = false
                this.scene.start('overState')
```

Set speed:

```
//change the speed at which different elements of the game run by m
//as the whether the game is meant to be playing is set as a boolea
    this.playSpeed = this.speedSetting * this.isPlaying
```

hud.js

Describes functionalities in the User Interface when playing the game where the map displays.

Load data from other scenes:

```
//these allow usage of attributes and functions from other scenes
        var hud = this.scene.get('hud')
        var hudLogic = this.scene.get('hudLogic')
        var hudLogicTower = this.scene.get('hudLogicTower')
        var mapLogic = this.scene.get('mapLogic')
```

Import the tile logic which is how the map is built and is written in another file later:

```
//call function that creates the settings container at the start
//this is so the building site tiles could be correctly interacted with
        hudLogic.startHud()
```

Add the menu button at top right:

```
//create the image of the circle at the top right of the game window
      this.circle = this.add.image(850, 50, 'circle').setInteractive()

//when the circle is clicked on, the function that makes the settings is called
       this.circle.on('pointerdown', function(){
           hudLogic.makeSettings();
           AudioManager.playEffect(SFX.BUTTON_CLICK);
       })
```

Add the speed-changing button at the bottom:

```
//create the speed button at the bottom left of the game window
      this.speedButton  = this.add.image(50, 650, 'arrow1').setInteractive()

//when the speed button is clicked on, the speed of the game changes
       this.speedButton.on('pointerdown', function(){
           hudLogic.changeSpeed();
           mapLogic.updateSpeed();
           AudioManager.playEffect(SFX.BUTTON_CLICK);
       })
```

Within **update()** method, after loading data from other files, (1) add buy-tower functionality; (2) update different game statistics; (3) let the timer runs normally:

```
   update()
   {
//this allows usage of functions from other scenes
       var hudLogic = this.scene.get('hudLogic')
       var hudLogicTower = this.scene.get('hudLogicTower')

//call the function that gives the buy tower buttons a tint if the
       hudLogicTower.updateTint()
//call the function that displays different stats stored in the ga
       hudLogic.showStats()
//call the function that displays the countdown timer between wave
       hudLogic.showTimer()
```

hudLogic.js

Manipulates the logic of HUD elements.

Set menu with 3 buttons, save the current map to be resumed:

```
//this function creates the setting window which contains buttons
    this.makeSettings = function(){
//save the delay time of the origin and towers so they can resume
        if(map.origin.delay){

//ensure that most elements in the game stop while the menu is active
        gameStats.isPlaying = false
```

Then draw buttons and set interactions with specific outcomes. This step is difficult since it requires programmers to think about all the elements in each state comprehensively.

Then draw the speed-changing button and set options:

```
//function for when the speed button is clicked on
    this.changeSpeed = function(){
//uses a switch as the options are discrete
//the speed setting increases by one except for when it i
//these just draw the buttons and change the settings, th
        switch(gameStats.speedSetting){
            case 1:
```

Then show the game stats:

```
//display the stats of the current playthrough
    this.showStats = function(){
//strings that will contain the stats from the gameStats scene
        var wave = 'wave: ' + (map.origin.waveCounter+1)
        var lives = 'lives: ' + gameStats.lives
```

And show the timer:

```
//this function displays the timer for when a wave is finished
    this.showTimer = function(){
//remove any previous iterations of the display of these stats
        mapLogic.removePrev(hud.timerText)
//check if the timer is active
```

hudLogicTower.js

Mainly manipulates the towers shown in the HUD, i.e. buy tower, sell tower, tower upgrades, etc.

After reusing data from other files,

this.makeTowerMenu = function(tile)

Function that creates the tower menu. Takes in as a parameter the tile that was clicked on which caused the function to be called.

this.makeMenuBuy = function(tile)

Function that displays all the towers that could be bought. Takes in the tile the that was clicked on.

this.updateTint = function()

Function that would dim the buy button in the tower menu if the cost of the tower exceeds the amount of money at the moment. The button reverts back to normal when there is enough money.

this.clickBuy = function(button, tile, tower)

Function that allows the buy buttons in the tower menu to be interacted with interactions involve hovering the mouse over is and clicking it to buy the tower. Takes in as parameters 1)the button in question, 2)the tile the tower menu is for and 3)the tower the button is for.

this.makeMenuStats = function(tile)

Shows the stats of the tower in the building site that was clicked on. Takes as a parameter the tile that was clicked on.

this.removeTower = function(tile)

Removes a tower from building site. Parameter: the tile the tower is to be removed from.

map.js

Mainly summarises all of the interactions and logics done by other files.

Set all the towers, enemies, and bullets into groups:

```
//the group that would contain all the enemies
    this.enemyGroup = this.add.group()

//the group that would contain all the towers
    this.towerGroup = this.add.group()

//the group that would contain all the bullets
    this.bulletGroup = this.physics.add.group()

//the group that would contain all stationary targets for AOE bullets
    this.targetGroup = this.add.group()
```

Then import tile logic. Set enemies' path.

Then add the 'start' button and set interactions.

In **update()** method:

Update (1) speed changes; (2) remove bullets; (3) determine the attack speed of towers; (4) searches for any enemies that are in a tower's range; (5) checks the state of the origin and creates an enemy if the state is the right one; (6) determine how fast the enemies spawn in a wave; (7) handles the countdown between waves; (8) call the next next wave when the rush button is clicked; (9) when a building site tile is clicked on, it opens up the tower menu in the HUD.

mapLogic.js

Contains the functions and methods that are generally called by map.js.

extends from Phaser.Scene. Contains the functions and methods that are generally called by map.js.

After loading attributes and functions from other scenes, draw the map and set up the interactivity within it:

this.drawTiles = function(tileCoords, tileset, size)

Function for drawing the tileset to the map. Takes in as parameters 1)the array of indices indicating what tile goes where, 2)the tileset image and 3)the size of the tiles.

this.makePath = function(tiles, origin, destination, pathTiles, size)

Function for making the path which the enemies will travel on. Takes in as parameters 1)the array of indices showing what tiles are where, 2)the indices of the origin and 3)destination, 4)the array of indices indicating what tiles the enemies could travel on and 5)the size of the tiles in pixels.

Then find the coordinates in tiles units the origin and destination tiles.

Create a zone object which will have many parameters controlling the spawning process.

Create a physics image object which when enemies go over, a life is lost and the enemy is removed.

```
'/it has zero height and width as it isn't seen, just allows enemies to overlap with it
        map.dest = map.physics.add.image(destinationCoords[0]*size+size/2, destinationCoords
        map.dest.displayWidth = 0
        map.dest.displayHeight = 0
```

To find the path, the external library easystarjs is used to do an A* search.

this.findTile = function(tiles, type)

Function used to find a specific tile in a tile array. Takes in as parameters 1)the array of indices indicating where different tiles go and 2)the index of the tile to find.

General functions:

this.updateSpeed = function()

Function for changing the movement speed of the current enemies and bullets as well as the reload speed of waves and towers and the cooldown time of of the origin during play. Within it: update the movement speed of bullets; the movement speed of enemies; the cooldown speed of the origin; the reload speed of the wave; the reload speed of the towers. Then add a ‘start’ button to let enemies in.

this.removePrev = function(object)

Function for removing the previous version of an object so there won't be multiple versions of the same object. Takes in as a parameter the object to attempt to remove.

mapLogicEnemy.js

Contains the functions and methods that are generally called by map.js relating to the enemy objects.

Functions

this.makeEnemy = function(key)

Function that makes the enemy object. Takes in as a parameter the key of enemy to make.

If the origin is ready to make another enemy: create a new enemy object; give it attributes for when it is moving and being attacked and its size; set how far the enemy has to travel; set its speed; set the HP bar (loads of codes); make the enemy follow the specific path at a specific speed; check if enemy reaches destination.

this.removeEnemy = function(enemy)

Function for removing the enemy. Parameter: the enemy to remove.

```
        this.removeEnemy = function(enemy){
//as the HP bar is made of graphics, its components need to be removed individually
            enemy.maxHPBar.destroy()
            enemy.HPBar.destroy()
//remove the enemy from the game
            enemy.destroy()
        }
```

this.debugEnemy = function(enemy)

Function for removing an enemy in case it is buggy. Parameter: the enemy to remove.

this.updateHPBar = function(enemy)

Function for updating the HP bar of an enemy. Parameter: the enemy whose HP bar to update.

this.startGame = function()

Function for starting the game proper by sending the first wave.

this.rushWave = function()

Function for starting the next wave prematurely when the rush button is clicked on while the countdown is active.

this.makeWave = function(waveData)

Function for changing the parameters of the origin so it is ready for another wave. Parameter: the data for the next wave.

this.updateWave = function()

Function for determining the speed at which the enemies are made. Also stops making enemies when enough have been made.

this.reloadWave = function()

Function for changing the state of the origin so more enemies could be made.

this.coolDown = function()

Function for seeing if the origin is ready for another wave and if it is, call a function to change the origin parameters to fit this state. Also determines if the winning conditions have been met in which case the game state will change to the game complete state.

this.callNextWave = function()

Function for calling the next wave.

this.enemyReachDest = function(enemy, dest)

Function for when an enemy reaches the destination. Parameters: 1)enemy; 2)destination.

mapLogicTower.js

Contains the functions and methods that are generally called by map.js relating to the tower objects.

Functions

this.clickSite = function()

Function for when a building site tile is clicked on.

this.makeTower = function(tile, key, size)

Used to create a tower at a specific point. Parameters: 1)the tile to make the tower in, 2)the key of the tower and 3)the size of the tiles.

this.clickTower = function(tower)

Function for when the tower is clicked on. Parameter: the tower that was clicked on.

this.searchEnemy = function(enemy)

Function for searching for an enemy within a tower's range. Parameter: the enemy being searched.

this.enemyFound = function(enemy, tower)

Function for when an enemy has been found. Parameters: 1)the enemy that was found and 2)the tower that found it.

this.makeBullet = function(enemy, tower)

Making the bullet. Takes in as parameters 1)the enemy being targeted and 2)the tower that the bullet originated from.

this.updateTower = function(tower)

Regulating the attack speed of the tower. Takes in as a parameter the tower being affected.

this.reloadTower = function(tower)

Changing the state attribute of a tower so it may fire another bullet. Parameter: the tower being affected.

this.moveBullet = function(bullet)

Moving the bullet. Takes in as a parameter, the bullet that is affected.

this.hitTarget = function(bullet, target)

Function for when the bullet reaches its target. Parameters: 1)the bullet and 2)the target.

this.damageEnemy = function(bullet, enemy)

Responsible for damaging the enemy. Parameters: 1)the bullet that will damage the enemy and 2)enemy itself.

this.updateBullet = function(bullet)

Function for when a target enemy is destroyed before a targeting bullet has made contact with it. Takes in as a parameter the affected bullet.

states

Each file is one class which is extended from Phaser.Scene, and each is set a keyword in the constructor with the format: `super('bootstrapState');`

bootstrap.js

Handles what happens when the winning conditions for the game are met. This file loads and initialises all of the Phaser objects for the audio manager in order to facilitate testing as the mocha.js testing suite crashes when it tried to load Phasers objects, meaning we had to remove this section from that code in order to run our tests.

First we import Phaser and the audioManager itself:

```
import Phaser from 'phaser';
import AudioManager, { MUSIC } from '../audiomanager/audioManager.mjs';
```

Then we launch all of our scenes necessary to run the game:

```
this.scene.launch('resources')
this.scene.launch('checkJSON')
this.scene.launch('gameRecords')
this.scene.launch('mapLogic')
this.scene.launch('mapLogicEnemy')
this.scene.launch('mapLogicTower')
this.scene.launch('hudLogic')
this.scene.launch('hudLogicTower')
```

Then we create our scene loader/switcher, allowing us to change between scenes:

```
update(time, delta) {
    if (this._loaded && this._elapsedTime > MIN_DISPLAY_TIME)
    {
        this._switchScenes();
    }
    this._elapsedTime += delta;
}

_switchScenes() {
    console.log('Loading Complete... Switching to Menu State');
    this._progressBox.destroy();
    this._progressBar.destroy();
    this.scene.stop('bootstrapState');
    this.scene.start('menuState');
    AudioManager.playMusic(MUSIC.MAIN);
}
```

complete.js

Handles what happens when the winning conditions for the game are met.

First we stop the other scenes to prevent them from overlaying this one:

```
this.scene.stop('gameStats')
this.scene.stop('hud')
this.scene.stop('map')
```

Load the gameRecords: `var gameRecords = this.scene.get('gameRecords');`

Create new variables and show them:

```
//this string will be displayed to show the score from the current playthroug
    var score = 'Score: ' + gameRecords.score;
//this string will be displayed to show the highest score from the current pl
    var topScore = 'Top Score: ' + gameRecords.topScore;
//this string will be displayed to show how many lives remain from the currer
    var lives = 'Lives Left: ' + gameRecords.lives;
```

Add logic to trigger the ‘happy ending’ page if the condition is ‘instant win’.

Display:

```
//display the previously made strings giving a new line to each
this.endResults = this.add.text(textX, textY, score + '\n' + topScore + '\n'
```

Create 3 buttons: ‘menu’, ‘level’, ‘restart’.

level.js

This is the level selection scene for selecting levels.

We close other scenes to prevent them from overlaying the menu:

```
this.scene.stop('gameStats')
this.scene.stop('hud')
this.scene.stop('map')
```

Add a title bar:

```
this.bar = this.add.image(450, 100, 'level_selection_bar')
```

Create the ‘back’ button at top left:

```
//creates an image that could be clicked on
        var back = this.add.image(50, 50, 'left').setInteractive()
//when the image is clicked on the game would be the menu state
        back.on('pointerdown', function () {
            this.scene.scene.start('treasureState');
            AudioManager.playEffect(SFX.BUTTON_CLICK);
```

Then add the functionality:

this.levelButtons = function()

This function positions the buttons which when clicked on would take the player to the level that was clicked on. Create a for loop to build level buttons.

this.makeButton = function(level, x, y)

This function is used to make a button for each level that could be played. Takes in as parameters 1)the key of the level and the 2)x and 3)y coordinates of where the button would be located.

this.clickButton = function(level)

Function to handle what happens when a level button is clicked on. Takes in as a parameter the name of the level.

Call the functions.

menu.js

Handles the main menu of the game.

First we stop other game scenes.

Add background image:

```
this.background_jungle = this.add.image(450, 350, 'background_jungle')
```

Then we create the ‘start’ button at bottom, and set interactions and sounds:

```
//create an image that does something when clicked on
        var start = this.add.image(450, 620, 'start').setInteractive()
        start.displayWidth = 200
        start.displayHeight = 160

//when the image is clicked on, the scene changes to the 'treasureState' sce
//code to change scenes from https://www.thepolyglotdeveloper.com/2020/09/sw
        start.on('pointerdown', function () {
            // Please debug using the below sentence. Try change it btw 'lev
            this.scene.scene.start('treasureState');  //changed by XYu Mar5.
            AudioManager.playEffect(SFX.BUTTON_CLICK);
        })
```

over.js

Handles what happens when the defeat conditions for the game are met.

After stopping other scenes, we display the results of the game:

```
//display the previously made strings giving a new line to each
        this.endResults = this.add.text(350, 0, 'Game Over \n' + score + '\n' +
        topScore + '\n' + waves, { font: '32px Arial' })
```

We then display 3 buttons:

```
//display some images, make them interactable and certain sizes
        var menu = this.add.image(450, 250, 'menu').setInteractive()
        menu.displayWidth = 200
        menu.displayHeight = 100
        var level = this.add.image(450, 400, 'level').setInteractive()
        level.displayWidth = 200
        level.displayHeight = 100
        var restart = this.add.image(450, 550, 'restart').setInteractive()
        restart.displayWidth = 200
        restart.displayHeight = 100
```

And set interactions and sounds.

play.js

Handles what happens when the game is being played.

Start all the scenes that make the game up itself:

```
this.scene.launch('gameStats')
this.scene.launch('map')
this.scene.launch('hud')
var gameRecords = this.scene.get('gameRecords');
```

Make sure ‘instant win’= False, since users haven’t set ‘the Law’ yet:

```
gameRecords.instantWin = false;
```

treasure.js

This is the 'treasure box' page which shows lists of characters.

Firstly, stop interactions from other pages:

```
    this.scene.stop('gameStats')
    this.scene.stop('hud')
    this.scene.stop('map')
```

Then, add the background picture from the resource.js:

```
this.background_treasure = this.add.image(450, 350, 'treasure_bg')
```

Create a 'start' button and set interactions.

utilities

checkJSON.js

This utility extends Phaser.io's default Phaser.Scene. It allows for the application to validate JSON files, ensuring their integrity and compatibility with the application.

Functions

checkDict(dict, keys)

Checks that every relevant value of a given dictionary is the right data type.

It takes, as input:

- *dict*, the dictionary to check.
- *keys*, a second dictionary that contains the data types for all of the relevant values for any dictionary used in the game. Only those covered by the key dictionary are checked.

checkLevel(dict, data, key)

This function is used specifically for checking the inner arrays of the level dictionaries.

It takes, as input:

- *dict*, the dictionary to add the data to.
- *data*, the dictionary containing all of the data to check.
- *key*, the key of the dictionary to check.

By default, all dictionaries pass, therefore, the data may be added to the larger level dictionary. If any of the tests fail, then it will not be added to the larger level dictionary.

gameRecords.js

This utility extends Phaser.io's default Phaser.Scene. It allows for levels to be selected

Parameters

- *levelSelect* (default: -1) This is how the levels are selected and the selection is preserved.
- *wavesSurvived* This records how many waves the player has survived.
- *lives* This records how many lives the player has remaining.
- *score* This records the highest score the player has attained during the current game session.
- *topScore* (default: 0) This records the highest score the player has attained during the current game session.
- *instantWin* (default: false)

Functions

updateTopScore*()*

Updates the top score by comparing it to the current score. If the current score is higher, then the top score is changed to the value of the current score.

resources.js

This utility pre-loads all of the necessary assets for the game.