Agile Software Project

Oct 2021, Group 8

Members: (Alex) Hoi Chu, Dimitri Vlachos, Jeremy Matthews, Sharif Khan, (Freda) Xiaoyun Yu

List of Contents

Introduction (Backgrounds, Scope and Motivations)

Aims and Objectives

Research

- Academic Research
- Market Research

Stakeholders

Timescale of Work

Questionnaire & Data Analysis

User Story & User Requirement

Requirements Specification

Use Case & UML

Prototyping

Development Process (with Iterations and Expectations)

Assumption Tests

- User Tests
- Accessibility Testing
- Usability Testing

Evaluation Techniques

Introduction

Tower Defence games are ubiquitous, popular, and engaging. This project proposes the development of our own **Tower Defence game with a stretch goal of implementing educational elements** to boost engagement both in the gaming aspect and the educational material. Our project takes a **user centred-design** approach to the development of requirements, objectives, and analysis.

Background

The recent pandemic, though devastating, has opened a lot of opportunities for distance work and distance collaboration. Now is a better time than ever to be collaborating remotely, as, not only have the tools for this become more readily available, but we, as individuals, are no doubt more experienced and used to this kind of work and collaboration by now. We therefore have the chance to develop computer-related products further.

Specifically, we want to focus on 'Tower Defence Games'. We hope to create a new TD Game that especially belongs to us. Through our teamwork and collaboration, we intend to create our own version of the quintessential tower defence game. Our game will be themed around animals, with humans as their primary adversary. As such, our map / maps will be animal themed. Forests or farms or something with an animal connection.

We chose to create a tower defence game as it was a **similar point of interest** for us all. We expressed an interest in games and education. Our initial brainstorming make us decide to try and create and educational version of a tower defence game, using gamified elements to encourage thought and learning. However, we realized that, due to time constraints, this may be a difficult goal to achieve and it is not part of our minimum viable product. Given the chance, this is definitely a stretch goal. But it is not the sole theme of our project.
Our group has a varied skillset, from organizers to programmers. As such, we have proportioned our workload to allow each of us to levy our abilities and contribute to the project. We have planning set up, a GitHub and our slack channel.

We have decided to **use JavaScript**, considering our shared experience and classes. However, we have yet to decide on a physics engine for the game as we are still split between matter.js and phaser.io, both are excellent applications for our purposes.

Scope

The scope for our project aims at a **generalised audience**, one which is at least passingly familiar with tower defence games and how they work. This would tend toward a **younger audience**, one which, if time allows, an educational angle would be effective on.

The scope of our project broadly encompasses the creation of a Tower Defence game in whole. With the minimum viable product being one which has a life system, an enemy, a tower, a map and a basic ai that

would control the enemies pathing, the towers and the player's lives.

The main limitations of our project is **time and experience**. While we may have grand ideas, it is important to keep the scope of our project reasonable as to be able to complete it with both the time and abilities which we collectively posses.

Motivation

Our motivation for the project was initially to find a way to integrate **education** into a tower defence game. Educational games and tower defence games typically do not mix, however, by gamifying the educational aspect, we can make the action of learning easier, faster, and more enjoyable than simply reading from a book.

Another motivation for our choice was our **mutual love of tower defence games**. It will be easier for us to work on a project that is a labour of love. Some of us have worked on a few projects in Unity in the past. Another member revealed that he is a game developer by trade, adding substantial knowledge experience to the group.

Through our brainstorming we have thought of a few methods that education could possibly be integrated: a question about history or math would appear, asking the player for a decision. Perhaps it asks the player to invest some of their money, giving two options which could in fact be formulas. Choosing the correct option would grant the player additional funds. Another question could reduce wave size or difficulty, or perhaps allow a tower to be upgraded.

Aims and Objectives

- On the whole, To build a **Tower Defense (TD) Game** which operates on computers.
- 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.
- 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) meeting arrangement, ddl middle Dec. (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.

Research

Academic Research

- According to Rummell[1], they are adapting **AI** to play tower defense games. His start point is to **solve** the already existing game, instead of creating such a game. However, he acknowledges that AI cannot solve tower defense game as well as a human can do. This may draw a conclusion that our tower defense game **encourages creation** which differs from other game categories; and also it attracts users by its uniqueness and enjoyment when creating their own arts.
- According to Groeschel and Schäfer[2], TD games are widely seen in **mobile** apps. We have seen types of Revenue Models in mobile app design, which may be helpful when we consider entering into the market.
- According to Yu Du, Jian Li, Xiao Hou, Hengtong Lu, etc[3], they had developed a method to **automatically generate levels** for one TD game, which means our **competitors are faster** to generate a new level than us.
- According to Vasudevan Janarthanan[4], there are indeed serious games with **educational theme**. He mentions that, playing games can **improve players' self esteem**. We think it's beneficial in both ways – for us, it helps us to attract users since they enjoy such a game; and for users, they indeed are emotionally healthy when playing our game.

Market Research

Full documents please see:
https://github.com/FredaXYu/ASP_Group8/tree/main/bg_research/Market_Research
https://github.com/FredaXYu/ASP_Group8/tree/main/bg_research/Research%20Results

According to Stream website the hottest-selling TD games -

	Theme	2D / 3D,	Design	Inspirations

[1] P. A. Rummell, "Adaptive AI to play tower defense game," 2011 16th International Conference on Computer Games (CGAMES), 2011, pp. 38-40, doi: 10.1109/CGAMES.2011.6000357.
[2] Michael Groeschel, Tim Schäfer, "Analysis of Mobile App Revenue Models Used in the Most Popular Games of the Tower Defense Genre on Google Play", Asian Journal of Computer and Information Systems (ISSN: 2321 – 5658) Volume 08– Issue 01, February 2020.
[3] Yu Du, Jian Li, Xiao Hou, Hengtong Lu, et al, "Automatic level Generation for Tower Defense Games", ITNEC 2019.
[4] V. Janarthanan, "Serious Video Games: Games for Education and Health," 2012 Ninth International Conference on Information Technology - New Generations, 2012, pp. 875-878, doi: 10.1109/ITNG.2012.79.

		angle of view		
Tribes of Midgard	Cute, Aesthetic	3D, moving	Adventure, Christmas, Tribes, Simulation	Very suitable for kids. Story-telling.
The Riftbreaker	War, Alien	3D, moving	Simulation	Noisy & Dark. Don't learn from it.
Plants vs. Zombies GOTY Edition	Cute, Zombie	2D, still	Cartoon, Casual	Specially written music, very relaxing, suitable for kids.
Creeper World 4	Very gentle war	Combined, still	Strategy	The map shows the topography of the whole local place. It's like the professional mission control room during a real war.
Mindustry	Pixel	2D, still	Base-building, in grids	The weapon and attack designs are simple. It's more like a puzzle game. The map is very big. In Morandi colours.
Kingdom Rush	Cute, War	2D, moving	Casual, story-telling	Special bg music, cute characters although in war theme, bright, relaxing

Looking through the top-rated TD games, we find out that most are: (1) in cartoon theme; (2) colourful; (3) casual. We will also make similar features.

Our **competitors**' advanages: (1) have designers who are good at drawing cartoon characters; (2) some of them have built 3D models; (3) are good at arranging the equipment library for towers and characters; (4) are cooperating with music-producing groups; (5) have a good team for propagation; (6) have a good system to arrange money profits from the users.

We must think about something **unique** that will lead us to win. We can: (1) just like Kingdom Two Crowns, we can have a very attractive story as the main clue; (2) don't have to draw 3D characters, but should use artistic designs for simple drawings; (3) can win on the background music; (4) should expand our user groups, especially for the students group, so that we can earn money from them; (5) at the end of development, contact the main game platforms to have a test; (6) can win on the educational meanings.

After examinating several TD games, we know:

1) Types of towers:
The cheap tower with average speed with low firepower
The slow, high range tower with high firepower
The slow, AOE(area of effect, in essence, targets a radius rather than an individual) tower
The slow, splash(hits to target also damage nearby creeps) tower
The fast but low firepower tower
The affliction(negatively impacts creeps. For example, poison or stun) tower
The tower that moves within a set area instead of being planted in one spot
Towers that generate building resources for future use over time

Towers that can target multiple creeps at once, different from splash or AOE damage as this is intentional as opposed to coincidental
Towers that periodically spawn ally units to stop and damage creeps
Towers that can create chain damage, meaning that attacks bounce between nearby creeps
The very expensive but powerful tower
The very slow, very high range, AOE tower with strong firepower
The very fast, very long range, AOE tower
Obstacles in the creeps' path, for example walls or landmines

2) Types of enemies:
The average health, average speed creep
The low health but fast creep
The high health but slow creep
The very high health, slow and low spawn rate creep. For example, bosses
Creeps that summon more creeps over time
Creeps that can disable towers for a period of time. For example, stun nearby towers so they don't attack for a short period of time.
Creeps that can help nearby creeps. For example, give creeps that give their allies shields, regenerate their health, link their health pools, boost their speed
Creeps that require certain damage type. For example, flying creeps that require anti-air towers to hit

Challenges: (1) pick out a brilliant story; (2) have a sense of design; (3) music production; (4) the system for users to pay, and we gain money; (5) unique combinations of game styles; etc.

Stakeholders:

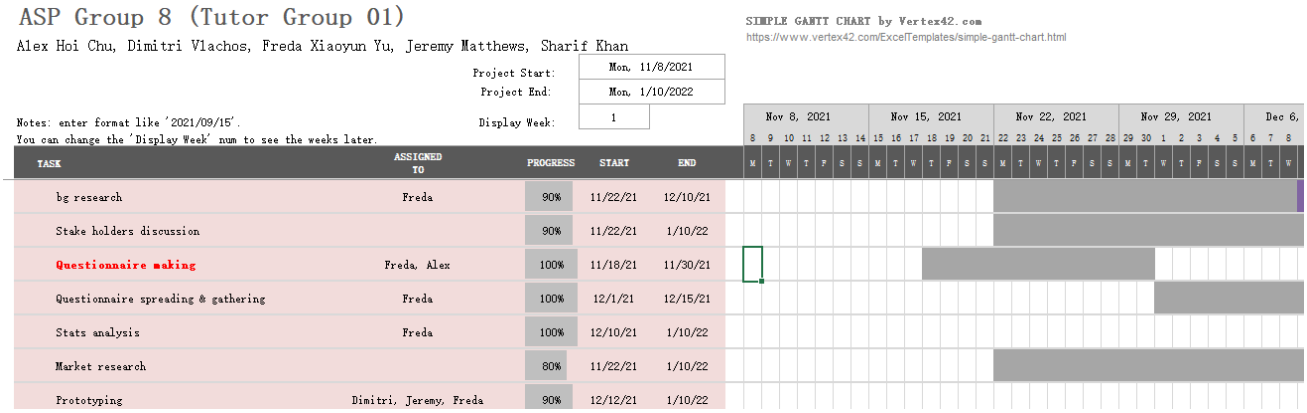
- End-users. Which may contain: western country users & eastern country users; adults & children; students; leaders; experienced game industry people, etc.
  Our relationships: (1) we earn money from them; (2) we teach knowledge and the right inspirations to them; (3) we get feedbacks from them so that our project could be better, and we learn from this process; (4) we build our reputation based on their positive evaluations and enthusiasim.
- Classmates in UoL. Our relationships: (1) they are the group who are most willing to fill in the questionnaires for us, since we perhaps will also fill in their questionnaires; (2) we learn from each other's projects, and the project-management structure; (3) we are potential competitors, since our grades would be different, and we guess UoL would keep our grades in bell-shape.
- Teachers in UoL. (1) Of course teachers hope to see the result of our project; they can provide advices once they are willing to; (2) they will strictly assess the outcome of our product, so we'd better manage our project both efficiently and successfully.
- Production company. Relationships: (1) if we sign a contract with a professional game-production company, then they will break one piece from the whole profit – they will earn some money from us; (2) they ensure that our project can be published to the end-users; (3) perhaps we can learn from them

since they are professional.

- Competitors in game industry who also make tower defense games. We hope to keep a healthy relationship with them – not competing, but learning and collaborating, since we all want to earn money from end-users. If we can find any experienced teams who make TD games, that would be a very good chance for us to learn.

Timescale of Work

Please refer to the Gantt Charts uploaded here (**milestones** are highlighted in red):
https://github.com/FredaXYu/ASP_Group8/tree/main/Gantt_chart/Updated_Versions_(please_add_here)
This folder contains both the **tasks before the midterm**, and **tasks after the midterm**.
A snapshot:



ASP Group 8 (Tutor Group 01)

Alex Hoi Chu, Dimitri Vlachos, Freda Xiaoyun Yu, Jeremy Matthews, Sharif Khan

SIMPLE GANTT CHART by Vertex42.com
https://www.vertex42.com/ExcelTemplates/simple-gantt-chart.html

Project Start: Mon, 11/8/2021
Project End: Mon, 1/10/2022
Display Week: 1

Notes: enter format like '2021/09/15'.
You can change the 'Display Week' num to see the weeks later.

TASK	ASSIGNED TO	PROGRESS	START	END
bg research	Freda	90%	11/22/21	12/10/21
Stake holders discussion		90%	11/22/21	1/10/22
Questionnaire making	Freda, Alex	100%	11/18/21	11/30/21
Questionnaire spreading & gathering	Freda	100%	12/1/21	12/15/21
Stats analysis	Freda	100%	12/10/21	1/10/22
Market research		80%	11/22/21	1/10/22
Prototyping	Dimitri, Jeremy, Freda	90%	12/12/21	1/10/22

Questionnaire & Data analysis

We have designed and written a series of questionnaires to gather potential user opinions. There are single-choice questions, multiple-choice questions, and free-text questions. We use the platform called '**Typeform**'. Since there are restrictions for non-VIP, we have to limit our question amount to 10, and also the response amount to 10.

Here are our questions:

1→ Hello, what is your gender? （性别？）*

Description (optional)

- A Male （男）
- B Female （女）
- C Others （其他）

Add choice

2→ Where are you from? （你来自世界上的哪里？）*

Description (optional)

- A Asia （亚洲）
- B North America （北美）
- C Central America （中美）
- D South America （南美）
- E Europe （欧洲）
- F Africa （非洲）
- G Middle East （中东）
- H Australia & Oceania （大洋洲）
- I Pacific / Atlantic Islands （太平洋/大西洋岛屿）

3→ What's your job? Where do you work? (Choose the one that best describes you) （你的工作？选一个最贴切的）*

Description (optional)

- A Experienced programmer / Game industry （资深程序员/游戏行业）
- B Student （学生）
- C IT related （信息技术）
- D Education （教育）
- E Government （政府）
- F Hospital & health （医院）
- G Bank （银行）
- H Airport （机场）
- I Arts & music （艺术&音乐）
- J Performance & host （表演&主持）
- K Communication （沟通）
- L Marketing （商科）
- M Free （自由职业）
- N Others （其他）

Add choice

4→ Do you know about Tower Defense Games? （你知道塔防游戏吗？）*

Description (optional)

- A No, I've never played it before. （不知道，没玩过）
- B Somewhat, I've played Plants vs. Zombies, but don't know the name of the game category. （知道一点，玩过植物大战僵尸/保卫萝卜，但不知道这个种类的游戏叫啥）
- C Yes, it's just building towers and then defense the base camp from enemies' attacks. （知道，就是为大本营筑炮塔抵御敌人的攻击）

Add choice

5→ How often do you play Tower Defense Games? （多久玩一次塔防游戏？）*

Description (optional)

- A Almost everyday （几乎每天）
- B 2~3 times a week （一周两三次）
- C Once a week （一周一次）
- D Once a month （一个月一次）
- E Seldom （从不）

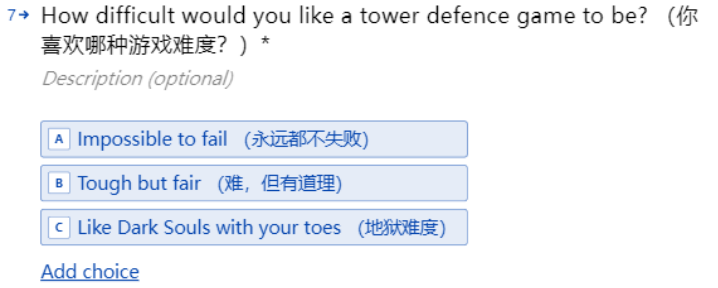
Add choice

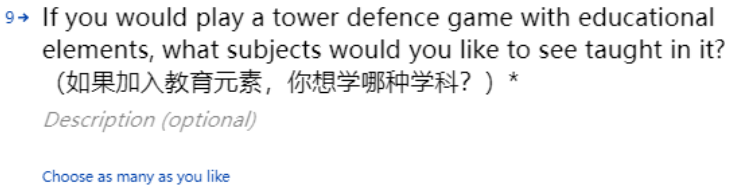
6→ Of the tower defence games you like, what game(s) did you like? （塔防游戏中，你最喜欢哪个？）

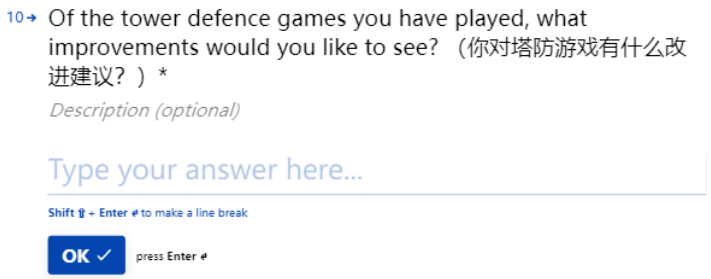
Description (optional)

Type your answer here...

OK ✓ press Enter ↵







To compensate the loss of response amount in Typeform platform and also to **understand the Asia market**, we have developed another questionnaire specifically for the mainland China area. We use **Tencent Document** platform. Also, we asked more questions including: 'Your age range? ' 'What would you like to see in a tower defence game that you haven't seen before? eg. mechanic, style. ' 'How long do you play TD games in one sitting? '

Finally we have got 17 responses in total. Here is the data result:
https://github.com/FredaXYu/ASP_Group8/blob/main/Questionnaire/Combined_results.xlsx

Our data analysis process:
https://github.com/FredaXYu/ASP_Group8/blob/main/Questionnaire/ASP_Questionnaire_Data_Analysis_XiaoyunYu.pdf Snapshot:

In [85]:

```
# Now we get to know the basic statistics for each column, using 'describe()' function:
df.describe()
```

Out[85]:

	#	gender	from	job	Do you know about Tower Defense Games? (你知道塔防游戏吗？)	How often do you play Tower Defense Games? (多久玩一次塔防游戏？)	Of the tower defence games you like, what game(s) did you like? (塔防游戏中，你最喜欢哪个？)	How difficult would you like a tower defence game to be? (你喜欢哪种游戏难度？)	Cute & cartoon (卡通可爱)	War-related (战争相关)
count	17	17	17	17	17	17	14	17	9	8
unique	17	2	3	8	3	4	14	3	1	1
top	Yilong	Male (男)	Asia (亚洲)	Student (学生)	Yes, it's just building towers and then defens...	Seldom (从不)	Haven't played it. 没玩过	Tough but fair (难，但有道理)	Cute & cartoon (卡通可爱)	War-related (战争相关)
freq	1	9	11	4	9	11	1	15	9	8

4 rows × 26 columns

Data set descriptions:

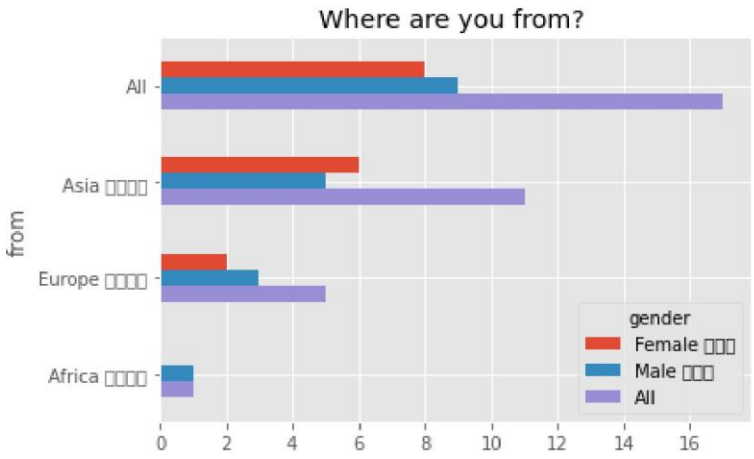
- **Sample size: 17. Chinese samples: 7; world-wide samples: 10.**
- Since there are question amount restrictions in the Typeform website, we have to limit our questions and delete others. There are 10 questions in Typeform questionnaire in total, and this form is prepared for the world-wide group. Also, there is a sample size limit for the Typeform, so we only have 10 samples.
- Since Xiaoyun Yu is just based in mainland China, she can collect opinions from the people surrounding her. She believes that in this way we can obtain more opoinions, so she has made another questionnaire

form using Tencent form collection method. There is no restrictions for question amount or sample size, so for the questionnaire's China version, we have asked almost all of our concentrated questions there.

- Finally, we have collected all the results into one data file - 'Combined_results.xlsx', where we only list the common 10 questions for both versions.

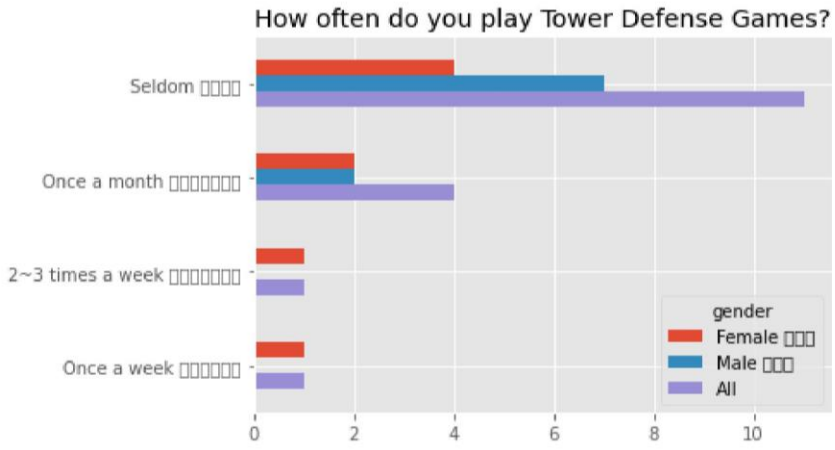
In [86]:

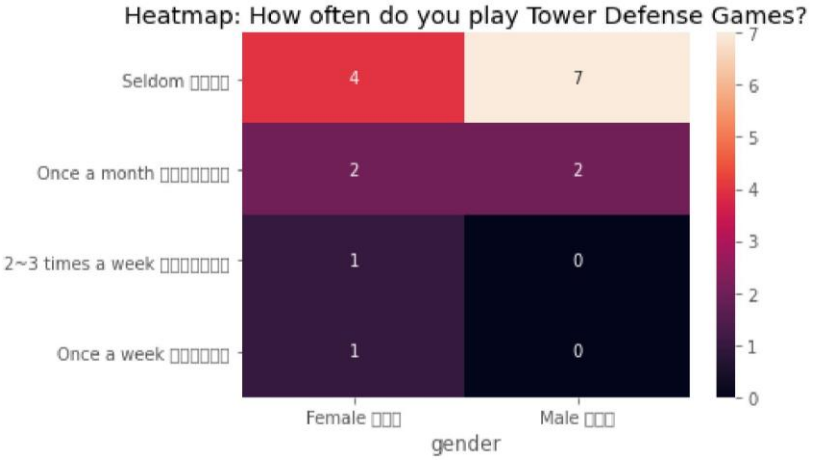
```
table = pd.crosstab(df['from'],
                    df['gender'], margins=True)
# table.drop('All', axis='rows', inplace=True)
table = table.sort_values(by='All', ascending=False)
# table.drop('All', inplace=True)

ax = table.plot.barh()
ax.invert_yaxis()
ax.set_title('Where are you from? ')
plt.show()
```



Insights:

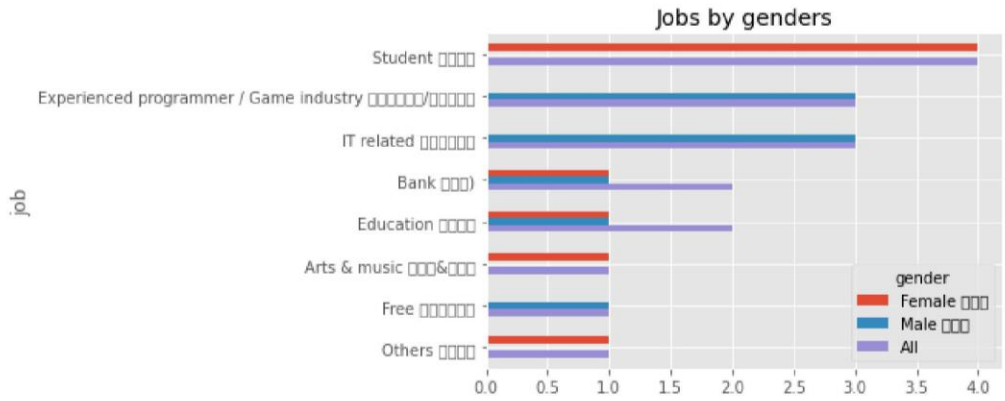
- There are 8 female guests, and 9 male guests in total. They are almost equal, which is quite good for us to analyze gender requests.
- **Most of our guests are from Asia (11 people).** 7 are from mainland China that's certain, and the other 4 may come from other areas.
- We have 5 Europe guests, and 1 Africa guest. These samples may help us to understand the Europe market, which perhaps is different from Asians' thoughts.
- Now we have questions: Do Asians have the same preference or different preferences from the European or African people? Can we combine these preferences?
- We need to decide our market range based on the location question - Do we need to proporgate our product only to Europe-America market? Or do we want only the Asia market? Or do we have bigger ambition to march to world-wide market (Can we combine all the interests)? What's our focus?





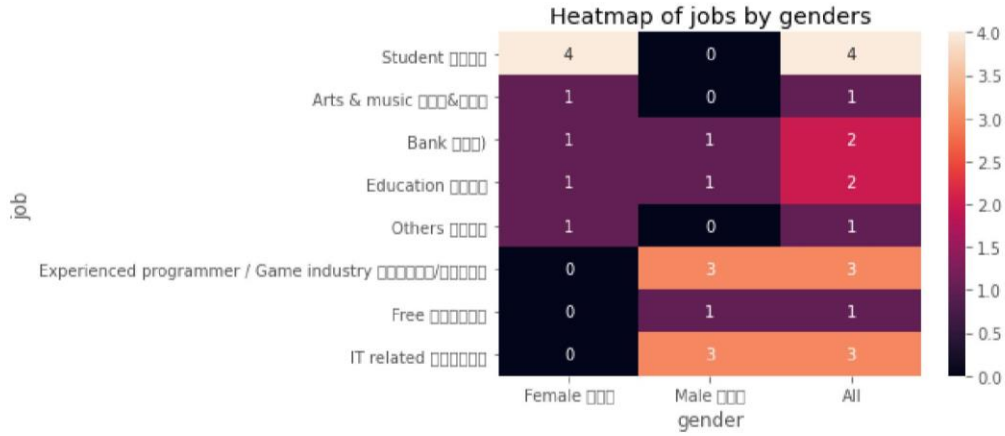
Insights:

- **Most of the people in our sample, don't play TD games at all or just play them quite few times.**
- The ones who play TD games more ofter, are females.



In [51]:

```
# Heat map:
table = pd.crosstab(df['job'],
                    df['gender'], margins=True)
# table.drop('All', axis='rows', inplace=True)
table = table.sort_values(by='Female （女）', ascending=False)
# table.drop('All', axis='columns', inplace=True)
table.drop('All', inplace=True)

ax = sns.heatmap(table, annot=True)
ax.set_title('Heatmap of jobs by genders')
plt.show()
```



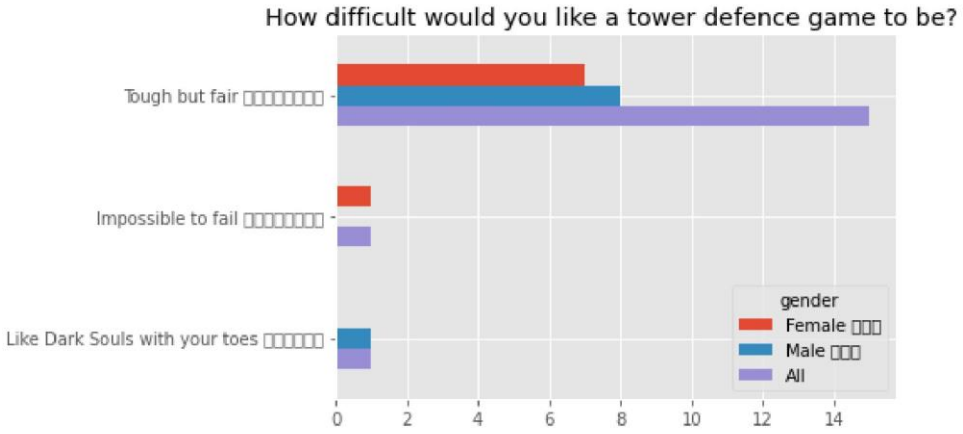
- We settle this job question, in order to: (1) figure out whether our data sample has **any bias with regard to occupations**. For example, do most of the answers come from programmers? If so, then there would be bias about their theme preference, topic preference, play duration and frequency, and so on. We need to emphasize our goal - we wish our product can be used by common people, and, we wish from our questionnaire, we can hear diversified voices so that later on we can settle down our target user groups.
- (2) If there are experienced programmers, then no problem, we have this honour to hear from them about **any advice** on our product, as we are all inexperienced to make TD game.
- There are many students. And of course, our main target group would be college students who are free for time and most possible to play video games.
- There are 3 experienced programmers, and 3 IT-related people, and 2 people working in educational positions. We need to be careful about any bias this may bring for us, and we indeed look forwared to their suggestions.

In [52]:

```
table = pd.crosstab(df['Do you know about Tower Defense Games? （你知道塔防游戏吗？）'],
                    df['gender'], margins=True)
# table.drop('All', axis='rows', inplace=True)
table = table.sort_values(by='All', ascending=False)
table.drop('All', inplace=True)

ax = table.plot.barh()
ax.invert_yaxis()
ax.set_title('Do you know about Tower Defense Games? ')
ax.set_ylabel(' ')
plt.show()
```

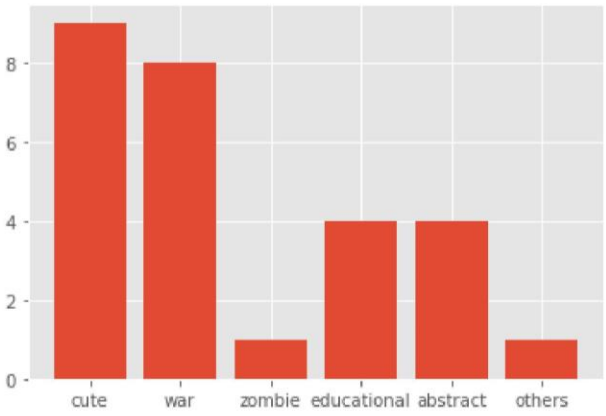
Insights:

- Interestingly, although many people seldom play TD games, still most of the people claim that they know what TD games are. Perhaps it's because the misleading of our choices, and people always don't want to admit that they have unknown knowledge about some simple concepts.
- There are 4 people state that they don't know about TD games at all. It means, for common people, there is **big possibility that our TD game would be their first TD game**. So we should make more tutorials, and, make more attrctive elements.



Insights:

- Most of the people hope to play games which are **a little bit difficult but with some sense**.
- One female hopes to play a game that never fails. That means, she hopes to relax as much as she could.
- One male hopes to challenge his max ability.

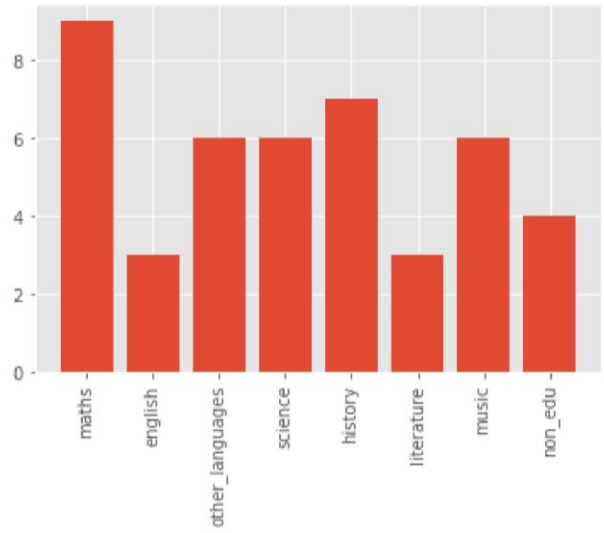


Insights:

- **'Cute & cartoon' receives the highest** vote rate. Some people who choose 'war-related' also have chosen 'cute & cartoon'. It means, no matter the theme is, we should make our game cartoon-like, instead of making it similar to real people or device.
- We have doubted whether 'cute & cartoon' is only preferred by Asians. But in fact, if we look into the answer chart, we can see that many European people and even males choose 'cute & cartoon'. It means, this choice is people's choice.
- The choice with silver gold is **'war-related'**. Both Asians and people from other places choose this choice. It means, we should make our game engaged with war-related elements - rivalry, weapons, fire, battlefield, etc. We should discover the attractive elements of the battlefield, then cartoonize them.
- For 'educational' the theme we care the most, we find that there are indeed 4 people support it. Of course our guest who works in education area has chosen this choice, and there are still other people would like to see educational elements.

In [91]:

```
plt.bar(subject_dictionary.keys(), subject_dictionary.values(), 0.8)
plt.xticks(rotation=90)
plt.show()
```

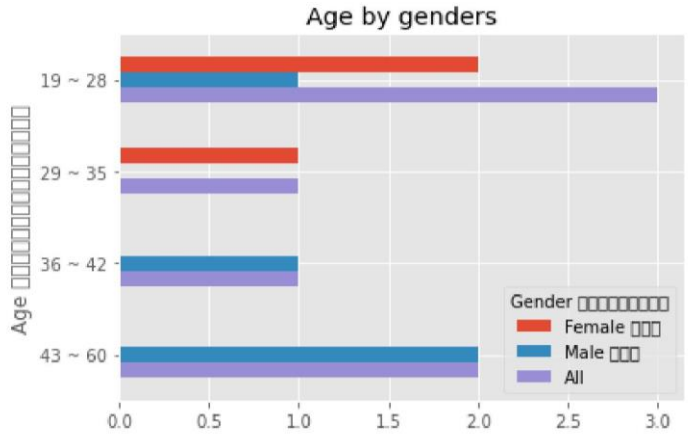


Insights:

- There are 17 people in our sample, and there are 4 people have chosen 'I don't like educational theme'. According to this result, we should be **very careful about how to implement any educational elements**. Are they boring? Are they serious? Are they tough? These could all lead to the loss of market.

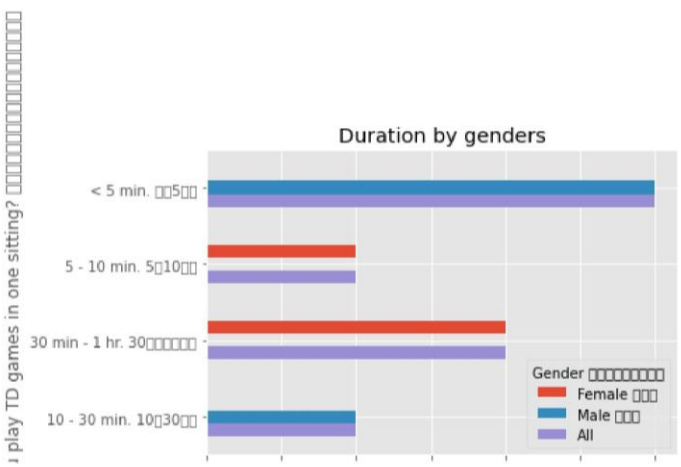
- In our result, **'maths' wins**. Perhaps some of the reason is that there are IT-related people and many programmers. Anyway, we should be confident if we hope to make a maths-related game.
- Since I've taken samples in mainland China, people seem to be lack of interests in learning English.
- People would like to learn history. We can consider whether we will make a game with a background taken from real history.
- People like science topic.
- There are indeed many more subjects and topics not included here.

Now let's check the additional questions asked in Chinese version questionnaire:



Insights:

- Top 1 group people in our sample (mainland China version) is young people.
- We have also many samples from middle-aged people.
- Many sampled people here are young females, and middle-aged males.



Insights:

- Almost a half of the people here only play TD games < 5 min, which means they seldom play this type of game. We should consider how to attract these group of people.
- Surprisingly, there are 3 females play TD games > 30 min in one sitting. We should consider how to meet their needs in developing a game with continous stories. And, we should consider how to constrain their usage of screen for protection of their health, as we truly hope to educate people.

In [ ]:

In [ ]:

Suggestions from all answers collected:

- 'Ability to speed up attacks if there is a level system, so you dont have to wait for the attack and defense to play out. '
- 'Skin customization. '

- 'More levels. '
- 'More fair less grind.'
- 'Naming squads or troops ability. '
- 'Bigger freedom for enemy's movement. '
- 'More skills and magic. '
- 'Prefer less payment. '
- 'I am slow to accept new things. Don't update the types of tower too often. Don't like big games. '
- 'I hope Plants vs Zombies can allow users to design the map or site just like Tank Battles. '
- 'I hope the shortcut keys would be more user-friendly. '

Our (Chinese) users hope to see these elements:

- 'Plants, animals, foods. '
- 'I hope to see the "Pass" sign at each level. I hope to own many types of weapons at the beginning, and my lives are full. I like the beautiful and bright scene, rather than dark colour design. '
- 'Still eyesight to prevent from feeling dizzy. Simple and small games with a sense of design. '
- 'Design. '

User Story & User Requirements

'A game is an artificial system for generating experieces. ' - Tynan Sylvester, *Designing Games*

'We create an artifact that a play interacts with, and cross our fingers that the experience that takes place during that interaction is something they will enjoy. ' - Jesse Schell, *The Art of Game Design*

I. Requirement acquisition

See questionnaire part.

II. Requirement analysis

Based on our questionnaire results, we can analyse user requirements for: (1) the theme / style of our game; (2) the topic / subject of our game; (3) user expectations for the settings / designs.

As we are doing agile software projects, we summarise user requirements from user stories.

Larger user stories are generally called **epics**. We can divide each epic into smaller user stories.

Full **User Epics-Break-Down document** please see:
https://github.com/FredaXYu/ASP_Group8/blob/main/midterm_proposal/User_story_requirements.doc
Snapshot:

Epics break down -

Epic:

Ability to speed up attacks if there is a level system, so you dont have to wait for the attack and defense to play out.

Break down story 1:

As a common user, I want to speed up the attacks. I know I am in low level (or didn't pay money). So I would somewhat like to play it if there's no speed-up buttons.
-> Requirements: (Functionality) Don't show the speed-up bottons for common users (low-level / didn't pay).

Break down story 2:

As a user who has paid the fee (or has played to high levels), I want to speed up the attacks, So that I can save time from unmeaningful attack animations.
-> Requirements: (Functionality) (1) Add a speed-up button (or skip button). (2) Set the visibility range – high-level players / have paid the fee. (3) Create speed-up animations.

Epic:

Naming squads or troops ability.

Break down story 1:

As a user, I want to see the names for squads or troops, So that I can recognise them.
-> Requirements: (Design) Show the names of squads / troops.

Break down story 2:

As a user, I want to rename the already existing squads or troops, So that I can feel the sense of personal characteristics.
-> Requirements: (User interface) Give users the ability to change the names.

. . .

Player Motivations:

- Master & creativity: We can leave space for users, such as only teaching them a little.
- Progression & power: We can encourage them when they progress to some milestones. Encourage them often. Admire them when they have powerful weapons. Reward them (with stars, weapons, characters) often.
- Complexity & puzzles: We can progressively add difficulties to the game.
- Elegance & aesthetics: We can draw beautiful shapes of maps, ex. in symmetry, with meaningful shapes, with harmonious colours and designs.

III. Requirements specification

Full document please see:
https://github.com/FredaXYu/ASP_Group8/blob/main/game_design/Complete/Requirements%20table%20v2.odt written by Alex.

Here is a snapshot:

0: Requirements for when the Software starts

ID	Requirement	Notes
0.0: Resource class		
0.0.0	The software shall have a ResourceObj class	This class is used to create an object that will store all the assets like images as well parameters for different parts of the game like tower damage and enemy movement speed.
0.1: Gameplay classes		
0.1.0	The software shall have a MapObj	This class is used to create the map

	class	object that will contain all the game elements that have position data like towers and enemies
0.1.1	The software shall have a GameStatsObj class	This class is used to create an object that will store game data that is not on the map like lives and money
0.1.2	The software shall have a HUDObj class	This class is used to create an object that display various information about the current level like current money and lives
0.2: Map classes		
0.2.0	The software shall have a OriginObj class	This class is used to create the object that enemies will appear from
0.2.1	The software shall have a DestinationObj class	This class is used to create the object that enemies will head towards
	. . .	
0.3: HUD classes		
0.3.0	The software shall have a TowerMenuObj class	This class is used to create an object that would display information about the current tower or house the tower shop
0.3.1	The software shall have a TowerMenuBuyObj class	This class is used to create an object that displays the tower shop
0.3.2	The software shall have a BuyTowerObj class	This class is used to create objects that represent the towers that could be bought
0.3.3	The software shall have a TowerMenuStatsObj class	This class is used to create an object that displays the stats of the currently selected tower and allows upgrading or selling of the tower
0.4: Settings class		
0.4.0	The software shall have a SettingsObj class	This class is used to create an object that displays different options like restarting the level
0.5: Level variable		
0.5.0	The software shall have a string variable called currentLevel	This records the name of the current or most recently played level
0.5.1	The software shall have a string variable called score	This is to record the latest score when the player wins or loses the level

0.6: Screen size variables		
0.6.0	The software shall have a float variable called screenHeight	This records how tall the game window is
	. . .	
0.7: State variables		
0.7.0	The software shall have have a boolean variable called isPlaying	This variable will determine whether or not to pause all game activity
	. . .	

1: Requirements for when the value of state is “initial load”

ID	Requirement	Notes
	While the value of state is “initial load”:	
1.0: State variable		
1.0.0	The value of isPlaying shall be False	
1.1: Outer dictionary		
1.1.0	The software shall have a ResourceObj object called Resource	
1.1.1	The software shall load all relevant data parameters to dictionary variables and add them to	The dictionary variables would be called: 1)levelData, 2) enemyData, 3) towerData, 4) bulletData 5) originData, 6) destData, 7) pathData, 8) siteData, 9) decoData. Some of these are 2D nested dictionaries with the innerkeys being the names of the assets and their respective values being their parameters.
1.2: Level data		
1.2.0	The software shall check whether each value of the inner dictionaries in levelData are the correct data type	The keys for levelData are 1) map, 2) wave, 3) money, 4) lives, 5) rest. The data types for the values should be 1) array for map, 2) list for wave, 3) integer for money, 4) integer for lives, 5) float for rest. Each entry in wave is a dictionary with 1) type, 2) quant and 3) rate. The data types of these are not checked here.
1.2.1	If any of the data types in levelData do not have the correct data type,	

	then the software shall end execution	
	. . .	
1.3: Enemy data		
1.3.0	The software shall check whether each value of the inner dictionaries in enemyData are the correct data type	The keys for enemyData are 1) name, 2) imageLink, 3) maxHP, 4) moveSpeed, 5) size, 6) reward. The data types for the values should be 1) string for name, 2) string for imageLink, 3) float for maxHP, 4) float for moveSpeed, 5) float for size, 6)integer for reward
	. . .	
1.4: Tower data		
1.4.0	The software shall check whether each value of the inner dictionaries in towerData are the correct data type	The keys for towerData are 1) name, 2) imageLink, 3) speed, 4) range, 5) cost, 6) upgradeKey, 7) sellAmount, 8) bulletKey. The data types for the values should be 1) string for name, 2) string for imageLink, 3) float for speed, 4) float for range, 5) integer for cost, 6) string for upgradeKey, 7) integer for sellAmount, 8) string for bulletKey
	. . .	
1.5: Bullet data		
1.5.0	The software shall check whether each value in the inner dictionaries in bulletData are the correct data type	The keys for bulletData are 1) imageLink, 2) size, 3) damage, 4) speed, 5) damageType, 6) AOE. The data types should be 1) string for imageLink, 2) float for size, 3) float for damage, 4) float for speed, 5) string for damageType, 6) float for AOE
1.5.1	If any of the data types in bulletData do not have the correct data type, then the software shall end execution	
	. . .	
1.6: Origin data		
1.6.0	The software shall check whether each value in originData are the	The only key for originData is imageLink which would have a

	correct data type	value of data type string
	. . .	
1.7: Destination data		
1.7.0	The software shall check whether each value in destData are the correct data type	The only key for destData is imageLink which would have a value of data type string
	. . .	
1.8: Path data		
1.8.0	The software shall check whether each value in pathData are the correct data type	The only key for pathData is imageLink which would have a value of data type string
	. . .	
1.9: Building Site data		
1.9.0	The software shall check whether each value in siteData are the correct data type	The only key for siteData is imageLink which would have a value of data type string
	. . .	
1.10: Decoration data		
1.10.0	The software shall check whether each value in decoData are the correct data type	The only key for decoData is imageLink which would have a value of data type string
	. . .	
1.11: Image elements		
	. . .	
1.11.1	When all the data types in all the dictionaries have been checked, the software shall load all relevant image elements to images	The names of these image elements would depend on what part of the game they are for, the name of the tile and what version of that image is requested. For example, if there are three versions of decoration tiles for grass, then the different image elements for grass would be named deco_grass_01, deco_grass_02 and deco_grass_03
1.11.2	The software shall check the data types of all image elements are all an image data type	
	. . .	
1.12: State change		
1.12.0	When all image elements have been confirmed to be the of an	

	image data type, the value of state shall be “main menu”	

2: Requirements for when the value of state is “main menu”

ID	Requirement	Notes
	While the value of state is “main menu”,:	
2.0: State variable		
2.0.0	The value of isPlaying shall be False	
2.1: Game name		
2.1.0	The software shall display the name of the game	
2.2: Start Game button		
2.2.0	The software shall display the “Start Game” button	
2.2.1	When the “Start Game” button is clicked on, the value of state shall be “level select”	This is when the mouse is within the bounds of the button and mouse button is clicked
2.3: Quit Game button		
2.3.0	The software shall display the “Quit Game” button	
2.3.1	When the “Quit Game” button is clicked on, the software shall end execution	This is when the mouse is within the bounds of the button and mouse button is clicked

3: Requirements for when the value of state is “level select”

ID	Requirement	Notes
	While the value of state is “level select”,:	
3.0: State variable		
3.0.0	The value of isPlaying shall be False	
3.1: Level buttons		
	. . .	
3.1.2	When the button for a level is clicked on, the value of state shall be “play”	This is when the mouse is within the bounds of the button and mouse button is clicked
3.1.3	When the button for a level is clicked on, the value of currentLevel shall be the name of the level	

3.2: Main Menu button		
3.2.0	The software shall display the “Back to Main Menu” button	
3.2.1	When the “Back to Main Menu” button is clicked on, the value of state shall be “main menu”	This is when the mouse is within the bounds of the button and mouse button is clicked

4: Requirements for when the value of state is “play”

ID	Requirement	Notes
	While the value of state is “play”,:	
4.0: State variable		
4.0.0	The value of isPlaying shall be True	
4.1: Gameplay objects		
4.1.0	The software shall have a MapObj object called Map	The starting parameter is the value of currentLevel which is a global variable
4.1.1	The software shall have a GameStatsObj object called GameStats	The starting parameter is the value of currentLevel which is a global variable
4.1.2	The software shall have a HUDObj object called HUD	

5: Requirements for the Map object

ID	Requirement	Notes
	While the state of state is “play”,:	
5.0: Initial parameter		
5.0.0	Map shall have a string parametercalled levelKey	
5.1: Initialise map		
5.1.0	Map shall have an array called map	
5.1.1	The value of map shall be the value of “map” in the (value of levelKey) dictionary in the levelData dictionary from Resource	To put it in code, map = (Resource.levelData[(value of levelKey)])[“map”]
5.2: Map dimension check		
5.2.0	The software shall check that map is a 2D nested array	The checking is done by getting the data type of all entries of the array. If all the entries of the initial array are also arrays, then the array has at least two dimensions. Next, check the data type of all the inner arrays as well. If any of them are arrays, then it has more than two dimensions and therefore will fail

		the check.
	. . .	
5.3: Map shape check		
5.3.4	The software shall check whether every inner array of map has the same length	
5.3.5	If some of the inner arrays of map have been found to be a different length, then the value of state shall be “main menu”	
	. . .	
5.4: Map elementscheck		
5.4.0	The software shall check that the elements in the inner arrays of map are all of string data type	This would require a nested for loop in order to traverse all entries
5.4.1	The software shall check that every inner element of map could be used to access an image element in images from Resource	There are two formats that this stringwould be found on map. The firstformat would be in the form of (asset type)_(asset name)_imageLink. To extract itwith code, the format would be Resource.((asset type)[(asset name)])[“imageLink”]. The second format would be (asset type)_(asset). To extract it with code, the format would be Resource.((asset type)[(asset name)]. For both, the result would beanother string that points to a specific image in images of Resource. Extraction of the components would be done similar to “tokenise” fromthe Object Oriented Programming module.
	. . .	
5.5: Map size		
	. . .	
5.5.1	The value of outer shall be the length of the outer array of map	
5.6: Tile size		
	. . .	
5.6.2	Map shall have a float variable called gridWidth	This is to determine how high each tile in the map would be on screen

	. . .	
5.7: Origin		
5.7.0	Map shall have a tuple of two integer variables called originLoc	If the language does not support tuple specifically, the size and data type stipulations could be enforced by checking for size and data types after creation.
5.7.1	The software shall search map for data that references the originData dictionary from Resource	In a simple map, there should only be one origin tile. As map is in order of how the game map is arranged, linear search would need to be done to ensure the entire array is searched. All the searching needs to do is determine if [asset type] is originData. If it is found, then it passes.
	. . .	
5.8: Destination		
5.8.0	Map shall have a tuple of two integer variables called destLoc	If the language does not support tuple specifically, the size and data type stipulations could be enforced by checking for size and data types after creation.
5.8.1	The software shall search map for data that references the destData dictionary from Resource	In a simple map, there should only be one destination tile. As map is in order of how the game map is arranged, linear search would need to be done to ensure the entire array is searched. All the searching needs to do is determine if [asset type] is destData. If it is found, then it passes.
	. . .	
5.9: Paths		
5.9.0	Map shall have an integer variable called pathCount	This is used to help naming when there are multiple instances of the same object
5.9.1	When Map is first created, pathCount shall be 0	
	. . .	
5.10: Building Sites		
5.10.0	Map shall have an integer variable	This is used to help naming when

	called siteCount	there are multiple instances of the same object
	. . .	
5.11: Decorations		
5.11.0	Map shall have an integer variable called decoCount	This is used to help naming when there are multiple instances of the same object
	. . .	
5.11.4	When an entry in map that references decoData has been found, the index of the inner array where it is found shall be saved to the first entry of decoLocTemp	The inner array could work like the x axis in that individual arrays are read horizontally
5.11.5	When an entry in map that references decoData has been found, the index of the outer array where it is found shall be saved to the second entry of decoLocTemp	The inner array could work like the y axis in that collections of arrays are read vertically
	. . .	
5.12: Waypoint list		
5.12.0	Map shall have a list of tuples of two float values called waypoints	
5.12.1	The software shall use the values of originLoc, destLoc and pathLoc to determine what the shortest route between the origin tile and destination tile traversing only path tiles would be with the aid of breadth first search	The code would be similar to the one found here: https://www.youtube.com/watch?v=KiCBXu4P-2Y. A few more variables and functions would be created in order for it to work. The final version would output the indices of the tiles on map that enemies would take would be added to waypoints.
	. . .	

6: Requirements for the GameStats object

ID	Requirement	Notes
	While the value of state is “play”,:	
6.0: Initial parameter		
6.0.0	GameStats shall have a string parametercalled levelKey	
6.1: Wave counter		
6.1.0	GameStats shall have an integer variable called waveCounter	This is used to keep track of what number the current wave is

	. . .	
6.2: Money		
	. . .	
6.2.1	The value of startMoney shall be the value of “money” in the (value of levelKey) dictionary in the levelData dictionary from Resource	To put it in code, startMoney = (Resource.levelData[(value of levelKey)])[“money”]
	. . .	
6.3: Score		
6.3.0	GameStats shall have an integer variable called score	
6.3.1	When GameStats is first created, the value of score shall be 0	
6.4: Lives		
6.4.0	GameStats shall have in integer variable called startLives	
6.4.1	The value of startLives shall be the value of “lives” in the (value of levelKey) dictionary in the levelData dictionary from Resource	To put it in code, startLives = (Resource.levelData[(value of levelKey)])[“lives”]
	. . .	
6.5: SpeedSettings		
	. . .	
6.5.1	When GameStats is first created, the value of speedSettings shall be 1	
6.6: Play speed		
6.6.0	GameStats shall have a float value called playSpeed	This determines the speed of the game but also takes into account whether the game should be paused
	. . .	

7: Requirements for the HUD object

ID	Requirement	Notes
	While the value of state is “play”,:	
7.0: Display values		
7.0.0	HUD shall display the value of lives from GameStats	
	. . .	
7.1: Settings button		
	. . .	
7.1.2	When the “Settings” button is clicked on, the value of state shall	

	be “settings”	
7.2: Speed Settings button		
	. . .	
7.2.2	When the “Speed Settings” button is clicked on and the value of speedSettings from GameStats is 0, the value of speedSettings from GameStats shall be 1	This is to toggle whether the game is moving forward in time or not
7.3: Tower menu		
7.3.0	HUD shall have a TowerMenuObj called TowerMenu	
7.4: Start button		
	. . .	
7.4.2	When the value of originState from Origin in Map is not “ready”, the “Start” button shall disappear	

8: Requirements for the Settings object

ID	Requirement	Notes
	While the value of state is “settings”,:	
8.0: State variable		
8.0.0	The value of isPlaying shall be False	
8.1: Resume button		
	. . .	
8.1.2	When the “Resume” button is clicked on, Settings shall be deleted	
8.2: Restart button		
	. . .	
8.2.4	When the “Restart” button is clicked on, the value of state shall be “play”	Since Map, GameStats and HUD don’t exist at this point and the game requires them, the software will create new copies of them.
8.3: Main Menu button		
	. . .	
8.4: Quit Game button		

9: Requirements for the Origin object

ID	Requirement	Notes
	While the value of state is “play”,:	

9.0: Initial parameters
9.1: Coordinates on screen
9.2: Display picture
9.3: Waves array
9.4: Check type
9.5: Check quant
9.6: Check rate
9.7: Wave dictionary
9.8: Type variable
9.10: Create speed
9.11: Quant
9.12: After creating
9.13: Rest state

10: Requirements for the Destination object

ID	Requirement	Notes
	While the value of state is “play”,:	
10.0: Initial parameters		
	. . .	
10.0.1	Destination shall have a tuple parameterof two floatscalled location	This would be the coordinate of this element in tiles. The x coordinate starts counting from the left and increasing towards the right and the y coordinate start counting from the top and increasing towards the bottom. The values need to be multiplied by gridWidth and gridHeight fromMap in order to display pic at the correct position
10.1: Coordinates on screen		
10.1.0	Destination shall have a float variable called x	This would be the x coordinate of the top left corner of Destination on screen
	. . .	
10.2: Display picture		
10.2.0	Destination shall display pic to the screen at position x, y with width and height being gridWidth and gridHeight fromMap respectively	

11: Requirements for the Path objects

ID	Requirement	Notes

	While the value of state is “play”,:	
11.0: Initial parameters		
11.0.0	Path shall have an image element parametercalled pic	This would be the image that occupies this tile
11.0.1	Path shall have a tuple parameterof two floatscalled location	This would be the coordinate of this element in tiles. The x coordinate starts counting from the left and increasing towards the right and the y coordinate start counting from the top and increasing towards the bottom. The values need to be multiplied by gridWidth and gridHeight fromMap in order to display pic at the correct position
11.1: Coordinates on screen		
11.1.0	Path shall have a float variable called x	This would be the x coordinate of the top left corner of Path on screen
11.1.1	The value of x shall be the value of the first element of location multiplied by the value of gridWidth fromMap	
	. . .	
11.2: Display picture		
	. . .	

12: Requirements for the Site objects

ID	Requirement	Notes
	While the value of state is “play”,:	
12.0: Initial parameters		
12.0.0	Site shall have an image element parameter called pic	This would be the image that occupies this tile
	. . .	
12.1: Coordinates on screen of top left corner		
12.1.0	Site shall have a float variable called x	This would be the x coordinate of the top left corner of Site on screen
12.1.1	The value of x shall be the value of the first element of location multiplied by the value of gridWidth fromMap	
	. . .	
12.2: Occupied state		
12.2.0	Site shall have a string variable	The value of this would be “empty”

	called occupation	if the building site is not occupied or it would be the key of a dictionary in towerData from Resource
	. . .	
12.3: Display picture		
12.3.0	When the value of occupation is “empty”, Site shall display pic to the screen at positions x, y with width and height being gridWidth and gridHeight fromMap respectively	
	. . .	
12.4: Active state		
12.4.0	Site shall have a boolean variable called active	This is used to determine which Site TowerMenu from HUD should focus attention on
12.4.1	When Site is first created, active shall be False	This means that by default, no Site is focused on.
12.4.2	When the mouse is clicked on the tile with Site, the value of active shall be true	
	. . .	
12.5: Bullet list		
12.5.0	Site shall have a list of BulletObj objects called bulletList	This is to keep track of all bullets that are on screen fired from this tile

13: Requirements for the Decoration objects

ID	Requirement	Notes
	While the value of state is “play”,:	
13.0: Initial parameters		
13.0.0	Decoration shall have an image element parametercalled pic	This would be the image that occupies this tile
13.0.1	Decoration shall have a tuple parameterof two floatscalled location	This would be the coordinate of this element in tiles. The x coordinate starts counting from the left and increasing towards the right and the y coordinate start counting from the top and increasing towards the bottom. The values need to be multiplied by gridWidth and gridHeight

		fromMap in order to display pic at the correct position
13.1: Coordinates on screen		
13.1.0	Decoration shall have a float variable called x	This would be the x coordinate of the top left corner of Decoration on screen
13.1.1	The value of x shall be the value of the first element of location multiplied by the value of gridWidth from Map	
	. . .	
13.2: Display picture		
13.2.0	Decoration shall display pic to the screen at position x, y with width and height being gridWidth and gridHeight from Map respectively	

14: Requirements for the TowerMenu object

ID	Requirement	Notes
	While the value of state is “play”,:	
14.0: Display nothing		
14.0.0	TowerMenu shall have a boolean variable called display	
14.0.1	When no BuildingSiteObj obejcts in siteList from Map have the value of active equal to True, the value of display shall be False	
	. . .	
14.1: Display something		
14.1.0	When one BuildingSiteObj obejct in siteList fromMap has the value of active equal to True, the value of display shall be True	
14.2: Menu objects		
14.2.0	When the value of display is True and the value of occupation of that Site is “empty”, TowerMenu shall have a TowerMenuBuyObj object called MenuBuy	
	. . .	

14.3: Delete menu objects		
14.3.0	When the value of display is False, the software shall delete MenuStats	
	. . .	

15: Requirements for the Enemy objects

ID	Requirement	Notes
	While the value of state is “play”,:	
15.0: Initial parameters		
15.0.1	Enemy shall have a string parametercalled enemyKey	This would help determine many of the other parameters that Enemy would have
	. . .	
15.1: Coordinates on screen		
15.1.0	Enemy shall have a float variable called x	
	. . .	
15.2: Extract picture		
15.2.0	Enemy shall have a string variable called imageLink	
15.2.1	The value of imageLink shall be the value of “imageLink” in the (value of enemyKey) dictionary in the enemyData dictionary from Resource	To put it in code, imageLink = (Resource.enemyData[(value of enemyKey)])[“imageLink”]]
	. . .	
15.3: Display picture		
15.3.0	Enemy shall have a float variable called size	
15.3.1	The value of size shall be the value of “size” in the (value of enemyKey) dictionary in the enemyData dictionary from Resource	To put it in code, size = (Resource.enemyData[(value of enemyKey)])[“size”]
	. . .	
15.4: Extract speed		
15.4.0	Enemy shall have a float variable called speedMaster	
15.4.1	The value of speedMaster shall be the value of “speed” in the (value of enemyKey) dictionary in the enemyData dictionary from Resource	To put it in code, speedMaster = (Resource.enemyData[(value of enemyKey)])[“speed”]

	. . .	
15.5: Extract waypoints		
15.5.0	Enemy shall have a listof tuples of two float variables called waypoints	This will store the different points Enemy will head towards on the way to Destination
	. . .	
15.7: Waypoints tuple		
15.6.0	Enemy shall have an integer variable called wayCount	This is to count how many waypoints have been traversed so far
	. . .	
15.7: Move to waypoint x		
15.7.0	When the value of the first entry of target is greater than the value of x by the value of w or greater, x shall increase by the value of speed	In other words, move right at the rate of the value ofspeed
	. . .	
15.8: Move to waypoint y		
	. . .	
15.9: Arrive at waypoint		
15.9.0	When the value of x is the same as the first entry of target and the value of y is the same as the value of the second entry of target, wayCount shall increase by 1	In other words, remove the entry at the top
	. . .	
15.10: Extract HP		
15.10.0	Enemy shall have a float variable called maxHP	
	. . .	
15.11: When enemy is defeated		
15.11.0	Enemy shall have an integer variable called reward	
15.11.1	The value of reward shall be the value of “reward” in [enemyKey] dictionary in enemyData dictionary in Resource	To put it in code, reward = (Resource.enemyData[(value of enemyKey)])[“reward”]
	. . .	
15.12: When enemy is buggy		

15.12.0	Enemy shall have an integer value called time	This is to get rid of an Enemy which is buggy and is stuck somewhere
	. . .	

16: Requirements for the Tower objects

ID	Requirement	Notes
	While the value of state is “play”,:	
16.0: Initial parameter		
16.0.0	Tower shall have a string parametercalled towerKey	This would help determine many of the other parameters that Tower would have
16.1: Coordinates of top left corner on screen		
	. . .	
16.1.3	The value of y shall be the value of y from Site multiplied by the value of gridHeight from Map	Site is referring to this object’s parent
16.2: Display picture		
	. . .	
16.3: Bullet count		
	. . .	
16.3.2	When the value of value of waveCount increases, the value of bulletCount shall be 0	
16.4: Extract range		
16.4.0	Tower shall have a float variable called range	
16.4.1	The value of range shall be the value of “range” in the (value of towerKey) dictionary in the towerData dictionary from Resource	To put it in code, range = (Resource.towerData[(value of towerKey])[“range”]
16.5: Extract bullet key		
16.5.0	Tower shall have a string variable called bulletKey	
16.5.1	The value of bulletKey shall be the value of “bulletKey” in the (value of towerKey) dictionary in the towerData dictionary from Resource	To put it in code, bulletKey = (Resource.towerData[(value of towerKey)])[“bulletKey”]
16.6: Tower state variable		

	. . .	
16.6.2	When the value of towerState is “ready” and the distance between Tower and an Enemy is equal to or less than range, towerState shall be “firing”	This compares the distance between it and all EnemyObj objects. Distance is calculated by the x and y coordinates of both, the difference between them and using the Pythagorean theorem on them
	. . .	
16.7: Attack speed		
16.7.0	Tower shall have a float variable called speedMaster	
16.7.1	The value of speedMaster shall be the value of “speed” in the (value of towerKey) dictionary in the towerData dictionary from Resource	To put it in code, speedMaster = (Resource.towerData[(value of towerKey)])[“speed”]
	. . .	

17: Requirements for the Bullet objects

ID	Requirement	Notes
	While the value of state is “play”,:	
17.0: Initial parameters		
	. . .	
17.0.2	Bullet shall have an integer parametercalled targetID	
17.1: Coordinates on map		
	. . .	
17.1.1	When Bullet is first created, the value of x shall be the value of the first entry of location	
	. . .	
17.2: Extract speed		
17.2.0	Bullet shall have a float variable called speedMaster	
17.2.1	The value of speedMaster shall be the value of “speed” in the (value of bulletKey) dictionary in the bulletData dictionary from Resource	To put it in code, speedMaster = (Resource.bulletData[(value of bulletKey)])[“speed”]
	. . .	

17.3: Extract damage type		
	. . .	
17.3.1	The value of damageType shall be the value of “damageType” in the (value of bulletType) dictionary in the bulletData dictionary from Resource	To put it in code, damageType = (Resource.bulletData[(value of bulletType)])[“damageType”]
17.4: Calculate target coordinates x		
17.4.0	Bullet shall have a float value called tarX	This is the x position of the target on screen
17.4.1	The value of tarXshall be the value of x of the Enemy object in enemyList fromOrigin fromMap that has the value of ID that is the same as the value of targetID	
17.4.2	When damageType is “AOE”, the value of tarXshall remainconstant	
17.5: Calculate target coordinates y		
17.5.0	Bullet shall have a float value called tarY	This is the y position of the target on screen
	. . . .	
17.6: Calculate displacement from target		
	. . .	
17.7: Calculate angle from target		
17.7.0	Bullet shall have a float value called angle	This is to get the angle between Bullet and the target so Bullet can go towards the direction of the target
	. . .	
17.8: Extract picture		
17.8.0	Bullet shall have a string variable called imageLink	
	. . .	
17.9: Extract size		
17.9.0	Bullet shall have a float variable called size	
17.9.1	The value of size shall be the value of “size” in the (value of bulletKey) dictionary in the bulletData dictionary from Resource	To put it in code, size = (Resource.bulletData[(value of bulletKey)])[“size”]
17.10: Calculate size on screen		
17.10.0	Bullet shall have a float variable	This is the width of Bullet on screen

	called w	
	. . .	
17.11: Display pictureon screen		
17.11.0	Bullet shall have a float variable called drawX	
	. . .	
17.12: Calculate velocity		
17.12.0	Bullet shall have a float value velX	
	. . .	
17.13: Calculate movement		
17.13.0	The value of x shall have the value of velXadded to it	
	. . .	
17.14: Extract damage		
17.14.0	Bullet shall have a float variable called damage	
17.14.1	The value of damage shall be the value of “damage” in the (value of bulletKey) dictionary in the bulletData dictionary from Resource	To put it in code, damage = (Resource.bulletData[(value of bulletKey)])[“damage”]
17.15: Calculate damage on single target		
17.15.0	When the values of disXand disYare 0 and the damageType is “single”, the HP value of the Enemy object in enemyList from Origin from Map with an ID value the same as targetID shall decrease by the value of damage	
17.16: Extract AOE range		
17.17: Calculate damage on AOE		
17.18:Delete bullet		

18: Requirements for the MenuBuy object

ID	Requirement	Notes
	While the value of state is “play”,:	
18.0: List of all possible towers		
18.0.0	MenuBuyshall have list of strings called towerList	This will contain the keys of dictionaries for all towers

	. . .	
18.1: List of all low level towers		
18.1.0	MenuBuy shall have a list of BuyTowerObjobjectscalled buyList	
18.1.1	MenuBuy shall add a BuyTowerObjobject for every element in initialList and name them after these entries	The starting parameter would be the element in initialList. For example, basic_00 = new BuyTowerObj(“basic_00”)
18.2: Display low level towers		
18.2.0	MenuBuy shall display all the elements in buyList	

19: Requirements for the MenuStats object

ID	Requirement	Notes
	While the value of state is “play”,:	
19.0: Current tower key		
19.0.0	MenuStats shall have a string variable called cTowerKey	
	. . .	
19.1: Current tower bullet key		
	. . .	
19.1.1	The value of cBulletKey shall be the value of “bulletKey” in the (value of cTowerKey) dictionary in the towerData dictionary from Resource	To put it in code, cBulletKey = (Resource.towerData[(value of cTowerKey)])[“bulletKey”]
19.2: Current tower name		
	. . .	
19.2.1	The value of cName shall be the value of “name” in the (value of cTowerKey) dictionary in the towerData dictionary from Resource	To put it in code, cName = (Resource.towerData[(value of cTowerKey)])[“name”]
	. . .	
19.3: Current tower attackdamage		
	. . .	
19.3.1	The value of uDamage shall be the value of “damage” in the (value of cBulletKey) dictionary in the bulletData dictionary from Resource	To put it in code, cDamage = (Resource.bulletData[(value of cBulletKey)])[“damage”]
	. . .	

19.4: Current tower range		
19.4.0	MenuStats shall have a float variable called cRange	This will contain the attack range of the current tower
	. . .	
19.5: Current tower attack speed		
19.6: Current tower AOE range		
19.7: Upgrade towerbutton		
19.8: Upgrade tower key		
19.9: Upgrade tower cost		
19.10: Upgrade tower bullet key		
19.11: Upgrade button glow or dim		
19.12: Upgrade tower name		
19.13: Upgrade tower attack damage		
19.14: Upgrade tower attack range		
19.15: Upgrade tower attack speed		
19.16: Upgrade tower AOE range		
19.17: Upgrade button clicked		
19.18: Sell amount		
19.19: Sell button		

20: Requirements for the BuyTower object

ID	Requirement	Notes
	While the value of state is “play”,:	
20.0: Initial parameter		
20.1: Extract bullet key		
20.2: Tower name		
20.3: Tower attack damage		
20.4: Tower attack range		
20.5: Tower attack speed		
20.6: Tower AOE range		
20.7: Tower cost		
20.8: Button glow or dim		
20.9: Buy tower		

21: Requirements for the value of state is “level complete”

ID	Requirement	Notes
	While the value of state is “level complete”,:	
21.0: State variable		
21.1: Display score		
21.2: Delete gameplay object		

21.3: Main Menu button
21.4: Replay button
21.5: Play Another Level button
21.6: Quit Game button

22: Requirements for the value of state is “game over”

ID	Requirement	Notes
	While the value of state is “game over”,:	
22.0: State variable		
22.1: Display score		
22.2: Delete gameplay objects		
22.3: Main Menu button		
22.4: Replay Level button		
22.5: Play Another Level button		
22.6: Quit Game button		

Iv. Requirements verification.

Correct? Consistent? Complete? Unambiguous? Operable? Testable? Necessary? Measurable? Traceable? We need to verify these items throught our development process with iterations.

V. Requirement change

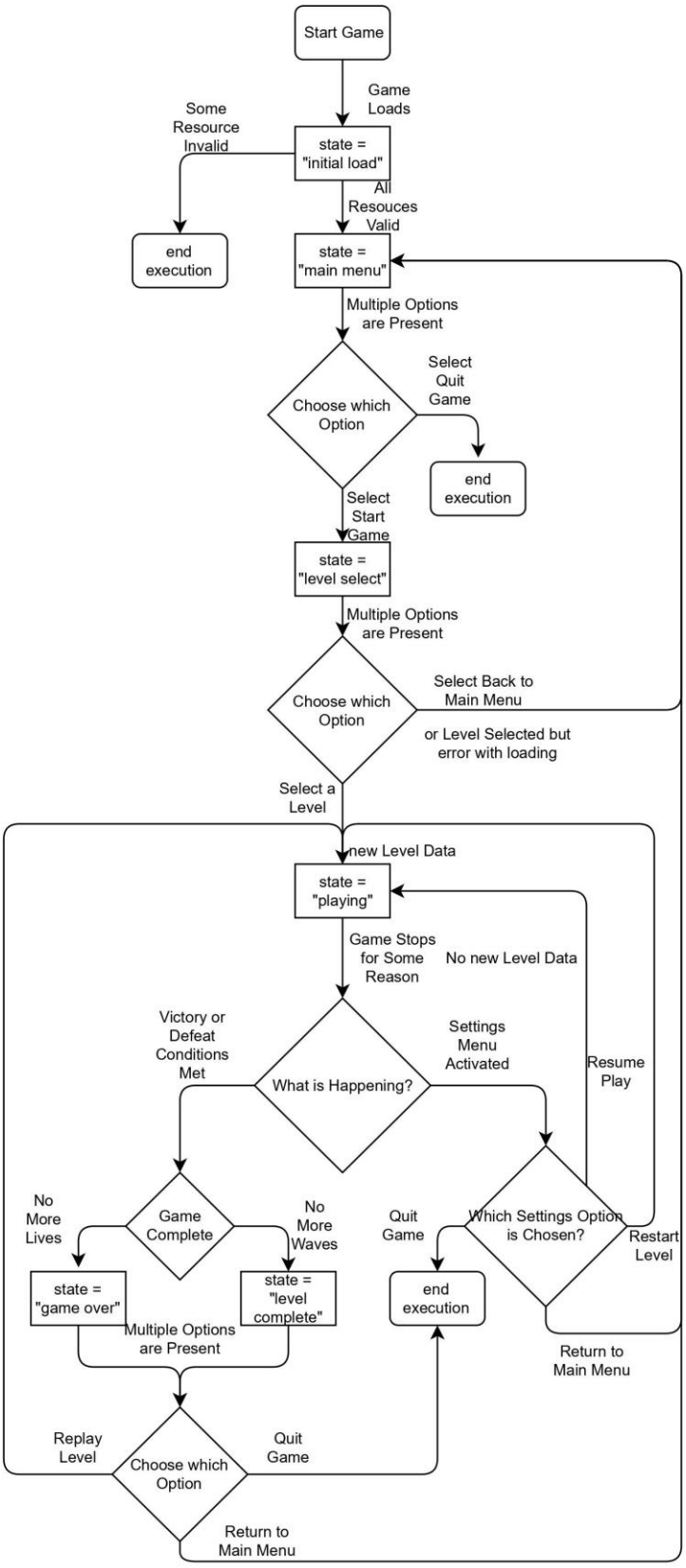
We are waiting for any change that may come from new user feedbacks.

Use Case & UML

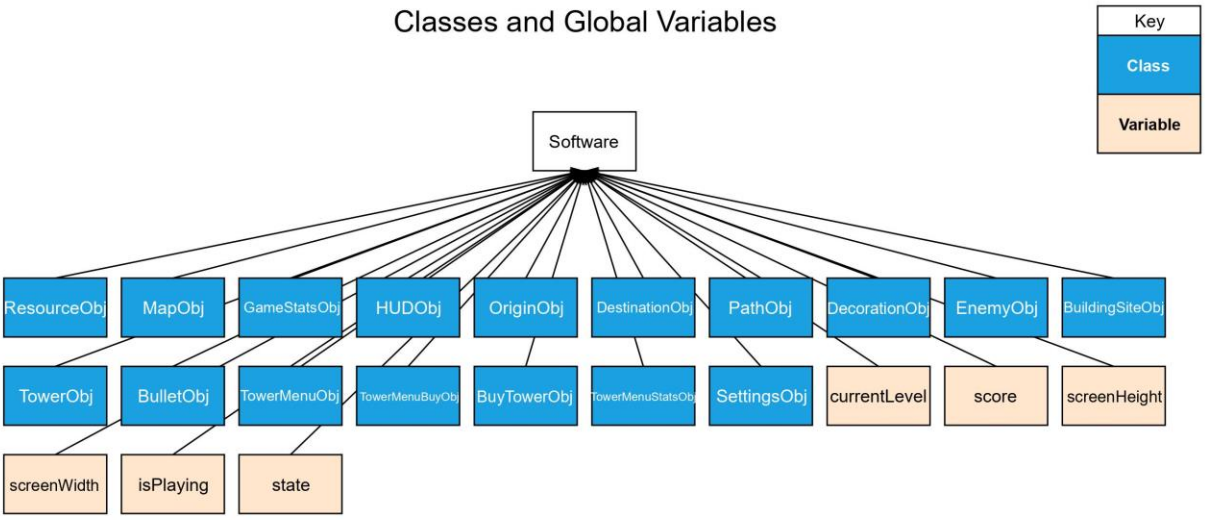
We have deeply discussed about which functionalities we should realise, and what subsystems each functionality should contain. Please refer to the document here:
https://github.com/FredaXYu/ASP_Group8/blob/main/game_design/Complete/Components.pdf

We (Alex) have written down some files that contain the basic elements and their relationships:

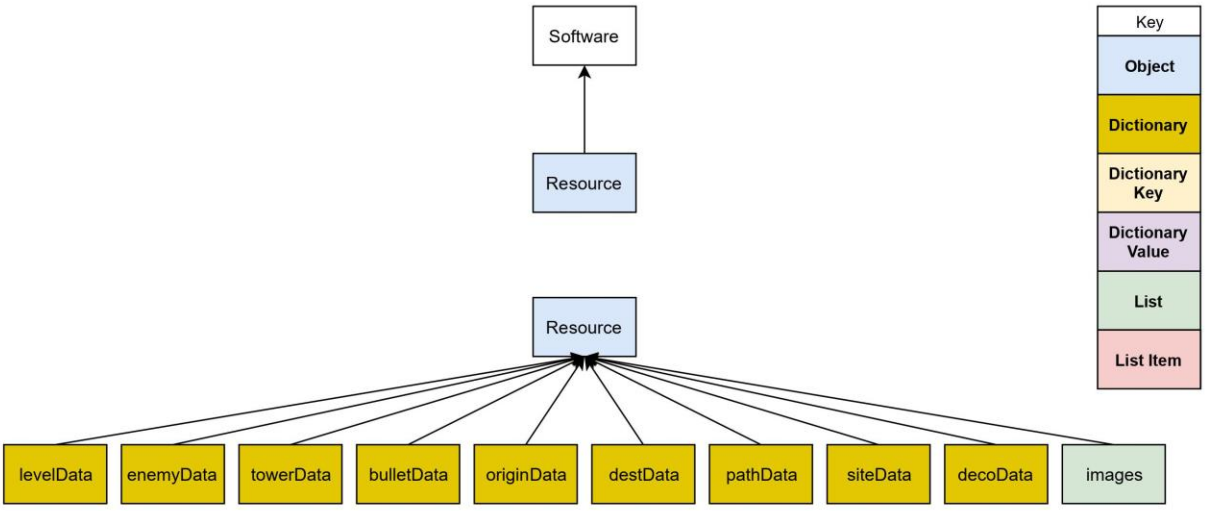
Structure Of All States Of The Software

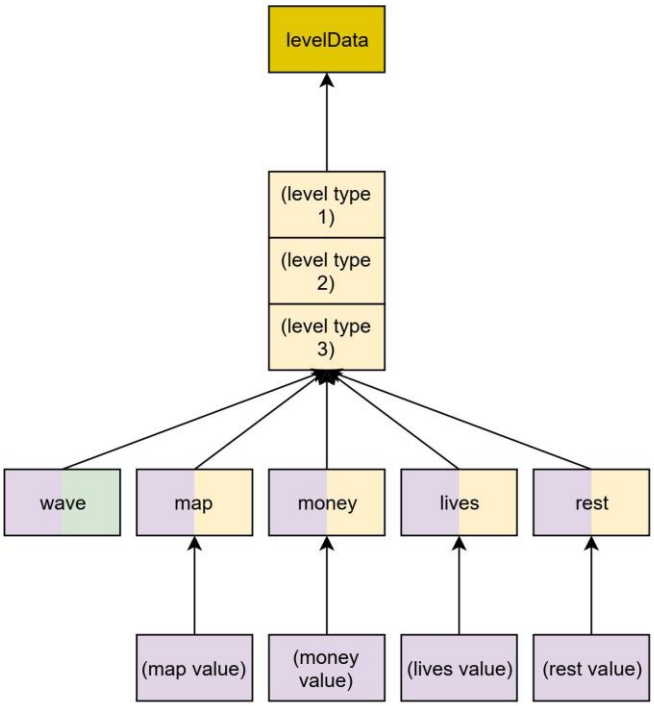


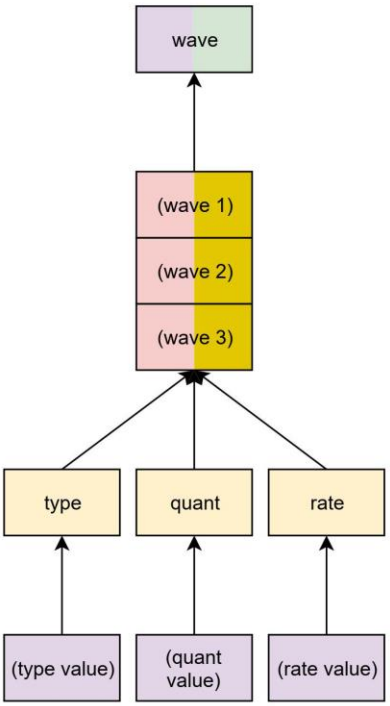
Classes and Global Variables

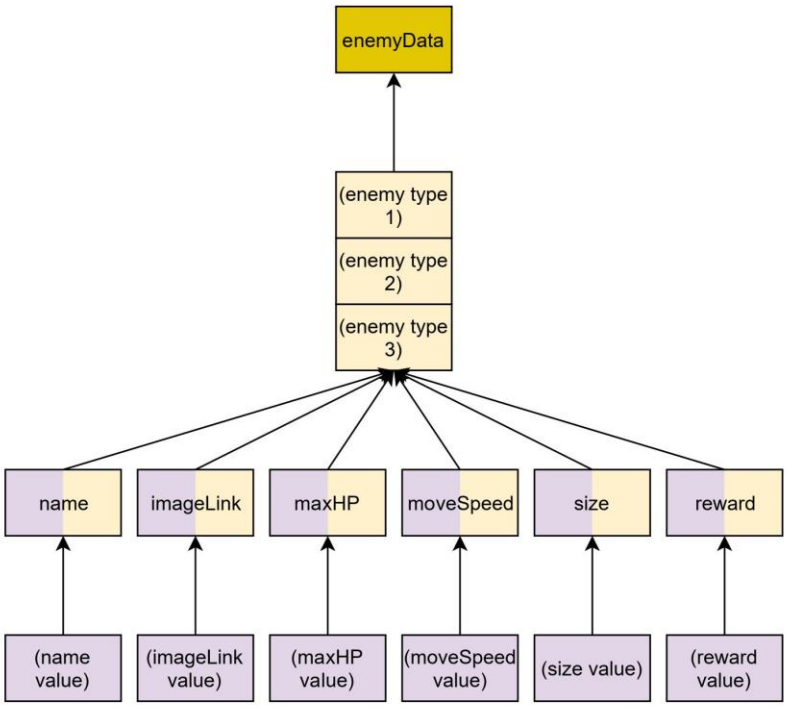


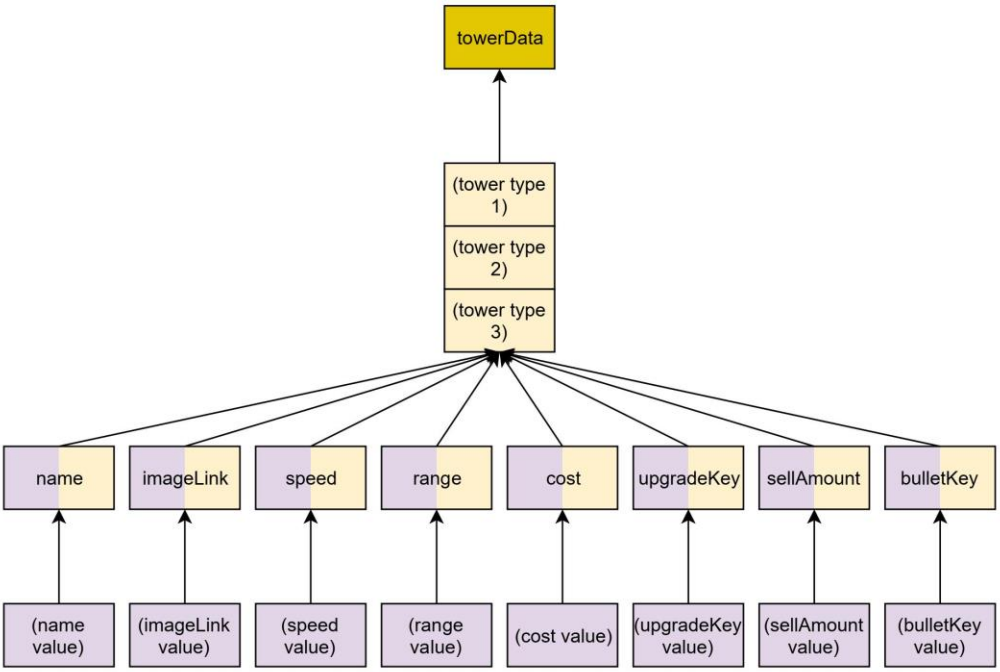
Structure of Resource

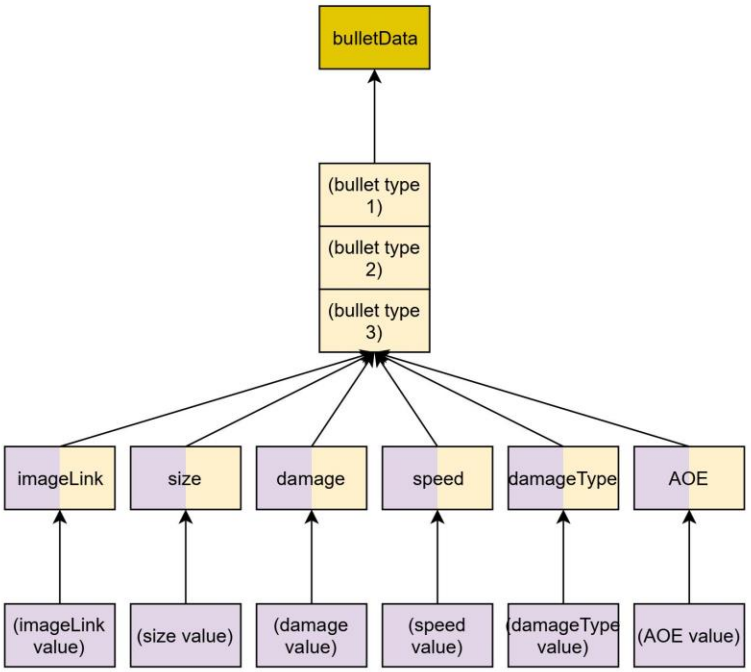


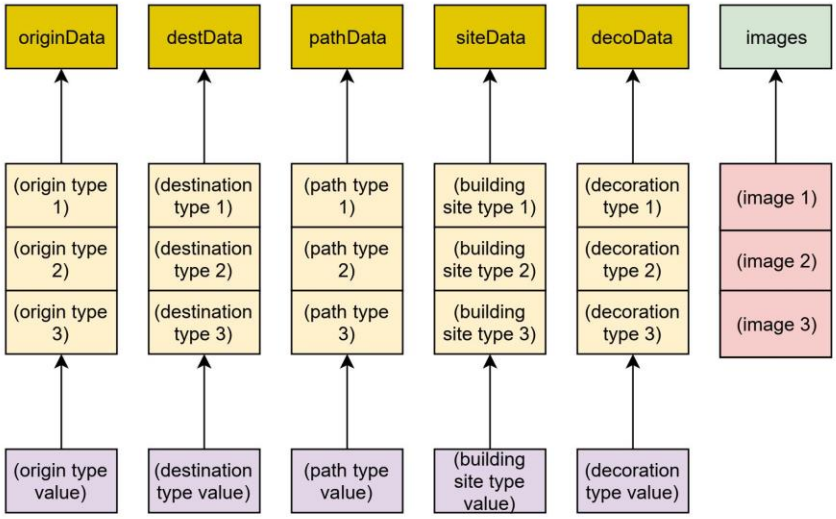

levelData
(level type 1)
(level type 2)
(level type 3)
wave
map
money
lives
rest
(map value)
(money value)
(lives value)
(rest value)


wave
(wave 1)
(wave 2)
(wave 3)
type
quant
rate
(type value)
(quant value)
(rate value)

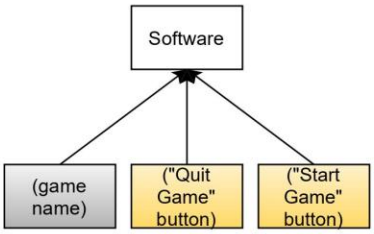

enemyData
(enemy type 1)
(enemy type 2)
(enemy type 3)
name
imageLink
maxHP
moveSpeed
size
reward
(name value)
(imageLink value)
(maxHP value)
(moveSpeed value)
(size value)
(reward value)

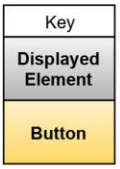

towerData
(tower type 1)
(tower type 2)
(tower type 3)
name
imageLink
speed
range
cost
upgradeKey
sellAmount
bulletKey
(name value)
(imageLink value)
(speed value)
(range value)
(cost value)
(upgradeKey value)
(sellAmount value)
(bulletKey value)


bulletData
(bullet type 1)
(bullet type 2)
(bullet type 3)
imageLink
size
damage
speed
damageType
AOE
(imageLink value)
(size value)
(damage value)
(speed value)
(damageType value)
(AOE value)



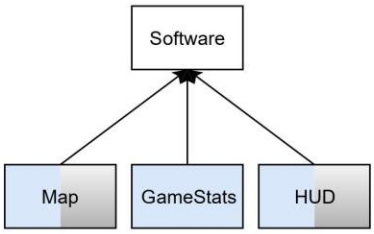
Software When state="main menu"

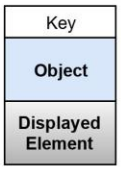




Note: This does not include global variables and classes.

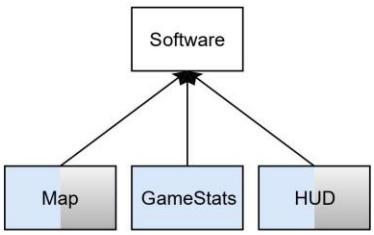
Software When state="level select"

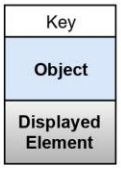




Note: This does not include global variables and classes.

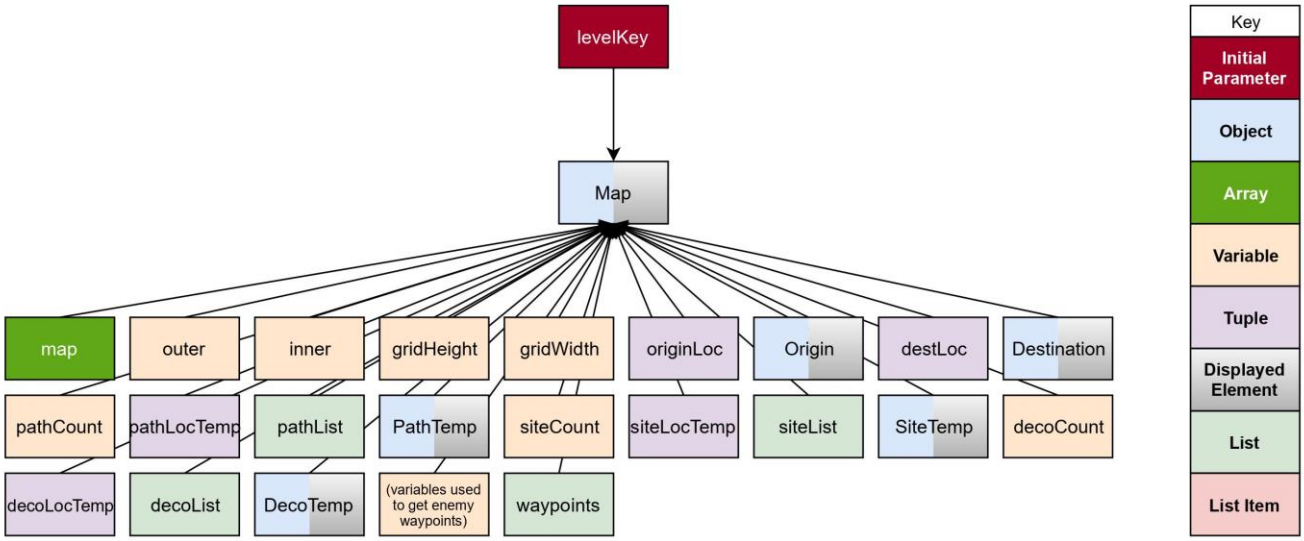
Software When state="play"

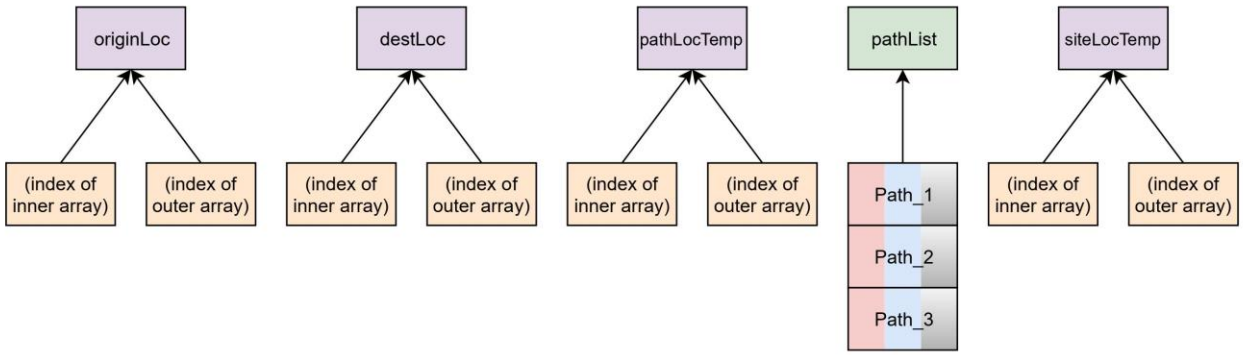


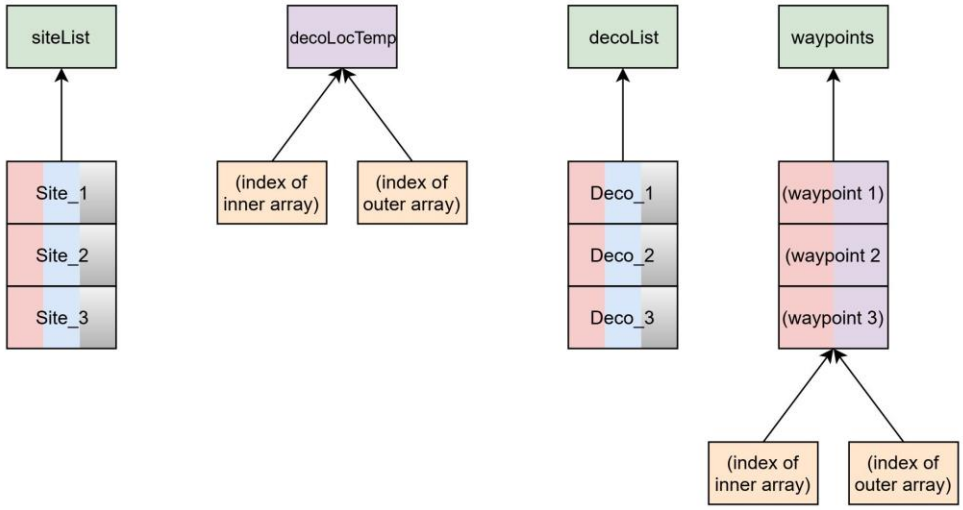


Note: This does not include global variables and classes.

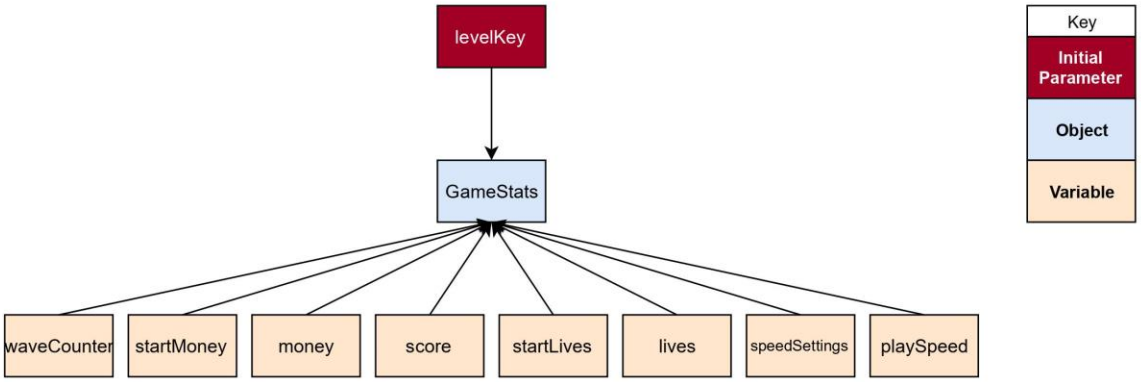
Structure of Map



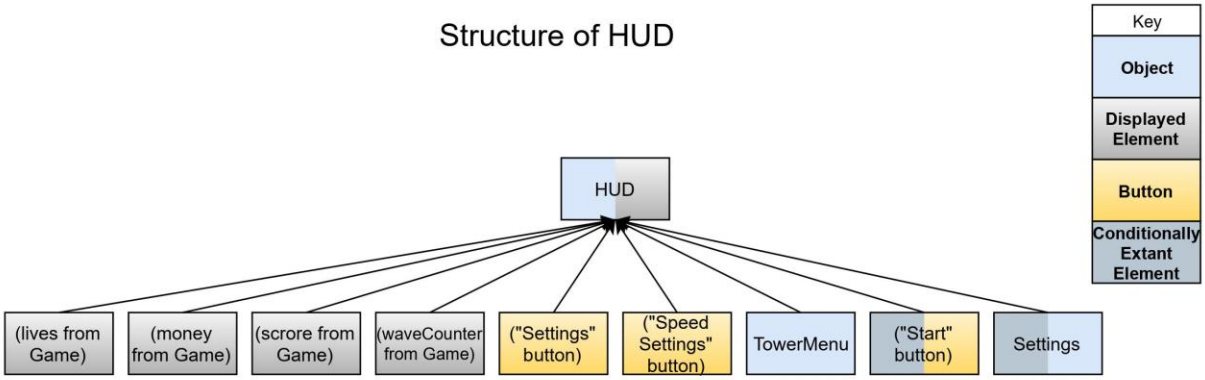




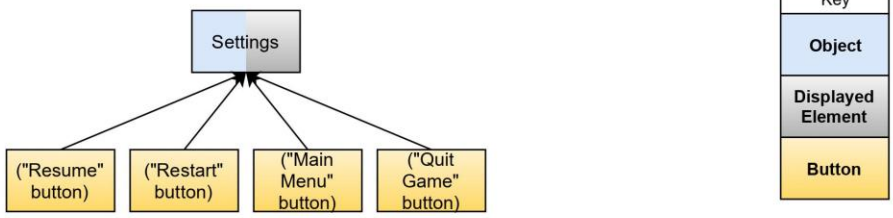
Structure of GameStats



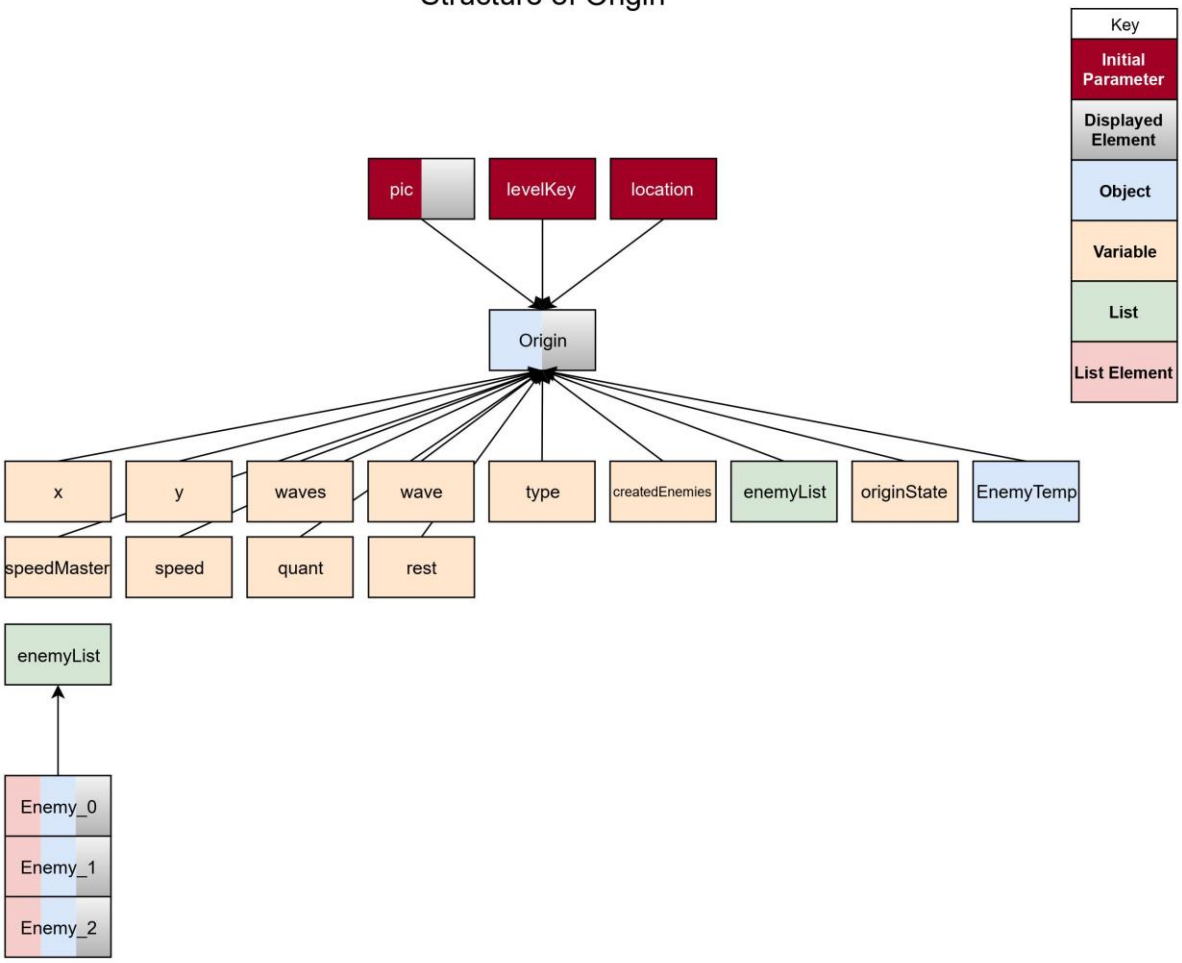
Structure of HUD



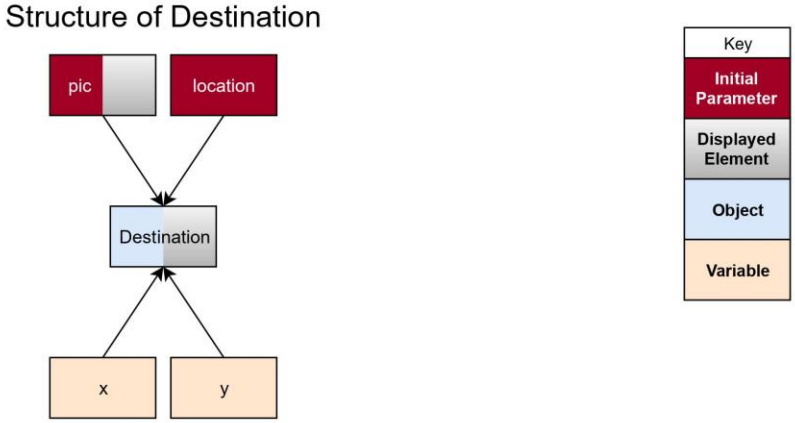
Structure of Settings



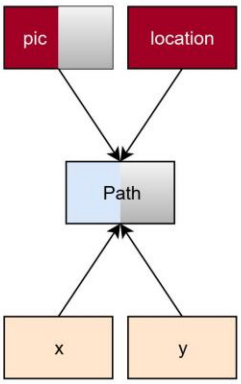
Structure of Origin

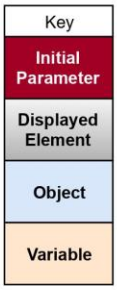


Structure of Destination

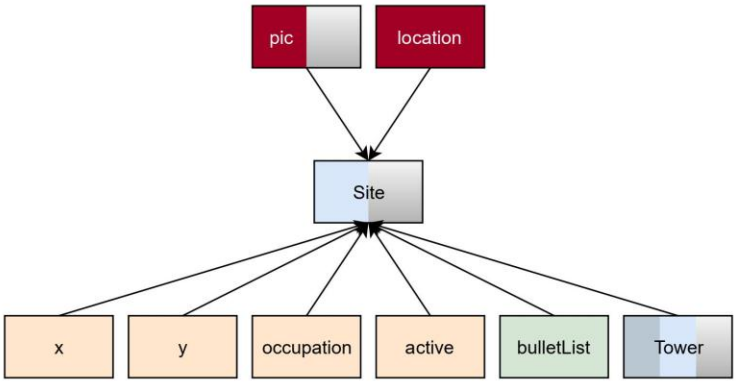


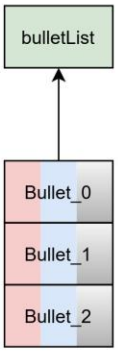
Structure of Path

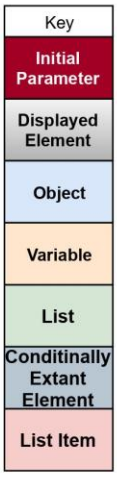




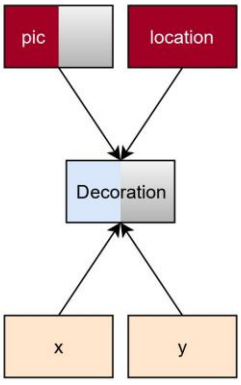
Structure of Site

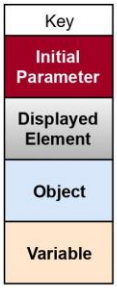




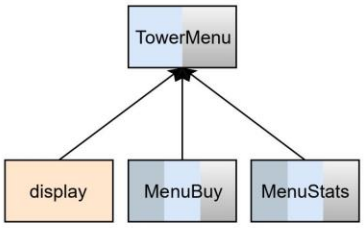


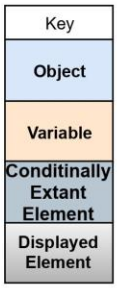
Structure of Decoration



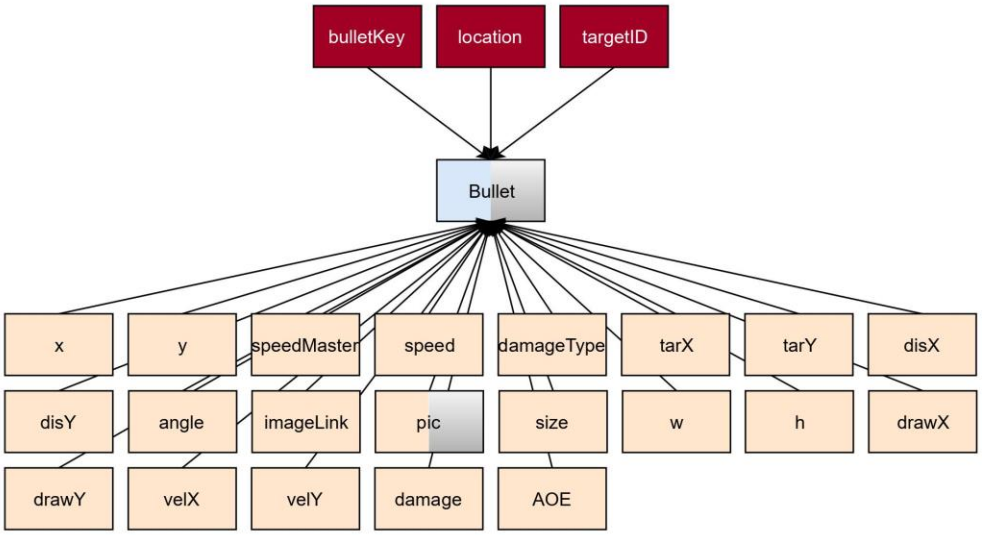


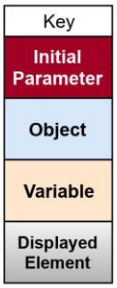
Structure of TowerMenu



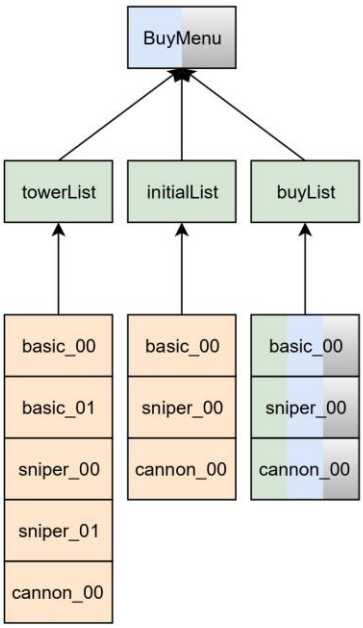


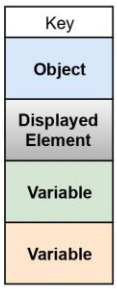
Structure of Bullet



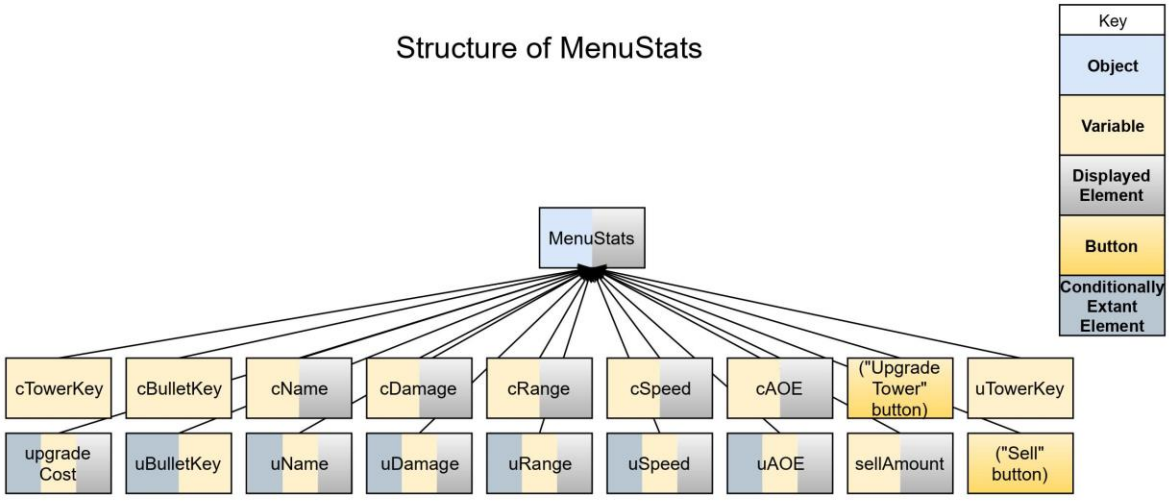


Structure of BuyMenu

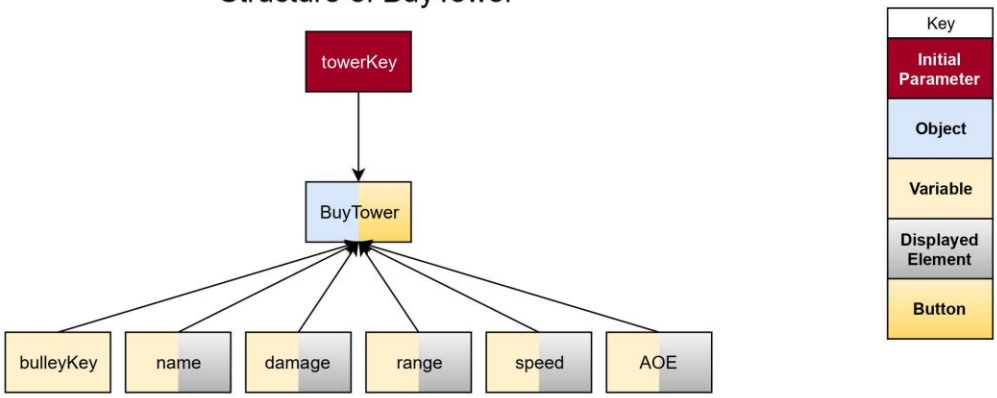




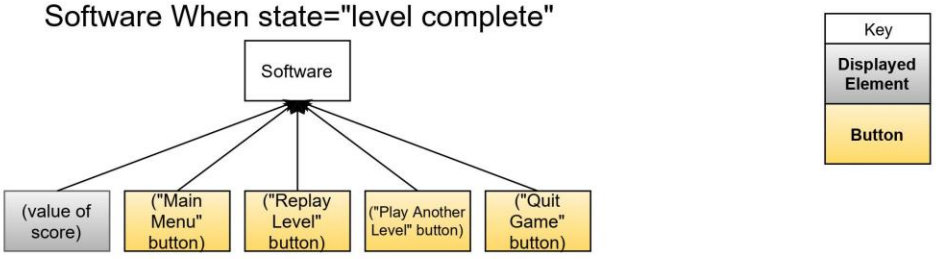
Structure of MenuStats



Structure of BuyTower



Software When state="level complete"



Software When state="game over"

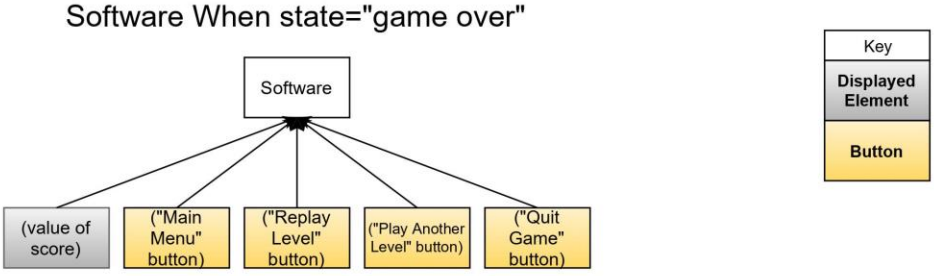
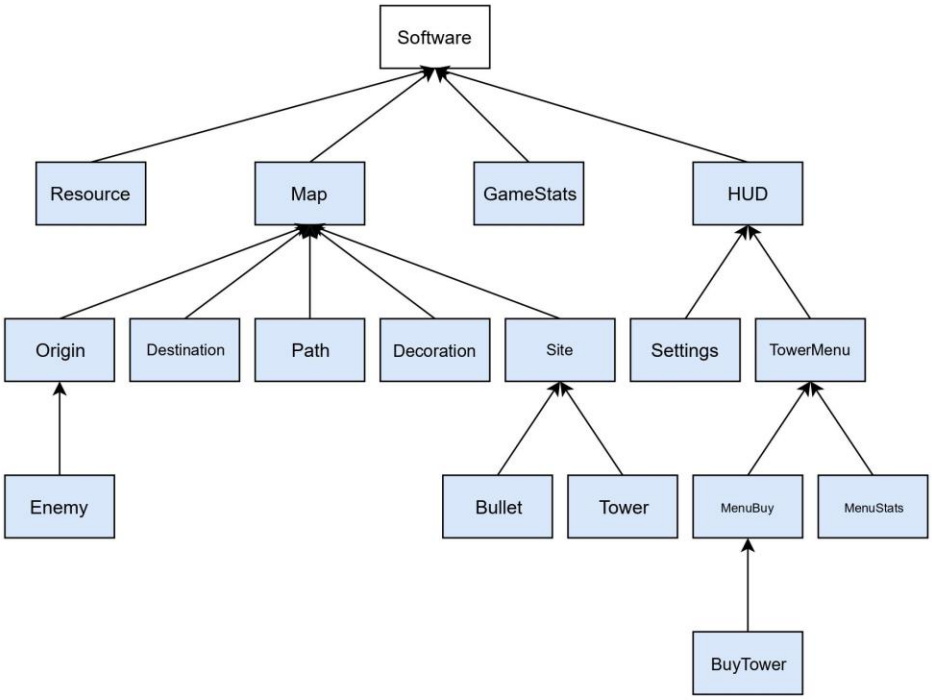


Diagram Of Where Different Objects Would Be Found In The Software



*Note: 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 All Objects That Read Data From Resource

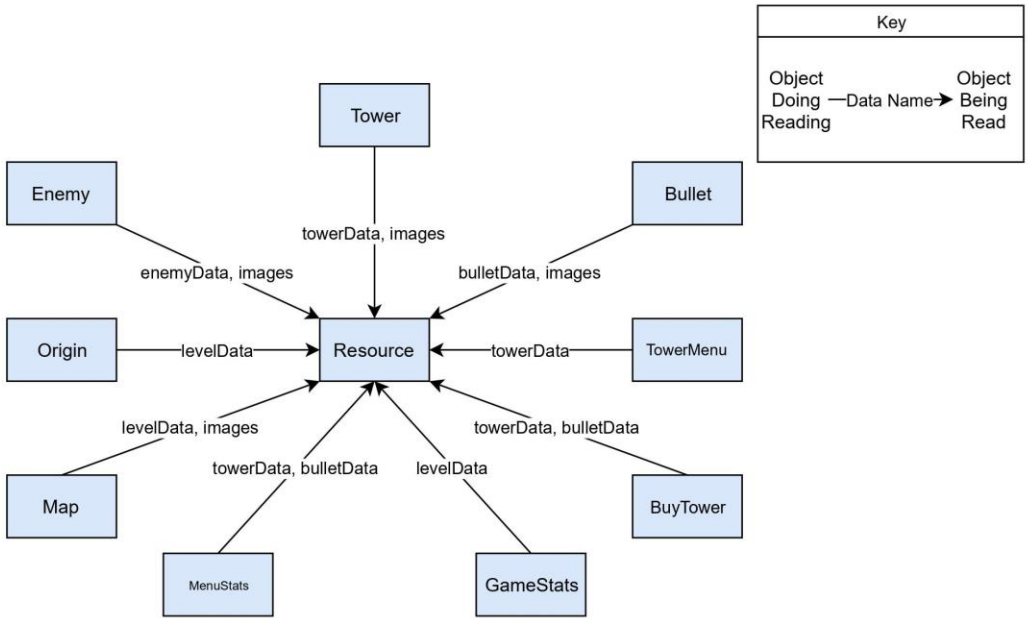
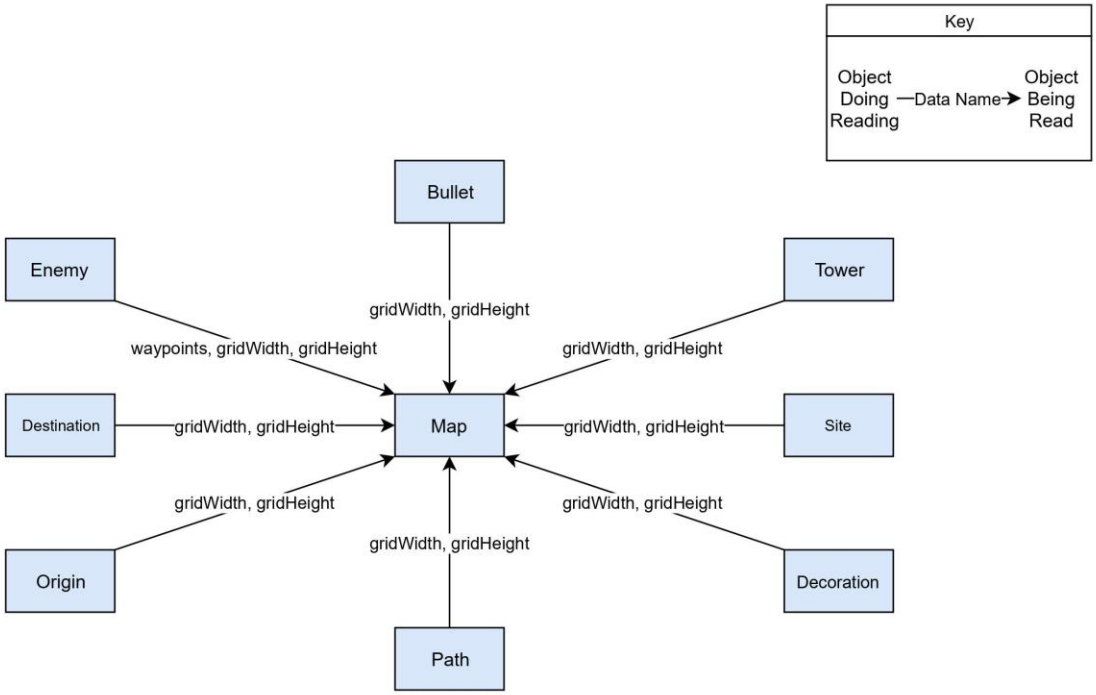


Diagram Of All Objects That Read Data From Map



Diagrams Of All Objects That Read Data From And Write Data To Origin

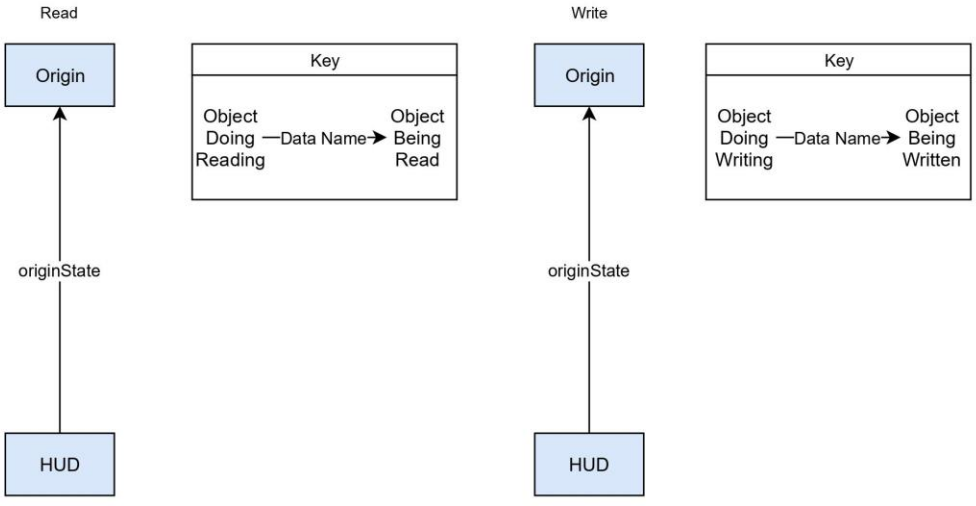


Diagram Of All Objects That Read Data From Site

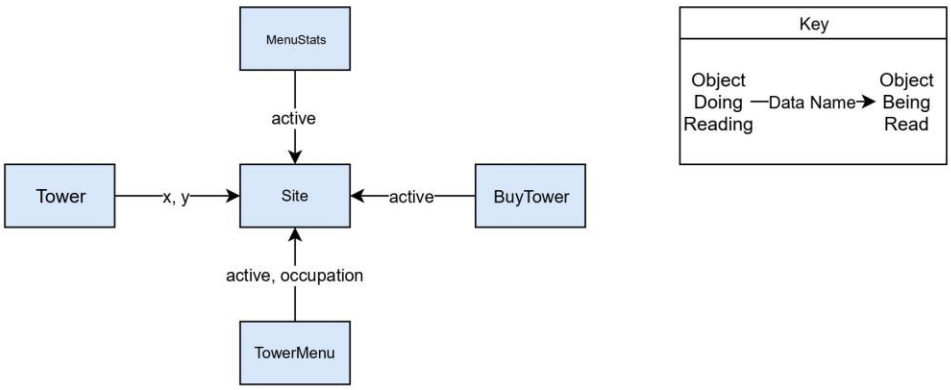
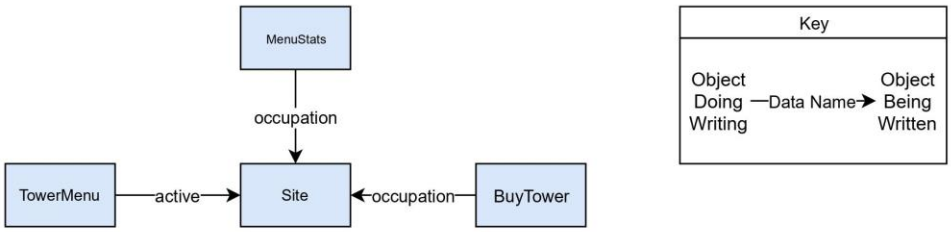
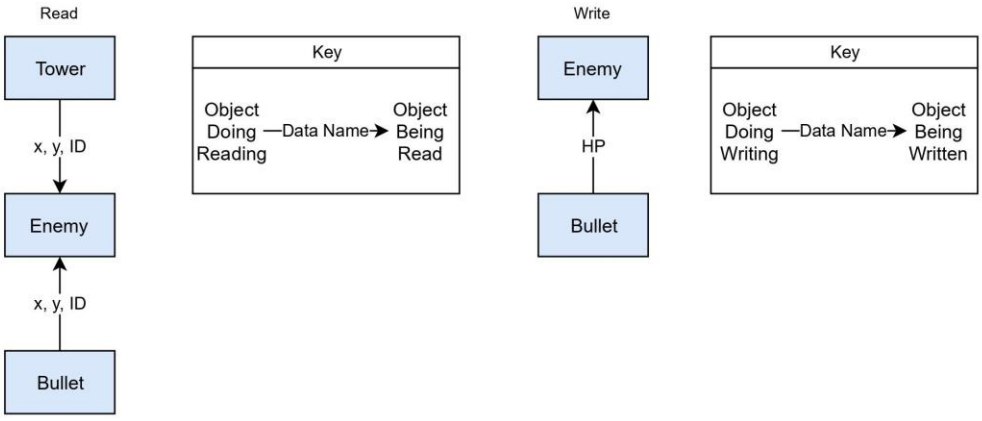


Diagram Of All Objects That Write Data To Site

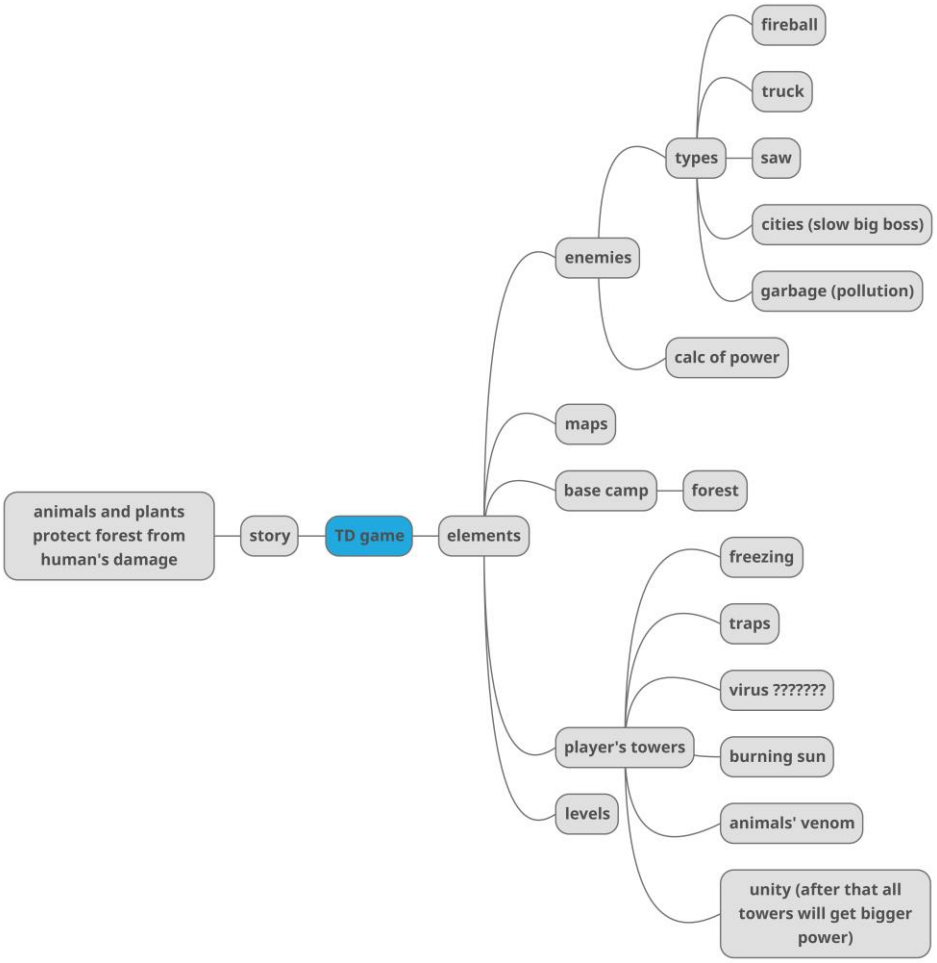


Diagrams Of All Objects That Read Data From And Write Data To Enemy



Prototyping

Xiaoyun (Freda) has designed an environment-protection TD game idea prototype. Here is her simple idea:



Make a story in the forest (the base camp) to help animals and plants to fight against humans' invasion.

The aim is to educate people to avoid the enemy's behaviors, and to educate them that fight against nature is not the long-term method. The only way is to embrace each other and then the Earth can smile.

- **Enemies?**

Human's methods:

Saw, trucks, fires (smoke butts), disposable tableware, polluted water, noise, plastics, etc.

There could be random moment when the enemy (human-beings) discovers his conscience, then they give up their intrusion, and embrace all of our animals and plants. And then, the whole forest is happier and happier. The level is passed automatically. This could be the best outcome in each level.

- **Towers?**

Animals and plants' solutions:

Utters' dams, bear's shouting, sun baking, freezing, twining of vines, heavy rain, typhoon, earthquakes, viruses, etc.

(It's a bit strange since these bad weathers are added as player's weapons...)

- **Expectations:**

1. A rush button, if the user is confident that his already settled towers are enough.
2. Should set a screen-using time, set the max amount as 30 min. After 30 min, the screen pop up a message: 'Time limit. Please stand out and take some exercises. Love your eyes, since all our Nature loves you. '
3. Lives: We can set our base camp as having 10 lives (it can bear with enemies' attacks for 10 times, gradually it gets shorter). When it's 0, the forest dies.
4. Money: We can create links to users' payment methods but it's too complex. We can set virtual money, at first give our user 1000 coins, then when they pass a level, award them 20 coins. Coins can be used to buy towers.

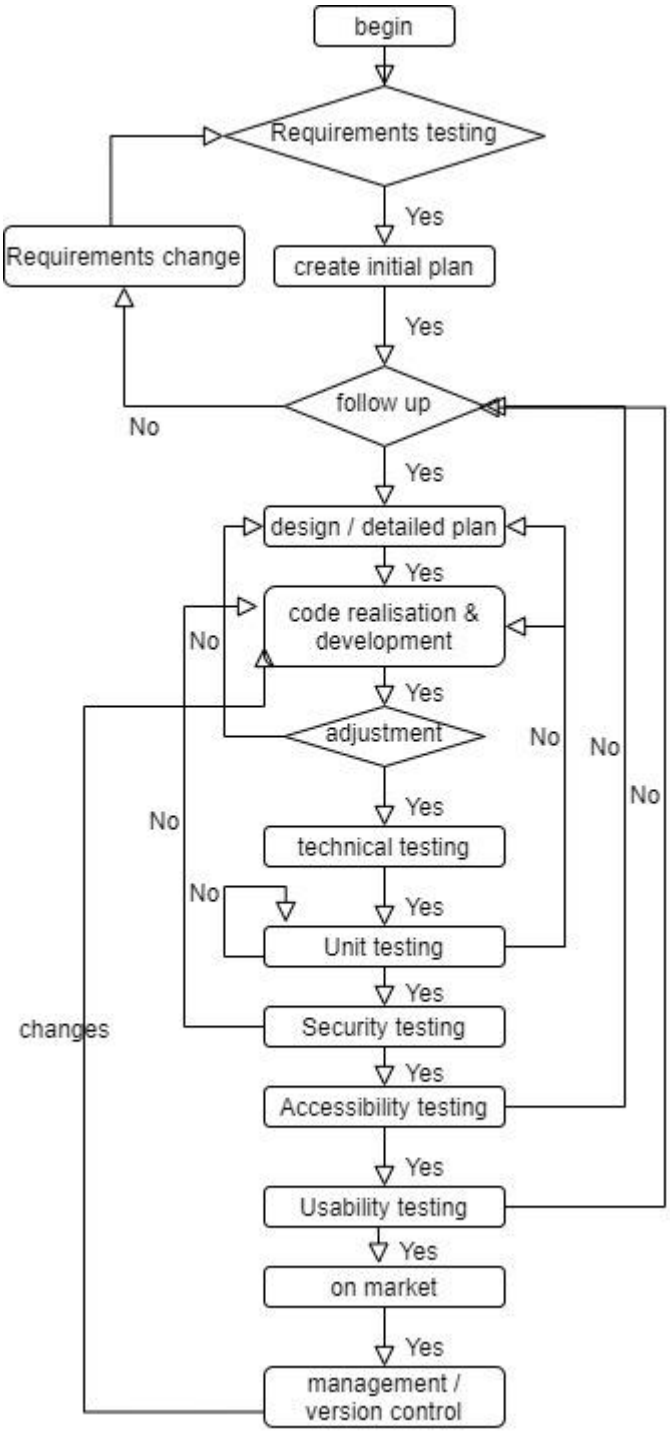
Strengths: (1) it has a complete story; (2) all the characters are similar to real life; (3) it's educational; (4) it's cutting-edge since nowadays people just need environmental protection sense to deal with the COVID-19 virus; (5) the story is simple and understandable.
Weaknesses: (1) what are the weapons for each tower? (2) can it be made a rival theme since the rival between humans and animals are not what we are proposed to teach?

While Jeremy has written down the initial codes:
https://github.com/FredaXYu/ASP_Group8/tree/main/prototype/Jeremy_cm2020_group8-main/cm2020_group8-main

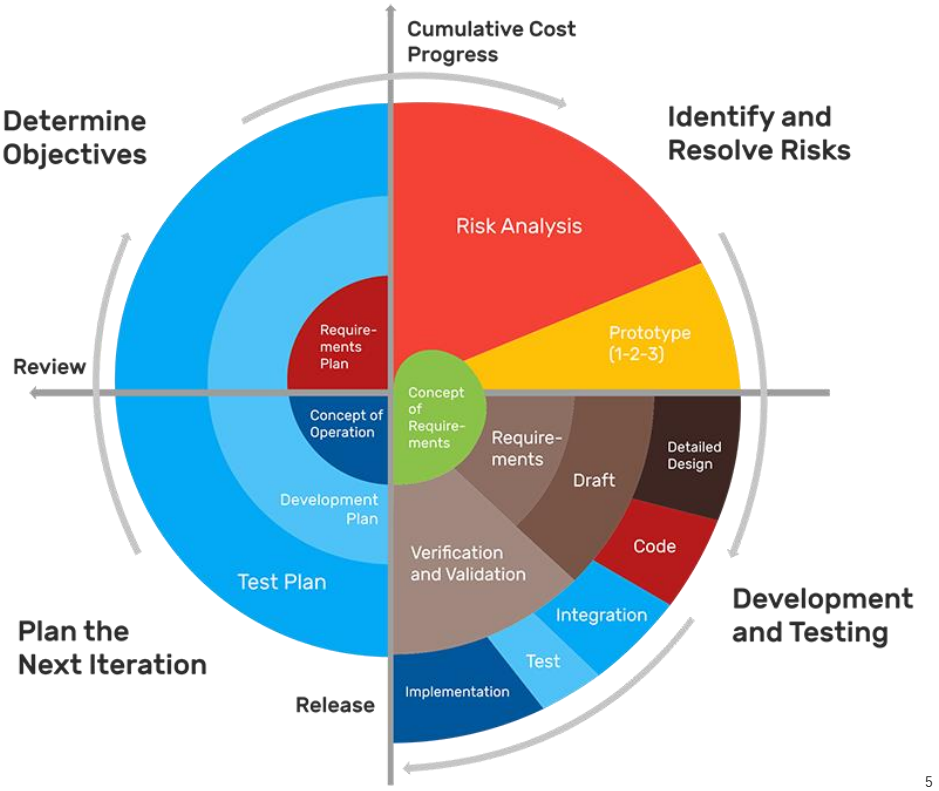
Strengths: (1) now we have a template code to work on; (2) the structure is more obvious than the specification once written down in codes.
Weaknesses: (1) it lacks stories and characters; (2) there are still more components that should be added here according to Alex's document;

Development Process (with Iterations and Expectations)

Here is a flowchart containing all the processes in a software development experience:



Alternatively, if we MUST adhere to one of the common development process models, then the **spiral model** fits best. However, we prefer to replace the 'Risk Analysis' part with 'User Needs Analysis', since our game project doesn't have many risks to resolve, but largely rely on the market (users).



[5]

As we are doing agile projects, we welcome any sudden changes.

Assumption Tests

User tests:

An Agile project should be utilizing User testing during the production and close to the end of the production cycle. Doing a mid-production test allows for any sudden course corrections that are needed based on player feedback.

While conducting these user tests, we should be looking for gain insight into to answer 4 main questions about our game:

1. To see if our **players enjoy** the main game mechanics of placing defenses to stop the swarm
2. To understand our users **first impressions** of theme of the game. Do they enjoy the artwork/audio?
3. To help gauge **if the levels are too difficult or too easy**. This will help us tune the levels and progression to enhance our players enjoyment.
4. To learn how our players will **interact** with the game. Are they able to easily navigate the User Interface? Are there any confusing menus which we can improve upon?

To do this, we will need to

1. Create a **distributable build** of the game. For our purposes, this can be as simple as a link to a

[5] Picture origin: https://techreceptives.com/tr_website/static/src/images/company/development_methodology/spiral-methodology.png

playable version of our game in a browser, or a zip file that the users can download that can be secured behind a simple password webpage.

2. Encourage people to participate in our user test through **sharing on social media**. Any interested participants would be sent the link to the game to test.
3. Send out a **survey** to be completed after playing the game to all our testers. We will want to gain information about both their gaming habits (if they are "hardcore" or "casual" gamers), as well as their thoughts on our game and any feedback to help us improve the experience.
4. After receiving the survey **feedback**, we will analyze the results. This will help us understand if we need to make any major changes to our game before we are finished. Any feedback that we wish to address can be brought into the next iteration cycle.

Accessibility testing:

According to IEEE dictionary, 'accessibility is the extent to which product system, services, environments, and facilities can be used by people from the population with the **widest range** of characteristics and capabilities to achieve a specified goal in a specified context of use'. There are some features:

Perceivable: Make sure (1) the platform or software we are using is realiable enough so that end users can easily find our game, then download it, then open it; (2) The pictures can be loaded; if cannot, then show a summary; (3) the sounds and music can be played; (4) it's acting in a normal speed; (5) the signt will not make people feeling dizzy; (6) make sure our project is legal locally (content, platform, etc.), so users can visit our product.
Operable: Make sure (1) once pressed certain mouse or keys, it will trigger specific operations; (2) the game would less likely to be stuck, and once stuck, it shows proper hints; (3) all the operations are user-friendly and reasonable.
Understandable: Make sure (1) users can understand the main story we are trying to tell them; (2) they understand the functionalities of all the buttons; (3) we must make tutorials once introducing a new thing (tower, weapon, enemy, map, level, button, etc.).
Robust: We want our project to work well even when it meets errors. Make sure (1) we write exception handling codes for each potential error, so that users will not be confused when they meet an error, and even they can report error to us; (2) use control flow to consider all the cases comprehensively, so that we can handle all the states of our project; (3) print out meaningful error messages; (4) keep our project running in spite of there are error messages; (5) how to terminate – return to home page, or pop out error messages then call exit.

Usability testing:

According to ISO 9241-11, 'Usability' is 'the extent to which a system, product, or service can be used by **specified users** to achieve specified goals with effectiveness, efficiency, and satisfaction in a specified context of use'.

We consider to obey **Nielson's 10 usability principles**:

1. Visibility of system **status**: We should: (1) print out the menu at the bottom or on the sidebar at all times;

(2) indicate where users are; (3) for each page, we always have the home button; (4) show the main map containing all unlocked levels and future levels; (5) perhaps will provide a 'treasure box' containing all the unlocked items (towers, enemies, weapons, etc.) and future ones.
2. Match between system and the **real world**: We should: (1) avoid making any abstract characters (since they would be hard to memorise); (2) choose a good story as our main clue; (2) carefully choose main characters as our towers, attack methods, and enemies. (3) make the background of each level more similar to the real world; (4) for the icons of each functionality (ex. sound, help, menu, home), we should draw proper shapes so that users can easily identify the functionality of each.
3. User control and **freedom**: We should give users the freedom to: (1) control the sound, music; (2) maybe customise the skins; (3) choose among types of towers; (4) choose which level to play among unlocked levels; (5)
4. **Consistency** and standards: (1) should check whether we use the **same** word to represent the specific item all the times; (2) should use the same style for all the buttons; (3) should not draw different shapes for the same item (prevent from 'character evolution', or, indicate clearly which state the character is in).
5. Error prevention: (1) delete error-prone conditions; (2) show users with a confirmation option before it goes to the error-prone conditions.
6. **Recognition** rather than recall: users don't need to memorise anything. We should (1) always show the names of towers; (2) before changing any weapon, show the name; (3) draw enemies indeed seems like evil characters; (4) draw our defense part indeed seems like kind-hearted characters; (5) emphasize the main troop that users should protect for; (6) always show the button 'treasure box' or 'character map' so that they can look for which character they are facing with; (7) show the menu all the time.
7. **Flexibility** and efficiency of use: (1) for inexperienced users, we show them tutorials; (2) for experienced users, we tailor the tutorials, and even skip some of the fighting scenes; (3) identify inexperienced and experienced user groups, by using account system and save each one's progress to our server.
8. Aesthetic and minimalist **design**: (1) for the main-story-introduction animation, only show the relevant information, and delete irrelevant names or actions; (2) don't introduce unmeaningful characters – develop deeply for each character; (3) don't make too many buttons – users may get lost in the menu; (4) don't overemphasize the background; (5) always emphasize the main information – don't draw things too messy.
9. **Help** users to recognise, diagnose, and recover from errors: (1) give users our contact email so that we can receive feedbacks and make updates; (2) if there are errors, make sure we show users a pop-up box with what error it is, and the solutions (and our contact email).
10. Help and **documentation**: We should: (1) print out the main menu in the home page; (2) always give users the 'Help' button whenever they are in any page; (3) summarise all the help functions into one complete document; (4) add a search functionality for the help document.

Evaluation Techniques

Current state:
(1) Initial codes are written down. Basic elements are constructed. Needs to expand and formed into details.
(2) User requirements have been collected, but we would expect more feedbacks.
(3) We must **learn to write more codes** for our TD game. Since we are beginners, we must learn from experienced people, or else our project could fail.

Benefits for what have done:

(1) We have written down an exhaustive UML file, including all the required elements in each level, and their complete interactions.
(2) We have developed a complete process of questionnaire gathering and data analysis.
(3) We have considered almost completely about all the states of our software, and refined a requirement document.
(4) In the requirement specification, we have considered exception handling for potential errors.
(5) The requriement contains basic calculations.
(6) We have a good time scale chart to manage our processes.
(7) Team working is the most efficient when each member is doing what he or she does best simultaneously.
(8) We understand there are preferences among user groups, and we know what they like.
(9) We have discussed about some operable stories (environment-protection, etc.).

Drawbacks and reflections:

(1) We are not professional game developers, so we must learn from scratch.
(2) The work is assigned unevenly to each person.
(3) We need to setup main milestones for the game code writing process.
(4) Many members have family issues or work, so we don't have much energy to refine our project.
(5) We still need to clarify our (main) target user groups – users in the whole world? Users in Asia? We may refine and change our decisions as the users' preferences change, as we are doing Agile.

Expectations:

(1) Pick out a good story to deliver.
(2) Work out all the main functionalities for a TD game.
(3) Select a reliable platform to test & release.
(4) Do user testings in Feb or Mar.
(5) Prepare for any changes in any initial design.
(6) Coding:
    - Construct basic structure based on the specifications.
    - Do calculations for the weapons.
    - Set weapon upgradings.
    - Construct menus and settings.
    - Design different levels.
    - ...