**TFTBot Project:**

**Analysis.**

The Problem.

Team-Fight Tactics, often abbreviated as T F T, is an auto-chess, strategy game, created by Riot Games and running on the engine same engine that powers the popular multiplayer online battle arena League of Legend. TFT is an incredibly complex, strategic game that requires a vast amount of game knowledge and experience. Getting skilled at TFT is a large time commitment and overall a slow process. One large hinderance is the lack a training or test mode resulting in the fact that the only way to improve at TFT is to play TFT. There is not a “test mode” for instance that allows you to create a team composition and battle it against a different one and so if you wanted to get experience with a certain team composition or item, you would have to hope you get the opportunity to play the team composition/ item, which is no guarantee, meaning you could waste a large amount of time unnecessarily trying to learn about a rare interaction. If Riot Games were to implement a custom or a test game mode, lots of time could be saved, leading to shortened learning time and quicker improvement at the video game. It would reduce the time commitment that new players need to improve at the game, allowing them to enjoy the game more than if they were left in the dark, as well as allowing experienced users to test out ideas without spending lots of time trying to get the opportunity to do so.

The Stakeholders and Solution.

Suket Arya is an avid TFT player. Within TFT, what Suket particularly enjoys is brainstorming and creating new ideas for team compositions and strategies to test out against other, more popular compositions. However, with his enrolment into university, he no longer has the time to spend multiple games attempting to get a favourable position to be able to test out a team composition, and then the numerous games after that to fine tune the strategy into something completely viable.

My solution is to create a program that can simulate a board/ battle between two team compositions accurately. Following on from this, through a UI, I aim to give the user the ability to place down units and teams, giving the opportunity to users to test out ideas and learn certain matchups, without having to try to recreate the situation in an actual game, which could be a very timely investment. Furthermore, it would give more effective, instant feedback and allow slight tweaking of a board so users could see what they could have done better, rerunning a similar battle multiple times to show them what they should have changed to win that battle in the future, allowing for faster learning and improvement for new players.

This tool aims to be useful for all TFT players, both those looking to scratch the itch of team building and strategic thinking, as well as those looking to improve quicker than their counterparts. Moreover, this tool will be useful at all levels of play, even for pro players who will enjoy the extra freedom it provides.

Existing Solutions:

There are no existing programs that allow for the simulation of TFT battles, however, as this aims to be an educational tool, I’ll compare this solution to what other educational tools out there for TFT.

***Guides and Videos from the Community:***

There are numerous educational guides available from the community that offer guidance, information and strategies within TFT. These are useful as a starting point for learning to play TFT, each having some advantages and disadvantages:

Positives:

* Can be a great starting point and can help beginners avoid simple pitfalls they often fall into.
* Can help pass on guidance from more experienced players to newer ones.

Negatives:

* Can be outdated or low quality
  + Guides must be constantly updated or else they risk giving outdated, invalid or entirely useless advice about the game.
  + The user also must be able to discern the quality of the guide, or there have to be good mechanisms in place for reviewing guides, otherwise new players can be given bad advice without knowing better, severely hindering their ability to improve.
* May not cover an issue or skill that a player needs to learn.
* Only useful for new players or semi advanced players

Examples:

* Mobalytics:
  + Graphical user interface, text, application, email

    Description automatically generated
  + Mobalytics offers a starting tutorial/ guide for beginners, helpful so for your first few games you aren’t completely thrown in the deep end with no help.
* Mobafire Community-Made Guide:
  + Text

    Description automatically generated
  + More in-depth community made guide available on mobafire, it offers some good tips, but now is very outdated, almost two years old. However, it is still one of the top options that comes up if you search for a TFT guide, so newer players could fall victim thinking it was still solid, relevant advice.

***Programs offering certain builds/ meta-advice:***

There are programs available which can run alongside TFT and give in game advice and overlays providing recommended build and team strategies directly in game.

Positives:

* Provides advice directly in the game
* Helps player stay up to date with the latest and most effective strategies known to the community at the time.

Negatives:

* Not adaptive
  + Provides no adaptive or specific advice to certain situations, simple informs you what strategies are popular at the time, no matter how viable that strategy is for you in that specific scenario.
* Only useful for new players or players completely out of touch with the meta (most effective tactics available).

Examples:

* Mobalytics App:
  + 
  + Mobalytics has an app you can install that offers in game advice as shown above. Handy as it allows you to select a team composition and gives you advice on where to place units, what items to go and what characters to get.
  + However as you progress further into the game, it gets less and less useful as if you never got the opportunity to go the team composition you selected, it offers nothing of value, furthermore, some players may try to force a composition they shouldn’t have due to their recommendation, being a hinderance rather than a help.

***Direct advice and coaching from better players:***

Newer players can always get better by following advice and direct coaching from more experienced players and is probably the most efficient way to get better at the moment.

Positives:

* Provides useful, accurate advice.
* Specific to your situation, can tell you the optimum move in any scenario.
* Can provide helpful tips and knowledge.
* Helpful at every skill level

Negatives:

* Not accessible to everyone.
* Requires a large time commitment from the better player.
* Can buy paid coaching which can be very expensive.

Examples:

* A screenshot of a computer

  Description automatically generated with medium confidence
* Live coaching is an effective way to rank up from the very top of top players, and is effective at higher ranks when other methods offer less and less benefit as you go up in rank.
* The downside is that it is very expensive, having to spend lots of money for even a relatively short period of time coaching.

Need for Computational Methods.

The problem at hand is uniquely suited to being solved by a computational methodology. With my project, we can use decomposition, breaking a large problem down into many smaller parts, to split up the project in numerous ways. Units, status effects and items within the game can be represented with classes and a board class could run iteratively until the battle is over. Moreover, my program is suited for abstraction. The original TFT always utilises abstraction, however, in order to complete my program, I will need to recreate/ simulate certain parts of TFT, but by utilising abstraction, I can ignore certain aspects of the game that are irrelevant to my needs, such as the item shop and in-depth graphics they offer. I can also utilise abstraction to create a simplified graph/ breakdown of my project: My project initially can be split into two sections, the initial recreation of the game and then the creation of an AI to run on said game. We can go further and further with decomposition, breaking down the recreation to:

* Recreating the Units
  + Recreating traits
  + Recreating each unique ability
* Recreating the Combat
  + Recreating the board
  + Recreating movement
* Recreating the economy/ shop
* Recreating the roulette

and many other steps, all of which could be broken down further. The creation of the AI can also be decomposed further, into:

* Creating the AI in Python
* Training the AI
  + Teaching the AI Unit placement
  + Teaching the AI effective economy management
  + Teaching the AI to consider and counter opponents.

My project is also suited to other parts of computational methodology

The initial creation of the game is suited to computational methodology thanks to the abstraction that can be used. The original TFT game is an abstraction of real life, not simulating things they don’t deem relevant to the game, such as not simulating gravity or jumping, accurate collisions, light rays and using other simplifications, such as only allowing units to move from one grid tile to the next and only allowing one unit to possess a grid tile at once. My recreation will use further abstraction, only including the parts relevant to the simulation of the gameplay, thus discarding features such as the GUI and cosmetics, as an AI has no use for it.

My recreation/ TFT is suited for computational methodology in other ways as well, such as pattern recognition. For example, rather than creating a unique variable for each Unit/ Character in the game, we can create a basic Structure/ Class to represent all of them, and simply change the base stats/ attributes so that they fit for each unit. The same applies for the items and traits, which each share overlap within their own category.

The second section, the AI, will also make use of reusable components. Rather than programming the deeply sophisticated and advanced machine learning algorithms that many vastly more experienced programmers have created over the years, I will feature and adapt them for my project.

Features.

There are two main essential features I would call crucial to the success of the project. The ability to, at any time, play against an AI with a difficulty level of your choosing and the ability to have an AI to review and give advice on *a game you are currently or have just played.*

The ability to play against an AI of chosen difficulty level is essential because this is what my program can offer versus getting advice/ playing with a high-level player. If available at anytime, using this program will be much easier and more convenient then finding and trying to ask a good player for guidance. Moreover, by allowing them to play against an AI with customisable difficulty level they can always have a good challenge that allows them to improve, without getting blown out of the water by vastly better players.

Furthermore by granting users the opportunity to get instantaneous feedback, it allows people to see their mistakes and how they could improve. Immediate feedback is the most important part of fast improvement and critical to speedy improvement. By offering immediate feedback we can offer a service only comparable to having a high level player in the call with you as you play and blow other methods of improvement out of the water. Moreover, by being adaptable and able to give advice not only in game but in review of games past, our program would offer far more utility than other programs that currently exist which only offer guidance before a game about powerful overall strategies to follow and nothing in or during the game.

Limitations.

There are a few limitations of the program and my development of the program. For starters, I have limited time and resources to pour into the project, which is accentuated by a lack of long term experience in programming and Rust in particular. The desired and full implementation of planned features may not be possible due to time constraints, especially as a set amount of time will have to be dedicated to working my way out of bugs/ pitfalls that more experienced programmers have seen before and know how to avoid.

Additionally, time constraints are a uniquely threatening prospect to a project like this which involves training AI, which unavoidably can take a large amount of time with little that can be done to speed it up. Good and efficient development of the programs to be as fast as possible is necessary to allow the AI to iterate through generations of learning quickly. This could be alleviated by the use of powerful hardware, however, as stated before, a lack of resources also means that this is improbable, meaning that the final product of the recreation and AI have to be highly efficient.

Likewise, the success of the project is dependent on the AI being able to progress to a high enough level to challenge skilled players at the game, a potentially untenable aim.

A potential limitation to the project is getting information from live games. While this isn’t necessary for the initial development of the project, once finished, in order to give feedback from live games they will have to be able to glean information from those games, which may be possible through the Riot Games API, or, if not, will have to be gained through looking at the game as an image in the program and harvest information that way.

Hardware.

To run the finished project, users will just need to have a computer able to run TeamFight Tactics and the storage space to store my program, as running it should have very little effect on the hardware of the computer. The computer must be running the operating system Windows, as I do not plan to make this program cross platform due to time constraints. Due to the (aimed) low performance cost of the program, any users who are able to run TeamFight Tactics should be able to simultaneously run my program. The (minimum) hardware requirements for TeamFight Tactics are listed below:

* 2Ghz processor (supporting SSE2 instruction set)
* 2GB Ram
* Shader version 2.0 capable video card
* 8gb free disk space

Success Criteria.

There are a few success criteria the project has to fulfil for the project to be called a success. The project has to:

* Create a basic recreation of the TFT game
  + Fast and efficient
  + Can interface with an AI
* Create an AI that can play TFT at a basic level
  + That considers economy and opponents
  + Can effectively strategize and place units
* Enable the AI to offer feedback on previous games played

Justify success criteria, limitations, hardware requirements

**Design.**

Breaking down the Problem.

*Draw.io Diagram*

