**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 based on the engine that the popular multiplayer online battle arena League of Legends runs on. 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, mainly due to the issue that Riot Games provide very restricted access to the game. To elaborate, 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 Solution. **Comparable to having a high level player with you coaching you.**

My solution is to create a program that can simulate a board/ battle between two team compositions accurately, giving the opportunity to users to test out ideas and learn certain matchups, without having to waste time trying to recreate the situation in an actual game, which could also cost them their rank in the process. Furthermore, it would give more effective 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, something that is not possible in an actual TFT game.

The Stakeholders:

I am creating this program for Suket Arya, an avid TFT player, who wants to be able to test out certain teams, but with university doesn’t have the time available to commit to play multiple, full games out just to try a single battle. This tool will also be useful for other TFT players who want to improve their skills and learn more about the game in a more productive and in-depth fashion.

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:

* Can be outdated or low quality
  + Guides have to be constantly updated or else they risk giving outdated, invalid or entirely useless advice about the game.
* Can be low quality/ incorrect:

Many guides can suffer from being incorrect or of low quality, whether due to malicious intent or not. The writer of the guide has to be experienced and willing to spend vast quantities of time creating a quality and accurate guide. If they wilfully make a mistake or not, any errors that are present in their work can harm the thousands of players who may choose to read it.

* Requires the user to search/ discern high quality guides and advice:

The user will have to search for high quality guides, discarding low quality ones and be able to discern high quality ones from low quality ones.

*Programs offering certain builds/ meta advice:*

There are certain programs you can download that will give you raw information about the game and current “meta” (most effective tactic available, so the best strategy available at the time, which often times its constantly updating).

* No adaptation

These programs do offer assistance and help formulate a strategy before the game, however, once in the game, they offer little to no guidance and cannot help you adapt to the game situation that may make your plan completely obsolete and untenable. Furthermore, because they do not gauge any information from the current game-state, some advice they offer could be incredibly and hopelessly incorrect and even do more harm than good if followed.

* Similar to community guides

They also offer no information that isn’t gainable with a quick internet search. Thus, these programs also share many of the short-fallings of community made guides, where the program has to be certain it is giving good advice and can risk giving incorrect or outdated guidance occasionally.

Furthermore, what happens if the common consensus on what the most effective way to play the game is incorrect, teaching new players potentially the incorrect way to play the game will slow down their development and improvement. If trained for long enough, the AI could potentially revolutionise the way that people play TFT by discovering new more effective strategies.

Our method also solves the other issues raised earlier. The program/ AI can offer trustworthy and correct advice and guidance instantaneously, backed up by experienced gained from thousands or millions of games played. It can adapt to every situation, as it is almost guaranteed to have encountered it before, or if not, can run a new simulation to provide the best advice to reach the best outcome. It can give feedback on what you should have done differently, around the clock, as you don’t have to rely on people to be available.

It can be constantly up to date or even ahead of the game, as patch notes are released in advance.

My solution fixes all the main issues that other methods of learning the game suffers from, providing the optimal and most efficient way to learn the game.

Mobalytics

Graphical user interface

Description automatically generated

TFT Wrapped

Graphical user interface, application

Description automatically generated

Mobafire

A screenshot of a computer screen

Description automatically generated with medium confidence

Item Cheat Sheet Outdated

A screenshot of a video game

Description automatically generated

Abilities Outdated

Graphical user interface

Description automatically generated

Mobalytics Guide

Graphical user interface, text, application

Description automatically generated

Outdated Mobafire guide:

A screenshot of a video game

Description automatically generated

Text

Description automatically generated

Need for Computational Method. Splitting the Project Somewhere, copypasta digram

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. 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*

