15-112 Term Project Proposal

The problem the term project attempts to address is the lack of diversity in real-time strategy games. Too many of these games have a strategic component but these games quickly die off in the gaming community due to it’s player base becoming bored. To address this problem, this real-time strategy term project will include multiple ways to win, guaranteed different gameplay every time, all the while maintaining the classis RTS feel. The multiple ways to win increase the complexity of the gameplay experience as users must focus on one of two victory conditions. They can either focus on a scientific victory where they research technology and build to win, or they can focus on a military victory and create an army that will destroy their opponents. There will also be a single player aspect where users can play by themselves to accomplish randomly generated objectives around the map. This increases the complexity of the project itself because it requires multiple ways to play the same game and inherently more code. Additionally, the game should have seamless multiplayer using sockets. While sockets themselves are complex this term project will allow hosts and clients to play multiplayer without ever opening the command prompt. The host will type their IP and a server will be opened and they will automatically connect to it. Clients will connect to the IP and never have to worry about understanding sockets. The overall game will be programmed in pygame which is not a library that was covered in class. This will add sprite complexity and the inherent complexity that comes with learning a new library. Lastly, this project is made more complex by the unique gameplay every time. The map itself is a 100 by 100 tile board with 25 randomly generated forest each containing 100 trees and 10 mines throughout the map. The gameplay is made more diverse by having a greater diversity in resources but the code itself is made more difficult by the randomly generated map. To ensure the maximum number of trees, the game uses a recursive backtracking algorithm to place the first tree and subsequently develop a path of 100 trees that do not overlap with anything else on the board. This is inherently more complex than other backtracking problems because of the required randomness in moves. Unlike floodfill for example, this algorithm cannot check top first every time because the likelihood that it is open is very high. This will result in a bunch of rows of trees. The randomness with the backtracking will create organic looking forests that will never be the same but always be the same size. Lastly, this project is complex because of the vastness of the concept. Lots of buildings and units have to be created with unique roles in the game and each has to be animated with its own functions.