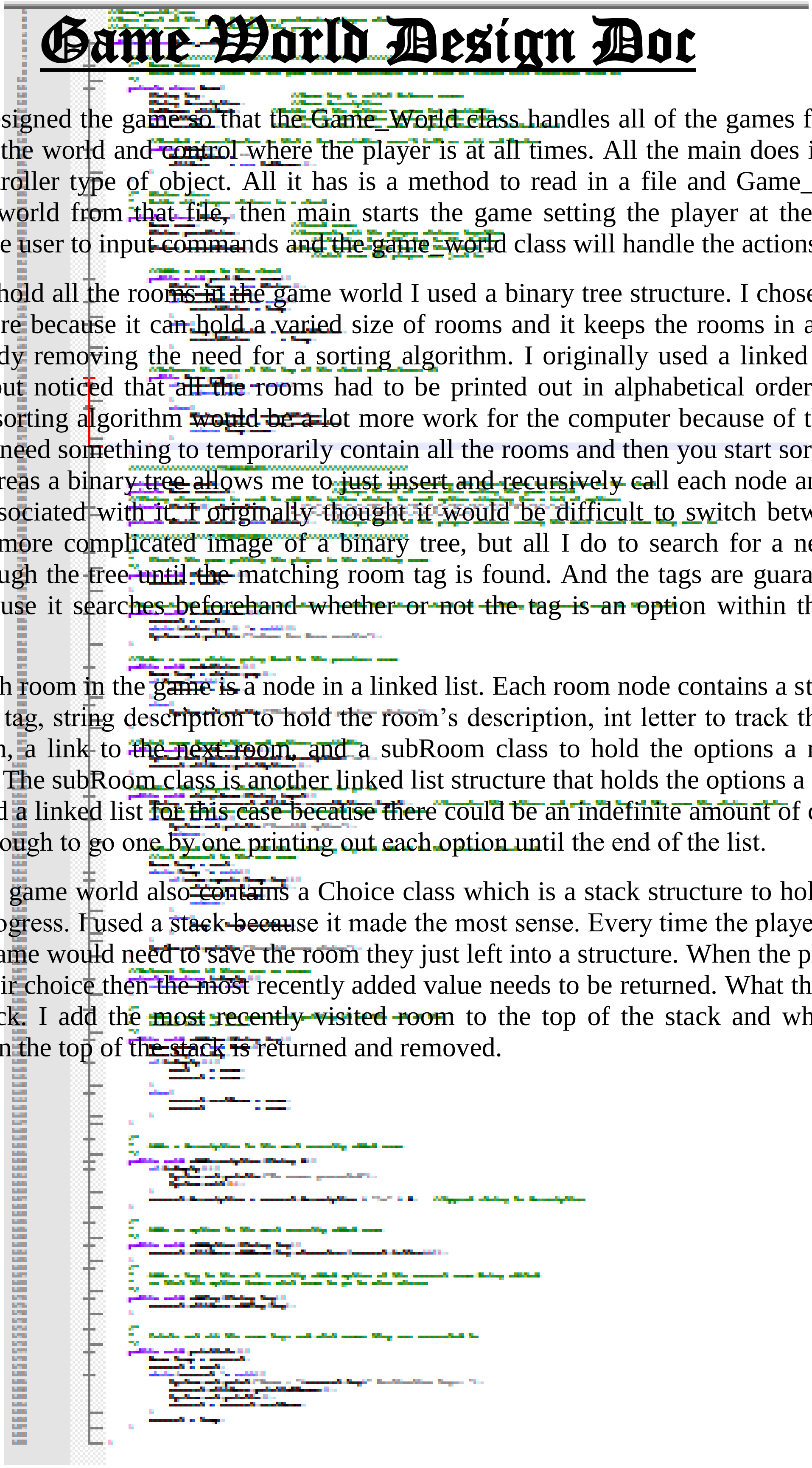
Game World Design Doc



I designed the game so that the Game_World class handles all of the games functions. It will create the world and control where the player is at all times. All the main does is act like a menu, controller type of object. All it has is a method to read in a file and Game_World will create the world from that file, then main starts the game setting the player at the first room allowing the user to input commands and the game_world class will handle the actions.

To hold all the rooms in the game world I used a binary tree structure. I chose the binary tree structure because it can hold a varied size of rooms and it keeps the rooms in alphabetical order already removing the need for a sorting algorithm. I originally used a linked list for the structure, but noticed that all the rooms had to be printed out in alphabetical order. I felt that creating a sorting algorithm would be a lot more work for the computer because of the fact that you would need something to temporarily contain all the rooms and then you start sorting it from there. Whereas a binary tree allows me to just insert and recursively call each node and print out the info associated with it. I originally thought it would be difficult to switch between rooms creating a more complicated image of a binary tree, but all I do to search for a new room is iterate through the tree until the matching room tag is found. And the tags are guaranteed to be found because it searches beforehand whether or not the tag is an option within the subroom options.

Each room in the game is a node in a linked list. Each room node contains a string tag for the room's tag, string description to hold the room's description, int letter to track the letter for each option, a link to the next room, and a subRoom class to hold the options a room could connect to. The subRoom class is another linked list structure that holds the options a room could have. I used a linked list for this case because there could be an indefinite amount of choices and it's easy enough to go one by one printing out each option until the end of the list.

The game world also contains a Choice class which is a stack structure to hold all of the player's progress. I used a stack because it made the most sense. Every time the player chooses a room the game would need to save the room they just left into a structure. When the player wants to undo their choice then the most recently added value needs to be returned. What that basically is, is a stack. I add the most recently visited room to the top of the stack and when undo is pressed then the top of the stack is returned and removed.