Text Adventure Report

When starting the game the first thing I looked at was other text adventure games and common commands that are used in these other text adventure games. From looking at all the Zork commands (Anon, Unknown) and reading a short list of common text adventure game commands , I was able to put together all the commands that I wanted to have in my game. From looking at these lists I noticed that a lot of the common commands such as movement and inventory all had multiple different inputs which did the same thing. An example of this is to move north you could either type north or n.

After figuring out what commands I was going to have in my game I had to figure out how my input parsing was going to work. I originally started with using the string. Contains() method and checked to see if the users input contained part of a command. This did work however I noticed that using this system could cause problems further in development when more items and commands were added. This was because someone may type a word that contains “use”. This could then trigger the use command instead of the actual command they wanted. Another problem that I realised I was going to have with this system was when I added the craft command. This was because I could check to see if “craft” was inputted but then I was left with a string containing the two items the user wants to craft with. After reading an article posted by Mircrosoft (Mircosoft, 2015) I changed the input parsing from the previous method to splitting each individual word into a string array. This allowed me to be able to check each word entered separately fixing my previous problem. Even though this system for input parsing was better than the previous there was still a major error that could happen when the user tried to input a command. This happened if the user failed to enter the whole of a command e.g. “craft stick” without adding the final item in their input. This caused the game to get a “System.IndexOutOfRangeException” as it would look for input[2] in the array which wasn’t part of the array. This meant that I had to check the array size before checking it the items entered were valid.

After I finished creating the craft and use commands I wanted to have new exits be added after certain items were used. An example of this is my game is when someone uses a rock to break a window. This was a problem because all of the location variables were in the game function. I originally kept all of this information about the locations in one function to keep the code contained and not having too many public functions that were unnecessary. Even though I needed to change any variable that I wanted to access outside of the game function to be public I still felt that I managed to keep the code generally neat and well contained.

Once I had figured out how I will manage input parsing I moved on to how I will store the player’s inventory and all the information about the items in the game. I started by creating variables for item names and descriptions which the player would see when they looked/examined an item. When the player added an item to their inventory or used an item functions would be called in the item class that add the item to the inventory list or remove the item from the inventory if the player uses it. One problem that I had when working on this was when the use tried to add an item that wasn’t in the scene as the game would try to add an item that never existed causing an error. I fixed this by checking the users input against all item names in the current location before trying to add it to the players inventory.

When designing the attack command for my game there was several different ways that I could have gone about handling the enemies and how things like damage and health are stored. I started by looking at other text adventure games to see how they handled combat. One game that stood out as having a very good combat system was dwarf fortress. Dwarf fortresses combat system works by having several different properties that can affect the player when fighting. These are the player’s weapon (which have several different statistics to it), opponents armour (which also has several different statistics) and several other small factors like the size of the monster you are fighting (Anon, 2015). After looking at this combat system I quickly realised that this would be too complex for my game so began looking at different ways to implement the combat for my game. One combat system that I saw suggested was simpler and choices were limited compared to dwarf fortress. This system was where the player will be notified of an enemy, player picks their action (attack/run), damage is dealt to player/enemy and then player has choice of actions again until either of the two die (Active Uniquie, 2013). I decided to use the bases of this combat system for my attack command as I felt the simplicity better matched the rest of my game. In game the system works so the player is notified that the monster is near them then the game waits for their input (They can either attack or move). If they attack and have a weapon then they will damage it allowing them to continue however if they attack without a weapon or try to run the monster will kill them.