**Development Report**

Initial Research

When planning, and researching my algorithm I ended up playing a few text adventure games before attempting the assessment. I played these games mostly for examples of different input parsers and commands. The games I played gave players the freedom to type commands, and could handle complex commands from intelligent parsers.

Games used for research

* Zork
* Jigsaw
* Galatea

When playing the games above all command parsing was very similar, the main format used was a verb then a noun for example “Take Rock” or “Attack Enemy”. I noticed from looking at these games that commands for movement and inventory had more than one input which did the same thing, for example to view you inventory you could type “i” or “inventory”. I wanted to add this to my algorithm as it didn’t force player to write long commands constantly, instead of players typing “Attack” every time they wanted to attack an enemy they could just type “a” or something similar.

Input Parsing

When deciding on my input parsing I first tried using the “Contains” method which would check to see if the users input contained part of a command (Dotnetperls.com), (msdn.microsoft.com).

 else if (command.Contains ("take"))

            {

                Foreach (Item item in currentLocation.getInventory ())

                {

                    If (command. Contains (item. get Name ().To String ().To Lower ()))

                    {

                        Console.WriteLine ("\n>You take the " + item.getName () + " and place it in your inventory ");

At first this method worked and there were no problems, however when I started adding more items and commands to the game I had a big problem. If I typed a word that contained “use” or “take” It would trigger those commands and not the command I wanted. To fix this problem I decided to change the input parsing into a string array, this allowed me to check each word separately. This method was a lot simpler than the previous methods, however you were not able to input complex commands or simultaneous commands.

if (input. Length > 0)

{

if (input [0] == "help" || input [0] == "h")

{

Console.Clear ();

Console.WriteLine (@"HELP!

When I finished my input parsing I then changed my focus to player inventory. To implement this I created variables for the item names and descriptions and created a list for the inventory.

//Initialises players inventory list and items

Inventory = new List<Item> ();

PlayerInv = new Item ();

I then built the level and placed the items in specific locations, the player has the ability to use the take command but can only take the items in the current location. Once a player uses the take command it remove that item from the location and add it the player’s inventory. This command ties into the use command, once a player uses an item in their inventory it removes that item. However items could only be used in specific location, for example using the crowbar on the on the lock shack. If player try and use items at a location where it doesn’t work a message will say “You can't do that!”

Location level3 = new Location("small shed", "This is a cluttered shed, could be something here.");

Item key = new Item();

key.name = "crowbar";

key. Description = "A crowbar. It could break a lock.";

level3.addItem(crowbar);

The next idea I tried for my game, was the ability to add exits once a certain item was used in game. For example, in my game you need to brake a lock with a crowbar, if you don’t use the crowbar on the lock that exit won’t be available. What it used were else if statement saying

else if (item1.name == "crowbar" && currentLocation.ToString() == "Small shack")

{

l7.addExit(new Exit(Exit.Directions.East, l8));

playerInv.RemoveItems (item1);

If the player uses the item named “crowbar” and their current location is the small shack it adds the new exit to the game and removes the crowbar form the player’s inventory. Obviously the player will need to use the item to progress in the game, so if they didn’t pick it up they will need to go back.

The combat system was my next step, I found several examples on how to give enemies a health value and give weapons a damage value, however I thought this would be too complex for my game so I decide on a very simple approach. I basically added 2 location where and enemy would appear for the player to attack. Players would only be able to attack if they have the weapon in this case a spear in their inventory, if they don’t have this they will die.

If (currentLocation.ToString () == "Wolf den")

{

if (item.name == "spear")

{

level6.addExit (new Exit (Exit.Directions.North, level7));

Console.Clear ();

ShowLocation ();

Console.ForegroundColor = ConsoleColor.Red;

Console.WriteLine ("You attack the wolf with the spear. It's wounded but manages to escape");

level6.setDescription ("You walk deeper into the forest. There is a blood trail left from the wolf.");

Return;

In conclusion the final iteration of the text adventure game allows the player to move, use items, store and check inventory, craft items, examine items, attack enemies and check help menu. I was also able to add a variety of input for the same function. If I was able to continue development I would try and make the game using the Contains method, as I believe I could have add more complex commands into the game and attempt simultaneous commands. I would also want to add a health value to enemies and a damage value to weapons. I would also try and find a way to create the game with less hard coded else if statements.

References

Anonymous (N.D.) *2D Array* [Online] Viewed On 30/12/16. Available At: <https://www.dotnetperls.com/2d>

Anonymous (2015) *Multidimensional Arrays* [Online] Viewed On 30/12/16. Available At: <https://msdn.microsoft.com/en-us/library/2yd9wwz4.aspx>

3D Buzz (N.D) *Hyperion Class Layout*[Online] Accessed On: 30/12/16. Available At: <https://www.3dbuzz.com/training/view/xna-volume-1/hyperion-text-adventure-game/hyperion-class-layout>

De Smet, B (2013) *c# 5.0 Unleashed.* SAMS. Indiana, USA.