ZORK of the Rings - Assignment

ICP-2018 - Natural Language Processing

Michael Smith
<eeu213@bangor.ac.uk>

May 13, 2015

Abstract

This report will go over the preparation, implementation and evaluation of my ZORK style game created using Python 2.7.9. We were tasked into creating a conversational agent to allow a human to play an Adventure (ZORK) style game, were we had to use natural language processing technologies. For more information about ZORK check out `http://inventory.webs.com/howtoplay.html`.

I decided to base the theme on my game around Lord of the Rings as there was a lot of content I was able to use and produce from the story itself my game's emphasis is on navigation around the mythical land of middle-earth with winning and losing conditions to give it a real game feel rather than just an adventure sandbox.

The introduction will go over my design and introduce what technologies I used, as well as go over specifically what I was asked to create according to the brief.

In addition this document will contain other design documentation such as the map which I created and used to navigate and connect each node up to each other, the map gives a really good insight in what the game from a CLI perspective is supposed to look like in a physical perspective.

This documentation will also include evidence of the workings of the program which mainly consists of a lot of unit testing on the separate nodes and events in those nodes to prove that the code does work, aswell as evaluation the output of the code and what I could of done better or what was not done well as well as comparing the technologies I used in comparison to what could of been used.

The final section will just include any references used without this document was well as the source-code which works on Python 2.7.9 using the Python4Netbeans Plugin on Netbeans 8.0.3.

Contents

1 Introduction

This section introduces you to the assignment, what was required and what technologies and tools I have used to meet those requirements.

1.1 Brief

The project brief was to "Develop a conversation agent to allow a human to play an Adventure (ZORK) style game." A conversational agent is a dialog system which is created with the purpose to understand and reply to human interactions via text the most common example of this being a chat-bot, the agent has to work out what you are saying and act accordingly by dissecting your information into chunks such as verbs and nouns to process what action you are doing (the verb) and what you are doing it to (the noun). A ZORK style game is a text based game which is dictated by the what the user types into the command window, essentially this type of a game is a heavily modified chat bot with a theme and a server.

1.2 Technologies

For this assignment we were limited to certain technologies to complete it; we couldn't just go away and try and implement it in Java. We were limited to use only

- Python - an open source programming language which more importantly has interaction with the following technologies.
- NLTK - natural language toolkit, is a library of tools and programs which allows you to process inputted text into it's meaning (natural language processing).
- NLP Pipeline is the pipeline the NLTK as a whole uses to process the text/language.
    1. raw text
    2. sentence segmentation
    3. sentence tokenization
    4. part of speech tagging (POS)
    5. named entity tagging
    6. relation tagging

2 Design

This section will go over the initial design process of my game following being given the brief, most importantly it will display the map which is the most visual aspect of the entire assignment as the game is in-fact text-based so having a visual concept to work around from the start was extremely helpful and almost pretty much required.

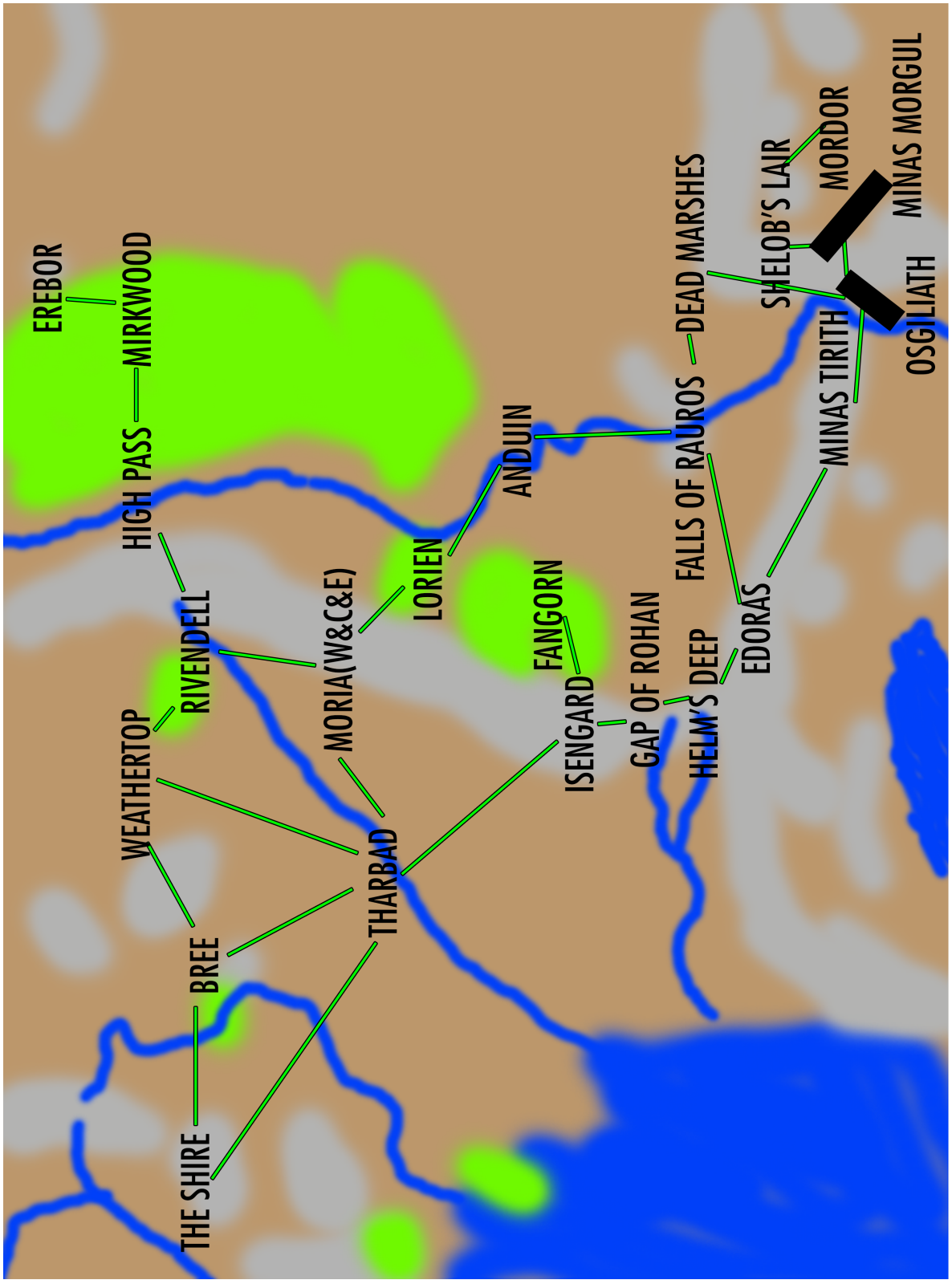
2.1 Concept

I was creating a game which would be based around adventure and navigation, in which I would achieve by having the user type for example "north" into the command window and the user will move to the next node to the north of it.

The next step was to think of a theme, I originally tried to create my own theme and create my own places, but instead I thought of trying to re-create existing themes into my style of game, I started my re-creating the world of Westeros but found the geography of the world to be too narrow (i.e most of the nodes would either be north or south of the user), so instead I looked around at various other known maps to find a suitable candidate. Where I found the map of middle earth which has perfect geometry for what I want to do as well as giving me my winning and losing conditions which is take the ring to Mordor, or die to a Balrog or giant spider!

2.2 Map

Now it would of been a mammoth task to re-create the entire world of middle-earth, so using this map as a base I took relevant nodes from this map and created my own map this time complete with North, South, East and West connectors for each node. Color coded grey for mountains, green for forest, blue for water and the rest is land.


EREBOR
MIRKWOOD
HIGH PASS
RIVENDELL
WEATHERTOP
BREE
THE SHIRE
THARBAD
MORIA(W&C&E)
LORIEN
ANDUIN
FANGORN
ISENGARD
GAP OF ROHAN
HELM'S DEEP
FALLS OF RAUROS
DEAD MARSHES
EDORAS
SHELOB'S LAIR
MORDOR
MINAS TIRITH
MINAS MORGUL
OSGILIATH

3 Code Implementation

This section will cover how I implemented my design in the previous section into code, this section will go through every file and function used; going over input, output, what it does and how it works. You can find all of the game code used in the appendix.

3.1 Game Handler

The first file handles the status of the game, this is a short file but it separates the actual game logic to the actual games existence, this handler is the main method for the game, it will be the executable.

The user input is the user actually clicking run and the output the user will get is a welcome message as well as the output message to say what location you are in, which is currently defaulted as "The Shire". The handler allows a programmer to easily change the starting location, maybe even create a random picker to have a different one every time you start the game.

The handler uses the print command to print out the welcome message, as well as importing the game logic file "game", then passing control over to that file by referencing the file following by ".location".

3.2 Location Nodes

This section covers all procedures within the game logic file, as all of the procedures are coded almost identically with the difference in each node being location information, for example the nodes to the east of "The Shire" is not the same as the nodes to the east of "Minas Tirith".

For the user input I am using the regular expression module "re.search" which searches a string for sequences which match the expression, you can access the regular expression modules by using the "import re" command.

So for each node the user visits the user has to input a direction, either north, south, east or west or (n, s, e, w) a mixture of lower-case or capitals is accepted due to the ".lower()" command appended onto the end of the "raw_input" statement however due to my regular expression of "(^(direction))" it only accepts that exact word for example "nOrTh" will work but "northd" will not work.

Once the user has entered something a decision is made by combining multiple "re.search" statements within the node, one statement for each direction this will process what direction the user wanted to go, if the criteria did not match any of the statements an "invalid input" message will be displayed as the user entered something which does not match the criteria such as "nortd".

So once the user as inputted a location successfully, the output will be to move to a different node or function, which has the exact same structure as the node previously described above with the only difference being that the location parameters are different to reflect what this node is connected to. The output moves you to the destined node which is different depending on what your current node is and then prints out to the user to say that you are now in that node.

3.3 Special Condition Nodes

Although the majority of the nodes are repetitive location nodes with similar inputs and outputs, there are certain nodes which have more special outcomes, which include the victory condition, the death condition and the bonus condition.

As this is a theme on Lord of the Rings, the win condition is of course; find Mordor, when the user successfully does then instead of the function asking the user for a new location to go to the game will end with a message saying "You reach Mordor you win". This condition is different and special because it does not loop like the other functions, instead the game will end and the user will have to click on the game handler to start the game again. This function does not require any user input during it being ran like the other methods as it does not use the "re.search" statements, it merely just prints out a message and exits the program.

Also there is the lose condition which I implemented in four possible commands in two different locations; within Moria and Shelob's Lair there are only two options: entrance and exit, so typically in cave like environment if you go in any other direction you are going the wrong way, which in this case will get you killed by what ever mythical creature is under the mountain or in the cave. For example if the user goes into Moria which typically has E for Moria East Gate and W for Moria West Gate and the user inputs N or S then they will be killed by the Balrog as they have clearly got lost and ventured the wrong way and into territory they should of left alone. This condition follows similar to the win condition as it will print out a message informing the user of his demise before exiting the program.

The final condition is an easter egg type condition, which is more of a combination between the win & loss condition and a normal location node, when the game is launched the user does not know of this condition and the path to reach this condition is vague, however upon reaching the location node Erebor and a special message is displayed saying that the user has found Erebor and have earned bonus points the difference between this function and the other special functions is that it does not stop the game after this has been triggered, also it uses regular expressions and input like the location nodes, this is essentially a location node but with an extra message which makes it special.

4 NLP Techniques

For this assignment the main NLP Technique I used to complete it was regular expressions, more specifically "re.search" I felt that with the required input from the user being as minimal as a single word or in some cases a single character I only needed a simple regular expression to determine if the word was that which was entered.

I could of used different methods such as following the NLTK pipeline. Which would allow the user to say type "run north", "fly north" and the tagger will be able to differentiate between the verb and the noun which I could then use to create more user-feedback such as outputting the to the user "you can't use the eagles until the end". This also would of been a good idea if I was going to extend my program more from navigation if there was more to the game than direction and more about actions such as "pick up [item]" it would of been more beneficial and efficient to use tagging of the input rather than multiple expressions. All of my regular expressions are as follows:

```
re.search("^(n)|^(north)", command)
re.search("^(s)|^(south)", command)
re.search("^(e)|^(east)", command)
re.search("^(w)|^(west)", command)
```

These are quite self explanatory, there are only four options in my game so there are only four expressions, each expressing is linked to it's corresponding if statement which runs whichever condition is necessary depending on the position of that particular location node.

The syntax for these expression:

- "^(direction)" - means only everything inside the brackets, for example nort will not work but north will.
- "the pipe symbol" - means or, it could be either expression it does not have to be both north and n
- command - means the parameter it is checking, this the raw_input command which is where the user stores his direction input, this command is already converted so what everything is lower-case which allows the user to

5 Testing

This section will go over all of the test history done on the game during and after the implementation process, it will go over every procedures input and output the wanted output and the actual output.

5.1 Input Testing

This tests all of the inputs the user has to make to the game, to make sure there isn't a hole in the regular expressions which lead to a break in the program flow where there should not be or an unexpected output like going north instead of south, or one of the expressions not accepting modified text such as sOutH. You can find the results for the Input Testing in Tables 1 to 4.

5.2 Output Testing

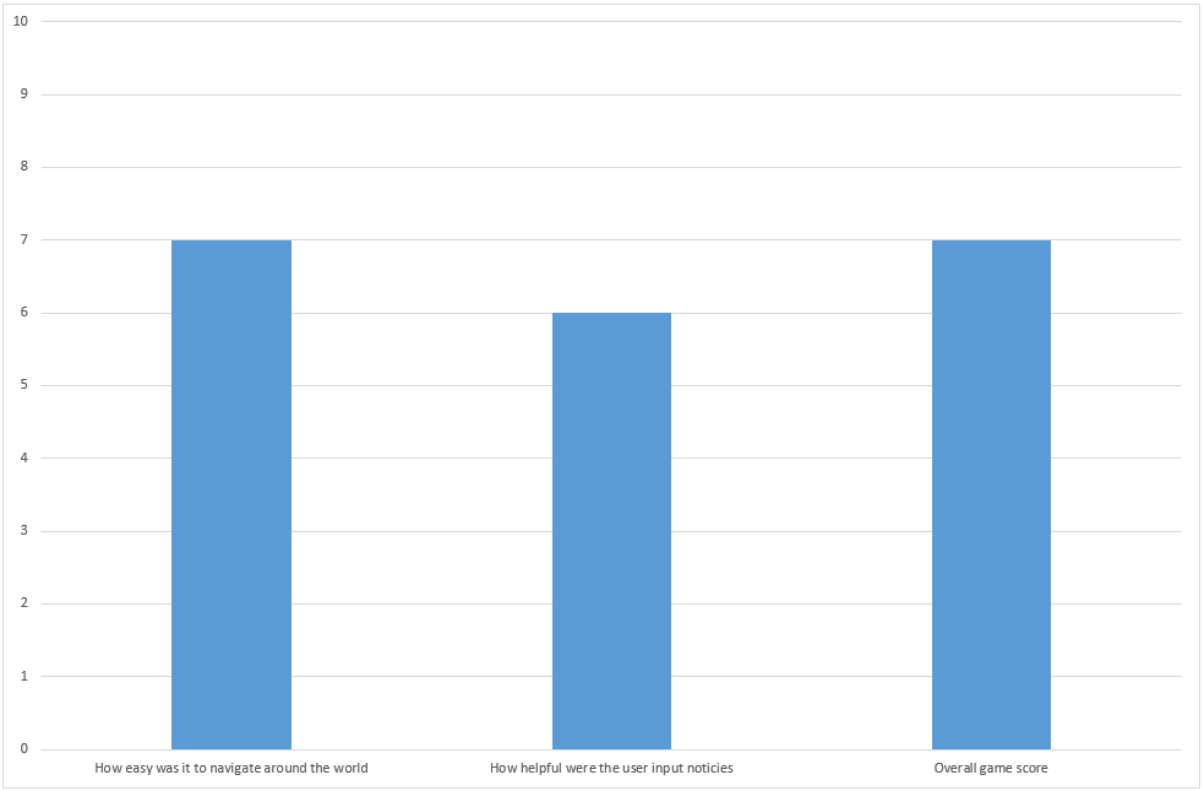
Here I ran tests on some of the outer nodes to test for any other possible errors within the code, this is to again ensure that there are not errors within the code, but this time instead of being the user input's error it is the programmers error, i.e miss-type in the function name of what node to go to next, the wrong node being linked up, no error message or return when there are no nodes linked up at all, since the combinations here are almost endless (nearly 100 possible combinations), I selected random nodes in the middle of the network and checked to see if they ended up the right place. As well as selecting four boundary nodes on each direction to ensure that there is an end node for each direction. In addition to this I tested the games winning losing and special conditions. You can find the results for the Output Testing in Table 5.

6 Evaluation

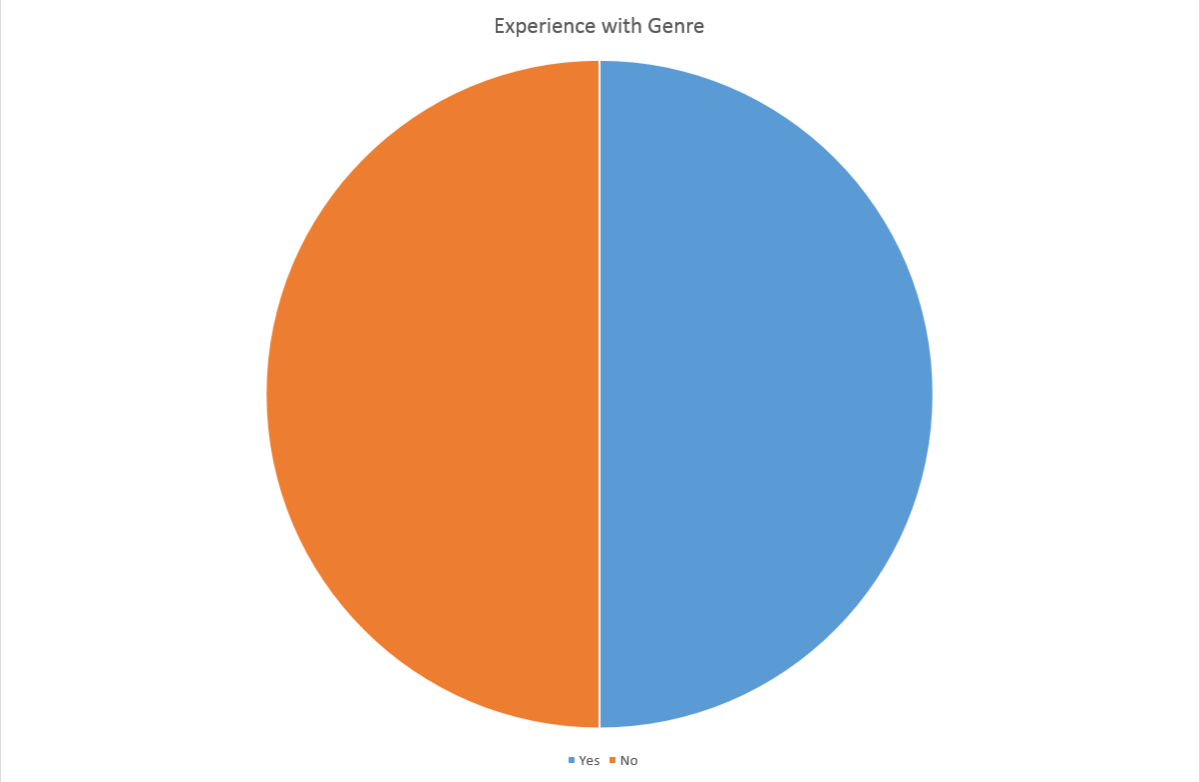
In this section I will go over my game and take a different perspective on it, and see what I done well, and what I did not so well on. As well as take feedback from others who played my game.

6.1 User feedback

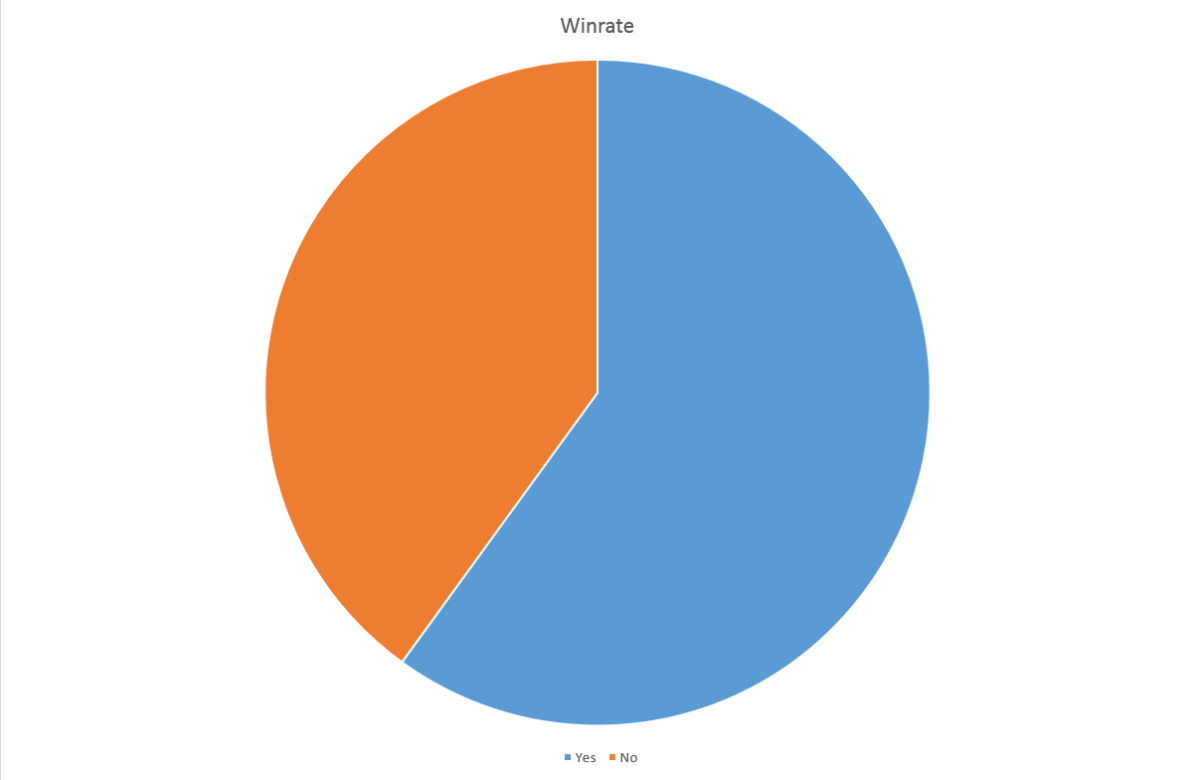
I had ten randomly selected players play my game and then they filled in a short survey, I collated the results and produced some tables.



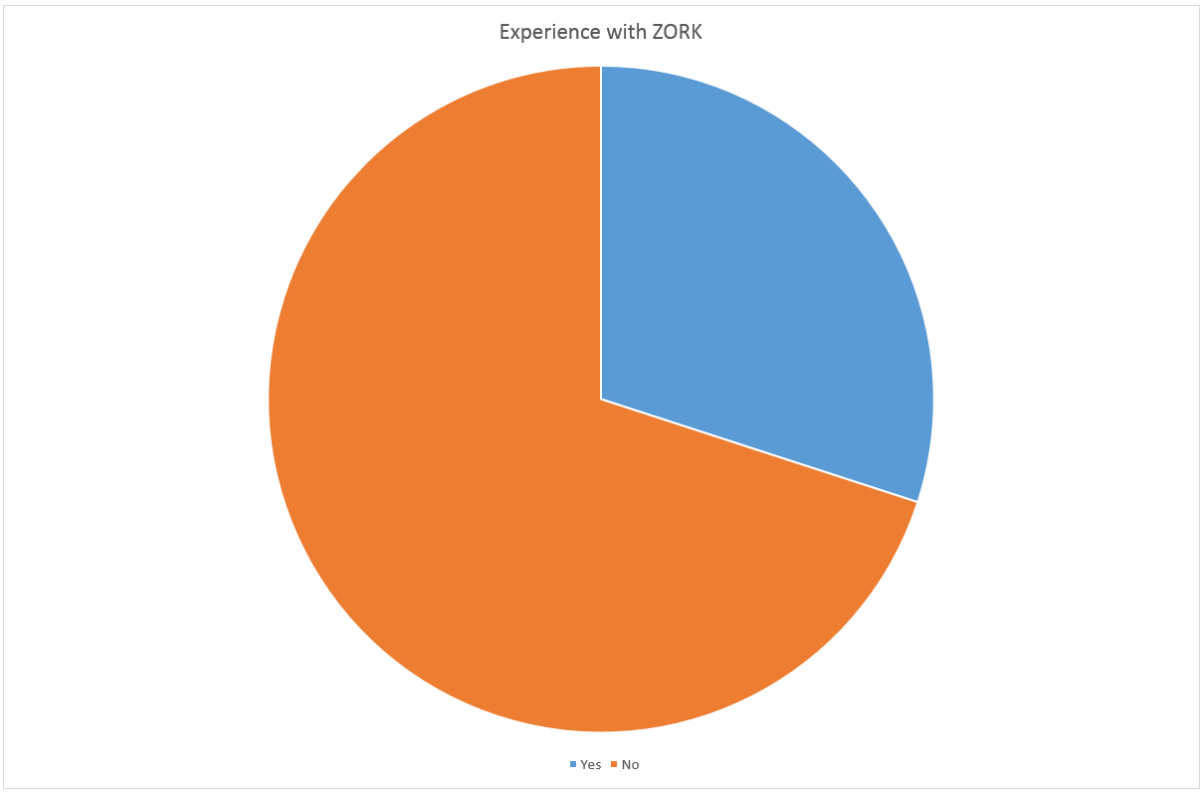
A graph to show users general experience when playing



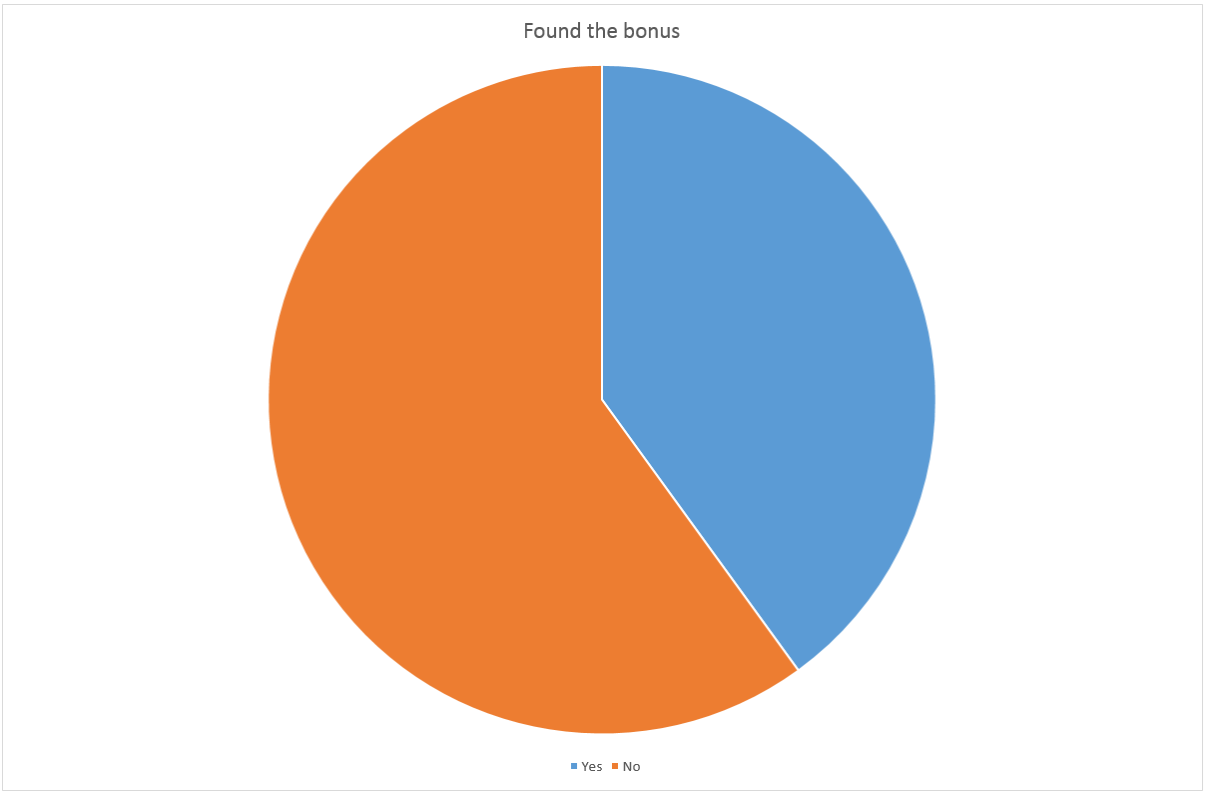
A graph to show all of the users experience with Lord of the Rings



A graph to show the win rate from all ten players



A graph to show users experience with ZORK



A graph to show how many users found the bonus

6.2 What went well

- I felt that the overall theme and flow of the game was rather successful, you can't instantly jump into the game and complete it by spamming buttons, you have to sit there, think and read to see where you are and where you have to go. Granted you could go online and find a map of middle-earth and follow it through since the game does flow the same way, but even to the most experience Lord of the Rings veterans you might have to sit there and think for a second to work out where you are.
- I think the input feedback was a good implementation, originally I did not have this, so when you got something wrong, like you tried to go north from the most northern point it did not say anything to say you did something wrong, now it will prompt the user to say "you can't go north from this position" instead of just blind guessing.
- The implementation of the lose and easter egg conditions gave the game an extra level which was not there originally, it was something else to look out for instead of their just being one way of ending the game there is three.

6.3 What could of been better

- I could of added more user options apart from navigation; instead of just moving around the map I could of added more interesting actions and events such as "to fight off the orcs press attack ten times", as well as adding other cool events which I could of linked to the genre.
- On top of this I could of improved the functionality of the navigation system, why just north, east, south and west why not north-east, south-west etc? My one reasoning for not doing this is simplicity in both expressions for the programmer and user-input, it will start getting easy to getting things mixed up. Although this feature would of made things more accurate geographically.
- As mentioned previously I could of explored more NLTK technologies which would of tied into my two points above nicely as more commands would of required an in-efficient amount of regular expression statements instead of just chunking phrases, as well was deciding where to go could of been done with a "go to place" rather than a go to direction in this way.

6.4 Conclusion

Overall I think I created a game with a solid ground and theme which meets the required project brief, there could of been design improvements aswell as quantity improvements in terms of node size, however taking everything I have created for this module and putting it to use logically for a game was a good task and I think the result has been an engaging and entertaining game for if not a few minutes of peoples time regardless of their experience with ZORK or Lord of the Rings.

7 Source Code

This section contains the Source Code for my Game, the source code is split up into two files the main method which controls the game state "lordoftherings.py" and the game logic which is called by the main method "game.py".

7.1 lordoftherings.py

```
__author__ = "Michael"
__date__ = "$09-May-2015 19:37:06$"
# this is the main method which controls the state of the game and calls...
# game.py to control the game logic
# game is based on the map and conditions of lord of the rings
# see: http://images1.fanpop.com/images/photos/2300000/Map-of-Middle-Earth-lord-of-the-r
import game # imports game.py to control the game logic
print ("Welcome to ZORK of the Rings, you are a hobbit with the" # welcome message
+ " one ring and you must reach Mordor before the Ring Wraiths catch you!")
game.theShire() # calls game.py and starts you in the shire
```

7.2 game.py

```
__author__ = "Michael Smith (eeu213)"
__date__ = "$09-May-2015 20:03:51$"

# game is based on the map and conditions of lord of the rings
# see: http://images1.fanpop.com/images/photos/2300000/Map-of-Middle-Earth-lord-of-the-r

import re # the regular expression module which provides regonition for patterns and str

def theShire(): # a node / state is a location in the physical world
# the current state means the current location where the player is at
    command = raw_input("You are in the Shire, which direction will you go? ").lower()
    # asks the user to input a direction (north east south or west
    # also accepts (n, e, s, w) in a mix of upper or lower case due to conversion
    if re.search("^(n)|^(north)", command): # this checks to see if n was entered...
    # or if north, both as whole words, if this is true then the statements are run
        print "You can't go North from The Shire!" # this location does not have...
        # any more locations connected to it from the north so it prints saying so
        theShire() # then it calls the same location again to ask for a new direction
    elif re.search("^(s)|^(south)", command): # this checks to see if s was entered...
    # or if south, both as whole words, if this is true then the statements are run
        tharbad() # thardbad is south of the shire so the state switches to tharbad
    elif re.search("^(e)|^(east)", command): # this checks to see if E was entered...
    # or if east, both as whole words, if this is true then the statements are run
        bree() # bree is east of the shire so the state switches to bree
    elif re.search("^(w)|^(west)", command): # this checks to see if W was entered...
    # or if east, both as whole words, if this is true then the statements are run
        print "You can't go West from The Shire!" # this location does not have...
        # any more locations connected to it from the north so it prints saying so
        theShire() # then it calls the same location again to ask for a new direction
    else:
        print "Please enter either north(n), south(s), east(e) or west(w)"
        # if an invalid input is put into the command window (i.e anything not north, e
        # south west or n, e, s ,w with or without capitals) then it will prompt...
        theShire() # so, then rerun the location asking for a valid input

def bree(): # this node and all of the following nodes all follow the same comments
    # and structure as the original node
```

```
    # see moria, shelobsLair, erebor and mordor nodes for special events
    command = raw_input("You are in Bree, which direction will you go? ").lower()
    if re.search("^(n)|^(north)", command):
        print "You can't go North from Bree!"
        bree()
    elif re.search("^(s)|^(south)", command):
        tharbad()
    elif re.search("^(e)|^(east)", command):
        weathertop()
    elif re.search("^(w)|^(west)", command):
        theShire()
    else:
        print "Please enter either north(n), south(s), east(e) or west(w)"
        bree()

def weathertop():
    command = raw_input("You are at Weathertop, which direction will you go? ").lower()
    if re.search("^(n)|^(north)", command):
        print "You can't go North from Weathertop!"
        weathertop()
    elif re.search("^(s)|^(south)", command):
        tharbad()
    elif re.search("^(e)|^(east)", command):
        rivendell()
    elif re.search("^(w)|^(west)", command):
        bree()
    else:
        print "Please enter either north(n), south(s), east(e) or west(w)"
        weathertop()

def rivendell():
    command = raw_input("You are in Rivendell, which direction will you go? ").lower()
    if re.search("^(n)|^(north)", command):
        print "You can't go North from Rivendell!"
        rivendell()
    elif re.search("^(s)|^(south)", command):
        moriaW()
    elif re.search("^(e)|^(east)", command):
        highPass()
    elif re.search("^(w)|^(west)", command):
        weathertop()
    else:
        print "Please enter either north(n), south(s), east(e) or west(w)"
        rivendell()

def moriaW():
    command = raw_input("You are at the West Gate of Moria, which direction will you go?
    if re.search("^(n)|^(north)", command):
        rivendell()
    elif re.search("^(s)|^(south)", command):
        isengard()
    elif re.search("^(e)|^(east)", command):
        moria()
    elif re.search("^(w)|^(west)", command):
        tharbad()
    else:
        print "Please enter either north(n), south(s), east(e) or west(w)"
        moriaW()
```

```
def moria():
    command = raw_input("You are in Moria, which direction will you go? ").lower()
    if re.search("^(n)|^(north)", command):
        print "You run into the Balrog and die. Game Over."
        # special event: if you input the incorrect direction to get out of moria
        # you find the mythical balrog and die and the game stops
    elif re.search("^(s)|^(south)", command):
        print "You run into the Balrog and die. Game Over."
        # special event: if you input the incorrect direction to get out of moria
        # you find the mythical balrog and die and the game stops
    elif re.search("^(e)|^(east)", command):
        moriaE()
    elif re.search("^(w)|^(west)", command):
        moriaW()
    else:
        print "Please enter either north(n), south(s), east(e) or west(w)"
        moria()

def moriaE():
    command = raw_input("You are at the East Gate of Moria, which direction will you go?
    if re.search("^(n)|^(north)", command):
        highPass()
    elif re.search("^(s)|^(south)", command):
        fangorn()
    elif re.search("^(e)|^(east)", command):
        lorien()
    elif re.search("^(w)|^(west)", command):
        moria()
    else:
        print "Please enter either north(n), south(s), east(e) or west(w)"
        moriaE()

def lorien():
    command = raw_input("You are in the forests of Lorien, which direction will you go?
    if re.search("^(n)|^(north)", command):
        highPass()
    elif re.search("^(s)|^(south)", command):
        fangorn()
    elif re.search("^(e)|^(east)", command):
        anduin()
    elif re.search("^(w)|^(west)", command):
        moriaE()
    else:
        print "Please enter either north(n), south(s), east(e) or west(w)"
        lorien()

def anduin():
    command = raw_input("You are following Anduin (The Great River), which direction wi
    if re.search("^(n)|^(north)", command):
        highPass()
    elif re.search("^(s)|^(south)", command):
        fallsOfRauros()
    elif re.search("^(e)|^(east)", command):
        print "You can't go East in Anduin"
        anduin()
    elif re.search("^(w)|^(west)", command):
        lorien()
```

```
    else:
        print "Please enter either north(n), south(s), east(e) or west(w)"
        anduin()

def fallsOfRauros():
    command = raw_input("You are at the Falls of Rauros, which direction will you go? ")
    if re.search("^(n)|^(north)", command):
        anduin()
    elif re.search("^(s)|^(south)", command):
        minasTirith()
    elif re.search("^(e)|^(east)", command):
        deadMarshes()
    elif re.search("^(w)|^(west)", command):
        edoras()
    else:
        print "Please enter either north(n), south(s), east(e) or west(w)"
        fallsOfRauros()

def deadMarshes():
    command = raw_input("You are in the Dead Marshes, which direction will you go? ").lo
    if re.search("^(n)|^(north)", command):
        print "You can't go north in the Dead Marshes"
        deadMarshes()
    elif re.search("^(s)|^(south)", command):
        osgiliath()
    elif re.search("^(e)|^(east)", command):
        print "You can't go East in the Dead Marshes"
        deadMarshes()
    elif re.search("^(w)|^(west)", command):
        fallOfRauros()
    else:
        print "Please enter either north(n), south(s), east(e) or west(w)"
        deadMarshes()

def osgiliath():
    command = raw_input("You are at the ruins of Osgiliath, which direction will you go?
    if re.search("^(n)|^(north)", command):
        deadMarshes()
    elif re.search("^(s)|^(south)", command):
        print "You can't go South in Osgiliath"
        osgiliath()
    elif re.search("^(e)|^(east)", command):
        minasMorgul()
    elif re.search("^(w)|^(west)", command):
        osgiliath()
    else:
        print "Please enter either north(n), south(s), east(e) or west(w)"
        osgiliath()

def minasMorgul():
    command = raw_input("You are at Minas Morgul, which direction will you go? ").lower(
    if re.search("^(n)|^(north)", command):
        deadmarshes()
    elif re.search("^(s)|^(south)", command):
        print "You can't go South from Minas Morgul"
        minasMorgul()
    elif re.search("^(e)|^(east)", command):
        shelobsLair()
```

```
    elif re.search("^(w)|^(west)", command):
        osgiliath()
    else:
        print "Please enter either north(n), south(s), east(e) or west(w)"
        minasMorgul()

def shelobsLair():
    command = raw_input("You are in Shelob's Lair (the lair of the spider), which direct
    if re.search("^(n)|^(north)", command):
        print "You find Shelob and die. Game Over."
        # special event: if you input the incorrect direction to get out of shelobs lair
        # you find the great spider shelob and die and the game stops
    elif re.search("^(s)|^(south)", command):
        print "You find Shelob and die. Game Over."
        # special event: if you input the incorrect direction to get out of shelobs lair
        # you find the great spider shelob and die and the game stops
    elif re.search("^(e)|^(east)", command):
        mordor()
    elif re.search("^(w)|^(west)", command):
        osgiliath()
    else:
        print "Please enter either north(n), south(s), east(e) or west(w)"
        shelobsLair()

def mordor():
    print "You reach Mordor you win"
    # special event: when you reach mordor, you complete the win condition so the game e

def tharbad():
    command = raw_input("You are in Tharbad, which direction will you go? ").lower()
    if re.search("^(n)|^(north)", command):
        bree()
    elif re.search("^(s)|^(south)", command):
        isengard()
    elif re.search("^(e)|^(east)", command):
        morieW()
    elif re.search("^(w)|^(west)", command):
        print "You can't go West of Tharbad"
        tharbad()
    else:
        print "Please enter either north(n), south(s), east(e) or west(w)"
        thardbad()

def isengard():
    command = raw_input("You are in Isengard, which direction will you go? ").lower()
    if re.search("^(n)|^(north)", command):
        moriaW()
    elif re.search("^(s)|^(south)", command):
        gapOfRohan()
    elif re.search("^(e)|^(east)", command):
        fangorn()
    elif re.search("^(w)|^(west)", command):
        print "You can't go West of Isengard"
        isengard()
    else:
        print "Please enter either north(n), south(s), east(e) or west(w)"
        isengard()
```

```
def gapOfRohan():
    command = raw_input("You are in the Gap of Rohan, which direction will you go? ").lo
    if re.search("^(n)|^(north)", command):
        isengard()
    elif re.search("^(s)|^(south)", command):
        helmsdeep()
    elif re.search("^(e)|^(east)", command):
        fallsOfRauros()
    elif re.search("^(w)|^(west)", command):
        print "You can't go West from the Gap of Rohan"
        gapOfRohan()
    else:
        print "Please enter either north(n), south(s), east(e) or west(w)"
        gapOfRohan()

def helmsdeep():
    command = raw_input("You are at Helms Deep, which direction will you go? ").lower()
    if re.search("^(n)|^(north)", command):
        fangorn()
    elif re.search("^(s)|^(south)", command):
        print "You can't go South from Helms Deep"
        helmsdeep()
    elif re.search("^(e)|^(east)", command):
        edoras()
    elif re.search("^(w)|^(west)", command):
        gapOfRohan()
    else:
        print "Please enter either north(n), south(s), east(e) or west(w)"
        helmsdeep()

def edoras():
    command = raw_input("You are at Edoras, which direction will you go? ").lower()
    if re.search("^(n)|^(north)", command):
        fangorn()
    elif re.search("^(s)|^(south)", command):
        print "You can't go South from Helms Deep"
        helmsdeep()
    elif re.search("^(e)|^(east)", command):
        minasTirith()
    elif re.search("^(w)|^(west)", command):
        print "You can't go West from Edoras"
        edoras()
    else:
        print "Please enter either north(n), south(s), east(e) or west(w)"
        edoras()

def fangorn():
    command = raw_input("You are in Fangorn Forest, which direction will you go? ").lowe
    if re.search("^(n)|^(north)", command):
        lorien()
    elif re.search("^(s)|^(south)", command):
        helmsdeep()
    elif re.search("^(e)|^(east)", command):
        anduin()
    elif re.search("^(w)|^(west)", command):
        isengard()
    else:
        print "Please enter either north(n), south(s), east(e) or west(w)"
```

```
        fangorn()

def minasTirith():
    command = raw_input("You are at Minas Tirith, which direction will you go? ").lower(
    if re.search("^(n)|^(north)", command):
        deadmarshes()
    elif re.search("^(s)|^(south)", command):
        print "You can't go South from Minas Tirith"
        minasTirith()
    elif re.search("^(e)|^(east)", command):
        osgiliath()
    elif re.search("^(w)|^(west)", command):
        edoras()
    else:
        print "Please enter either north(n), south(s), east(e) or west(w)"
        minasTirith()

def highPass():
    command = raw_input("You are at the High Pass, which direction will you go? ").lower
    if re.search("^(n)|^(north)", command):
        print "You can't go North in the High Pass"
        highPass()
    elif re.search("^(s)|^(south)", command):
        lorien()
    elif re.search("^(e)|^(east)", command):
        mirkwood()
    elif re.search("^(w)|^(west)", command):
        rivendell()
    else:
        print "Please enter either north(n), south(s), east(e) or west(w)"
        highPass()

def mirkwood():
    command = raw_input("You are in Mirkwood Forest, which direction will you go? ").low
    if re.search("^(n)|^(north)", command):
        erebor()
    elif re.search("^(s)|^(south)", command):
        print "You can't go South in Mirkwood"
        mirkwood()
    elif re.search("^(e)|^(east)", command):
        print "You can't go East in Mirkwood"
        mirkwood()
    elif re.search("^(w)|^(west)", command):
        highPass()
    else:
        print "Please enter either north(n), south(s), east(e) or west(w)"
        mirkwood()

def erebor():
    #special event: when you first enter Erebor you earn bonus points then you can carry
    command = raw_input("You have found Erebor, you earn bonus points! Which direction
    if re.search("^(n)|^(north)", command):
        print "You can't go North in Erebor"
        erebor()
    elif re.search("^(s)|^(south)", command):
        mirkwood()
    elif re.search("^(e)|^(east)", command):
        print "You can't go East in Erebor"
```

```
        erebor()
    elif re.search("^(w)|^(west)", command):
        print "You can't go West in Erebor"
        erebor()
    else:
        print "Please enter either north(n), south(s), east(e) or west(w)"
        mirkwood()
```

Table 1: Testing North Input on the Fangorn Node

Test	**User Input**	**Expected Output**	**Actual Output**	**Status**
When the full location in lower case is inputted by the user	north	"You are in the forests of Lorien"	"You are in the forests of Lorien"	Pass
When the shortcut location in lower case is inputted by the user	n	"You are in the forests of Lorien"	"You are in the forests of Lorien"	Pass
When the full location in upper case is inputted by the user	NORTH	"You are in the forests of Lorien"	"You are in the forests of Lorien"	Pass
When the shortcut location in upper case is inputted by the user	N	"You are in the forests of Lorien"	"You are in the forests of Lorien"	Pass
When the full location is inputted with a capital	North	"You are in the forests of Lorien"	"You are in the forests of Lorien"	Pass
When the full location is inputted with a mixture of capitals	nOrtH	"You are in the forests of Lorien"	"You are in the forests of Lorien"	Pass
When something else in inputted	nortd	"Please enter either north(n), south(s), east(e) or west(w)"	"Please enter either north(n), south(s), east(e) or west(w)"	Pass
When null is inputted	{null}	Please enter either north(n), south(s), east(e) or west(w)	Please enter either north(n), south(s), east(e) or west(w)	Pass

Table 2: Testing South Input on the Fangorn Node

Test	**User Input**	**Expected Output**	**Actual Output**	**Status**
When the full location in lower case is inputted by the user	south	"You are at Helms Deep"	"You are at Helms Deep"	Pass
When the shortcut location in lower case is inputted by the user	s	"You are at Helms Deep"	"You are at Helms Deep"	Pass
When the full location in upper case is inputted by the user	SOUTH	"You are at Helms Deep"	"You are at Helms Deep"	Pass
When the shortcut location in upper case is inputted by the user	S	"You are at Helms Deep"	"You are at Helms Deep"	Pass
When the full location is inputted with a capital	South	"You are at Helms Deep"	"You are at Helms Deep"	Pass
When the full location is inputted with a mixture of capitals	soUtH	"You are at Helms Deep"	"You are at Helms Deep"	Pass
When something else is inputted	sou	Please enter either north(n), south(s), east(e) or west(w)	Please enter either north(n), south(s), east(e) or west(w)	Pass
When null is inputted	{null}	Please enter either north(n), south(s), east(e) or west(w)	Please enter either north(n), south(s), east(e) or west(w)	Pass

Table 3: Testing East Input on Fangorn Node

Test	**User Input**	**Expected Output**	**Actual Output**	**Status**
When the full location in lower case is inputted by the user	east	You are following Anduin (The Great River)"	You are following Anduin (The Great River)"	Pass
When the shortcut location in lower case is inputted by the user	e	You are following Anduin (The Great River)"	You are following Anduin (The Great River)"	Pass
When the full location in upper case is inputted by the user	EAST	You are following Anduin (The Great River)"	You are following Anduin (The Great River)"	Pass
When the shortcut location in upper case is inputted by the user	E	You are following Anduin (The Great River)"	You are following Anduin (The Great River)"	Pass
When the full location is inputted with a capital	East	You are following Anduin (The Great River)"	You are following Anduin (The Great River)"	Pass
When the full location is inputted with a mixture of capitals	eAsT	You are following Anduin (The Great River)"	You are following Anduin (The Great River)"	Pass
When something else is inputted	est	Please enter either north(n), south(s), east(e) or west(w)	Please enter either north(n), south(s), east(e) or west(w)	Pass
When null is inputted	{null}	Please enter either north(n), south(s), east(e) or west(w)	Please enter either north(n), south(s), east(e) or west(w)	Pass

Table 4: Testing West Input on Fangorn Node

Test	**User Input**	**Expected Output**	**Actual Output**	**Status**
When the full location in lower case is inputted by the user	west	"You are in Isengard"	"You are in Isengard"	Pass
When the shortcut location in lower case is inputted by the user	w	"You are in Isengard"	"You are in Isengard"	Pass
When the full location in upper case is inputted by the user	WEST	"You are in Isengard"	"You are in Isengard"	Pass
When the shortcut location in upper case is inputted by the user	W	"You are in Isengard"	"You are in Isengard"	Pass
When the full location is inputted with a capital	West	"You are in Isengard"	"You are in Isengard"	Pass
When the full location is inputted with a mixture of capitals	weST	"You are in Isengard"	"You are in Isengard"	Pass
When something else is inputted	we	Please enter either north(n), south(s), east(e) or west(w)	Please enter either north(n), south(s), east(e) or west(w)	Pass
When null is inputted	{null}	Please enter either north(n), south(s), east(e) or west(w)	Please enter either north(n), south(s), east(e) or west(w)	Pass

Table 5: Testing Output on Boundary and Random Nodes as well as game state conditions

Test	**Node**	**Expected Output**	**Actual Output**	**Status**
Going North from a north boundary node	The Shire	"You can't go North from The Shire"	"You can't go North from The Shire"	Pass
Going South from a south boundary node	Helms Deep	"You can't go South from Helm's Deep"	"You can't go South from Helm's Deep"	Pass
Going West from a west boundary node	Gap of Rohan	"You can't go West from Gap of Rohan"	"You can't go West from Gap of Rohan"	Pass
Going East from an east boundary node	Mirkwood	"You can't go East in Mirkwood"	"You can't go East in Mirkwood"	Pass
Going North from a random node	Lorien	High Pass	High Pass	Pass
Going South from a random node	Rivendell	Moria West Gate	Moria West Gate	Pass
Going East from a random node	Osgiliath	Minas Morgul	Minas Morgul	Pass
Going West from a random node	Weathertop	Bree	Bree	Pass
Test the win condition	Mordor	"You reached Mordor. You win."	"You reached Mordor. You win."	Pass
Test the Moria lose condition	Moria	"You run into the Balrog and die. Game Over."	"You run into the Balrog and die. Game Over."	Pass
Test the Shelob's Lair lose condition	Shelob's Lair	"You find Shelob and die. Game Over."	"You find Shelob and die. Game Over."	Pass
Test the Erebor special condition	Erebor	"You have found Erebor, you earn bonus points!"	"You have found Erebor, you earn bonus points!"	Pass