The Forest of Lost Souls

The text based fantasy RPG.

Team Members

Connor - Project Lead/Manager
Assigning tasks, organisation, collation of data, formatting and working across multiple roles.

Ian - Story and Theme Creator
Dialogue script writing, thematic choices, characters and writing/ one of the puzzles.

Ifty - Puzzle Coding
Coded one of the puzzles and generally assisted in the development of the whole project.

Nour - Decision Making Builder
Coded the skeleton for decision making and implemented stat points and loss/win conditions.

Ryan - Research and Development
Researched various cosmetic and gameplay options, worked on a class based combat system.

What is the game about?

This is a text-based adventure game made in Python about your brave escape from a haunted forest in Australia, as you enter the forest, you encounter many dangers and mysteries that you need to overcome.

There is only one player character, yourself! Your choices impact your development, these are risky choices that might kill you, or you might gain skills that you can use to escape.

You'll encounter multiple non-player characters; a vampire, a ranger, a witch and a stranger will either help or hinder your progression.

There are over 16 different endings, with roughly 12 routes ending in defeat the odds are stacked against you, be sure to make smart decisions.

How did we plan the game?

Brainstorming sessions

- Discussing the required project.
- Coming up with potential story plots.
- Discussing associated topics and systems which would relate to the story plots.

Themes and Goals

- Agreeing on the Theme of the story plot.
- Discussing and agreeing on pathways and potential outcomes.
- Figuring out what the overall goal of the project would be.

Ideas and Planning

- Create a workflow flowchart which would show the agreed pathways, outcomes and goals.
- Creation of a trello board, which can be dynamically assigned and worked through accordingly.

Assigning Roles

- Discussing the strengths of the individual team members.
- Assigning jobs and roles which suit the team members.

Connor - Project Lead/Manager

Responsible for assigning tasks amongst team members, keeping the Trello board up to date and using it to organise these tasks.

Regularly checking for progress with these tasks and assisting with them if necessary, also ‘reining in’ ideas when they became unrealistic or out of scope.

I was responsible for collating and formatting, everyone would provide their ideas and I would link them all together in one document, such as the flowchart.

This applied to code as well, I took everyone's individual code and combined them into one script, making sure everything worked correctly together.

Ian - Story and Theme Creator

I was assigned to script writing and the witch section, this involved coming up with three riddles for the witch and coding them in to the game.

For script writing I was required to add a more dark fantasy feel to parts of the game and add a dark ambience in this fashion.

After this was done, I advised and guided the other members, to ensure I was still an integral part of the team.

I was also actively involved in several brainstorming sessions.

Ifty - Puzzle Coding

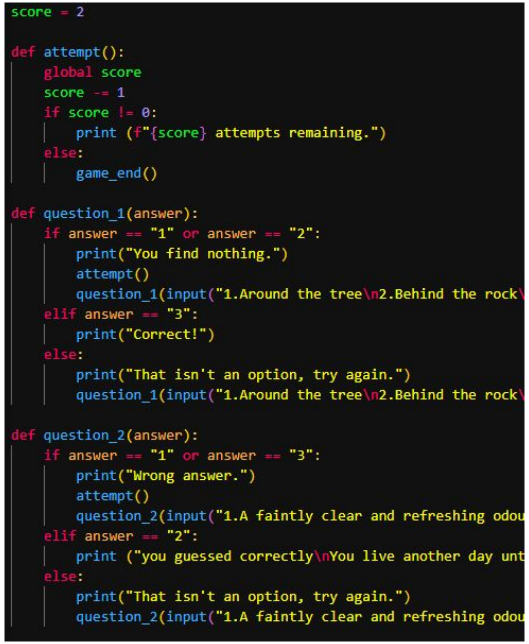
The task I was assigned was coding one of the puzzles.

I assisted in the the various brainstorming sessions we had.

I attempted to research visuals around appearance but was not successful with providing a solution.

I did need to ask for help from various team members in writing/fixing code.

I helped with testing the final version.



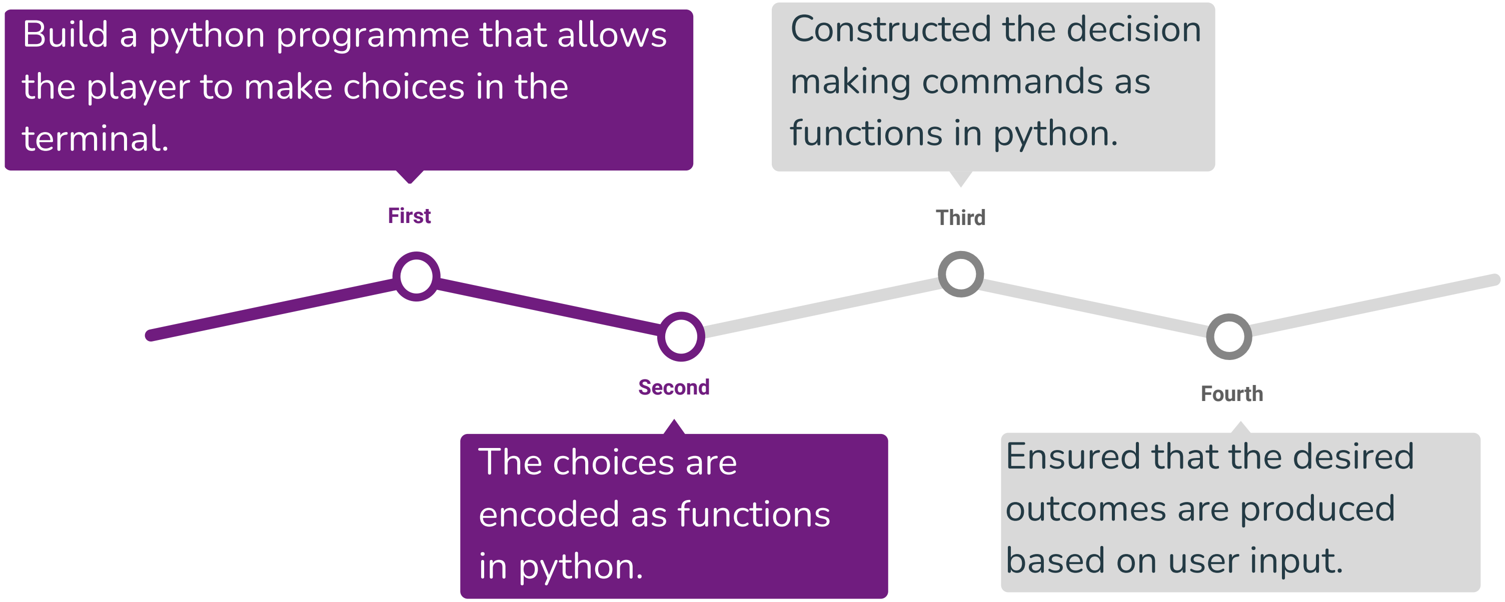
```python
score = 2

def attempt():
    global score
    score -= 1
    if score != 0:
        print (f"{score} attempts remaining.")
    else:
        game_end()

def question_1(answer):
    if answer == "1" or answer == "2":
        print("You find nothing.")
        attempt()
        question_1(input("1.Around the tree\n2.Behind the rock\
    elif answer == "3":
        print("Correct!")
    else:
        print("That isn't an option, try again.")
        question_1(input("1.Around the tree\n2.Behind the rock\

def question_2(answer):
    if answer == "1" or answer == "3":
        print("Wrong answer.")
        attempt()
        question_2(input("1.A faintly clear and refreshing odou
    elif answer == "2":
        print ("you guessed correctly\nYou live another day unt
    else:
        print("That isn't an option, try again.")
        question_2(input("1.A faintly clear and refreshing odou
```

Nour - Decision Making Builder



Build a python programme that allows the player to make choices in the terminal.

First

The choices are encoded as functions in python.

Second

Constructed the decision making commands as functions in python.

Third

Ensured that the desired outcomes are produced based on user input.

Fourth

Ryan - Research and Development

Research, Testing, Trial and Error, Flowchart Planning and Visual planning.

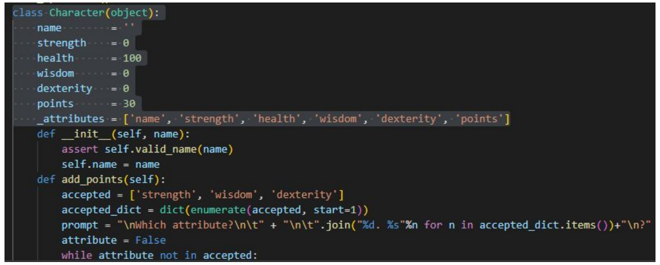
- Colorama, ASCII, PYFiglet, Class, Objects, Dynamic coding.
- PY Modules, Internal and External available modules.
- ASCII plain text conversion for banner and art planning.

Research of more complex structures.

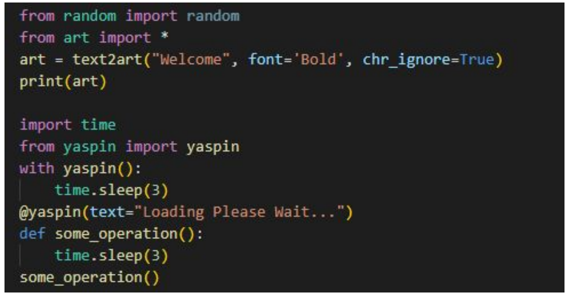
- Class Objects, which would allow an object to be called and dynamically called at any point.
- Potential modules: Yaspin, Text2Art, Unicode_Laterals, Art, PYFiglet.
- Use of websites, such as onlineASCIITools which allowed plain text conversion into ASCII Banner which would allow python to use Type function to print.

Class Objects, External Modules etc.

- Use of Class Objects, to create a dynamic experience, attribute and battle system.
- Determining and discussing scope of the potential use case.
- Liaising and determining the potential function of it.



```
class Character(object):
    name        = ''
    strength    = 0
    health      = 100
    wisdom      = 0
    dexterity   = 0
    points      = 30
    _attributes = ['name', 'strength', 'health', 'wisdom', 'dexterity', 'points']
    def __init__(self, name):
        assert self.valid_name(name)
        self.name = name
    def add_points(self):
        accepted = ['strength', 'wisdom', 'dexterity']
        accepted_dict = dict(enumerate(accepted, start=1))
        prompt = "\nWhich attribute?\n\t" + "\n\t".join("%d. %s"%n for n in accepted_dict.items())+"\n?"
        attribute = False
        while attribute not in accepted:
```



```
from random import random
from art import *
art = text2art("Welcome", font='Bold', chr_ignore=True)
print(art)

import time
from yaspin import yaspin
with yaspin():
    time.sleep(3)
@yaspin(text="Loading Please Wait...")
def some_operation():
    time.sleep(3)
some_operation()
```

What if we incorporated frontend?

If we incorporated frontend to the Python code, the game would have been much more aesthetically pleasing but would have taken far too long to produce.

It would have required people to learn new coding languages specific to front end, taking even more time.

It could not have been ready in only two days, as we would need time to integrate the front and back end elements with each other and produce both ends.

Trello

Advantages

- ❖ Easy to use and follow.
- ❖ Integrative tool that allows participation from all team members.
- ❖ Allows members to follow the progress of the work from different tasks.

Disadvantages

- ❖ Constant email notifications throughout the day.
- ❖ With a larger project the amount of cards would become unwieldy and hard to manage.

Slack

Advantages

- Allows for quick exchange of information among team members through messages.
- Allows the exchange of code among team members using code blocks, helps with debugging and improves the code.

Disadvantages

- Constant email notifications throughout the day.
- Addictive due to constant distractions with messages.

Working in a Team

We found the process of working in a team to be highly advantageous as we could prioritise our strongest points whilst covering each others weakest points.

We did not use official stand ups, but there was a morning discussion that loosely planned out the day ahead.

The first day was quite disorganised from a team standpoint, no clear project manager was assigned and tasks were quite fluid, on the second day we discussed this and assigned a definitive team leader to speed up the process.

We found using the Kanban method on Trello very useful in the context of teamwork, as we could all see who was assigned to which role.

The game was coded piecemeal by all members of the team independently, these elements were then integrated into one main project programme.

Testing

Everyone tested their section of code to ensure that it functioned correctly, then once each part was integrated together the entire programme was tested for functionality.

There was then an element of user testing where we gave our code to Demi and Yasir (our tutors) to let them test it, as well as all members of the team playing through the game to find bugs and forward them to the project lead to be fixed.

These testing phases were effective in reducing the amount of bugs at each stage, most importantly the user testing phase at the end was critical in removing bugs before we handed in the final code.

What if we did it alone?

Working alone allows for more control over the entire project, including the implementation of ideas and creative elements, it’s all your work so it’s your choice as to how everything works, in this respect it could result in a good project.

However, working alone will typically take longer as you don’t have the physical ability to match the speed of a full team, as well as reducing the quality of the end product and increasing bugs, especially if you’re not as proficient as other people.

Working in a team enabled us to work with different skill sets, which allowed for lots of creativity, without the team it’s likely these ideas would have been missed, it also helped us to deliver the project within a relatively short timescale.

Overview

The collaborative effort between the team allowed us to create a game that was able to broadcast each individuals skills, it highlighted the strengths of each individual, allowing them to contribute effectively. The outcome was a well polished game without too many bugs, which followed the flowchart accurately and was presented very well. It was also an excellent team building exercise which allowed individuals to gain knowledge and experience in skills which may have needed improvement by discussing and learning from each other.

If we did this again we would research and utilise a larger range of Python's features, such as classes. Understanding more about Python would have allowed the team to build a programme that works more efficiently. When we found bugs in the code it became a trial and error approach to fix them, wasting a lot of time, next time we would ask our tutors for more advice and look into the issues further, to learn more from them and to develop our coding abilities.

Python itself was easy to understand, but using the languages features to produce a game was a difficult task initially, requiring refinement of specific elements which were translated into functions, once this was done the process of connecting the blocks was simple enough but took lot's of time, there's plenty of room for improvement and optimisation.