Name: Pranav Tumminkatti(23) Class: 2L

**Visuals can be deceptive, You need to be the detective**

***Overview of my game***

For this project, I made not just one but three Adventure RPG Games, with the namely Horror in the Pacific, Madness at Hobbsgate and Horror in the Darkness. And yes, you’re right, all of them are based on the much-loved genre of horror. The story texts for these games are obtained from the Text-Based Adventure RPG iPhone apps, Horror in the Pacific, Horror in the Darkness and Madness at Hobbsgate, all three of them created by the same developer, Karmic Shift Studios… Horror in the Pacific is my most developed game out of the 3, simply because this was the only one I had originally planned to work on…I made the other 2 games simply because I felt that I was well-familiarised with the code structure and I did have a lot of time at hand, so I thought, why not make another game like this? After making the second one, I liked it so much that I was even more motivated so, I went on to code a third game after that… Horror in the Pacific is a full-fledged game where there is an end to the game, unlike the other 2 games which end at the first chapter, asking the user to subscribe to the pro version to unlock the rest of the chapters… I could have still gone with the story, however, in the app, as the story progressed, various other functions like talk, listen, investigate became absolutely necessary for the game to progress and trying to code these functions in a small text-based game would indeed have been a very daring task for me to attempt… Despite this, I tried coding the talk function in the game, but failed miserably due to the problem of endless possibilities. What do I mean by that? Put very simply, the player can type in whatever he/she wants to say to the character. I didn’t know how to code the game such that the system would “anticipate” the user’s input. Eg. If user enters\*\*\*\*\*: print(‘ok’). The \*\*\*\*\* had endless possibilities to it so, I didn’t think and I still don’t think that it is a good idea to hard-code the result of every single possible input, thus I ditched the idea of coding that function and ended the 2 games at the first chapter… I did indeed face these problems while coding my main game, Horror in the Pacific, however, I resolved it by creating my own story to end the game/kill the player. (Example: in the app, after the player enters the Maintenance Shed, he talks to a workman who tells him the password to unlock the door to the beach, from where the story progresses, however, in my game, I wrote out my own story about the toxic gas which kills the player when he enters the Maintenance Shed…). There is only 1 “winning”/“surviving” state in the game which is after you successfully decode the message and diffuse the bomb.

***Horror in the Darkness***

This story revolves around a former army man, now detective, who’s late comrade, Tommy’s widow, Annabelle, needed some help. Something about it seems really off to him and he decides to travel to Annabelle’s house, Wentsire Manor, located on a small island off the coast of Brunswick, Maine. A few hours and a short boat ride out there to make sure everything is fine, then back home by supper, or so he thought…

***Fresh from the app*:** Fresh back from the horrors of World War 2, the protagonist was working as a private investigator. Not hugely successful, but it paid the bills. Mostly. But when the widow of his old friend needed his help, nothing could have prepared him for the bizarre and uncanny events that were about to unfold at the haunted Wentsire Manor…

The game combines a bleak environment and a sense of underlying dread with moments of comedy to keep you entertained…

***Horror in the Pacific***

This game revolves on a crime detective who is assigned to the case of a few tourists who had disappeared from a resort on Malaki Island near Papua New Guinea…After landing on the airstrip, he is offered several different choices to make, with death lurking right behind him and success, all the way at a bar, in a resort…

***Fresh from the app:*** As the third game in the “Horror in the Darkness” series, Horror in the Pacific takes place several years after Horror at Innsport and Horror in the Darkness. Called once again to investigate a case of a missing person, the protagonist is drawn to a small resort island in the Pacific, to uncover an all new string of macabre horrors…

***Madness at Hobbsgate***

This game revolves around a girl named Lacy who makes her way into Hobbsgate Asylum for the Criminally Insane, to meet an unknown person whom she repeatedly mentions during her journey. In her journey, she is faced with several different obstacles while getting to the inmate wards to meet that ghost person. In my game, Lacy creeps her way up into the asylum and into the ward filled with patients seeking treatment. She is injected with an unknown fluid which makes her unconscious and when she wakes up, she finds herself locked up in an asylum cell…

***Fresh from the app:*** As the fourth game in the “Horror in the Darkness” series, the protagonist finds herself trapped within the walls of Hobbsgate Asylum for the Criminally Insane. Unsure of how she came to be there, she finds out quickly enough that the line between sane and insane can often be blurry…

***Overview of my code***

***Structure***

I have tried to make my code as clean as possible by following a set structure. I believe that structuring your code makes testing and troubleshooting significantly easier, especially when you have 1500+ lines of code to look through to find the origin of the irritating error. If you have a well structured code, you know where to look at to fix/edit a certain execution of the code.

For my code, this is how I have structured it:

* Name and Class as comments
* Import modules
* Helplines
* Art
* Functions
* Directory for Horror in the Pacific
* Directory for Madness at Hobbsgate
* Directory for Horror in the Darkness
* Beginning Formalities
* Main Code

***User friendliness and playability***

I think this is the main aspect and also the main speciality of my game which sets it apart from the rest, I think? Jokes apart, user friendliness and playability was the main focus of my game throughout my entire coding journey. First of all, I determined my target audience, users who knew python and users who didn’t. I decided to focus on the users who didn’t know python because if the game is user-friendly enough for them then, users who know python should not have any problem with the game. With every line of code that I wrote, I asked myself, “if I didn’t know python, would I understand this?” Using this very technique, I have made my game as user friendly and as convenient for the user as possible and to make this work, I have implemented several different features in my game. Firstly, input validation…It is used everywhere in my code and it helps to check if the user has entered a valid input or not. If the player has entered an invalid input, the system automatically asks for another input. This feature also greatly benefits the code, ensuring that it does not run into any errors because of the input. Put very simply, you are telling the system, exactly what it wants to hear… Secondly, I have included a help function which the user can call and refer to, if he/she forgets the possible moves or instructions. Thirdly, I have tried to make my game seem as realistic as possible. To do this, I firstly used the time.sleep() function to make the outputs feel more timely and realistic. Secondly, I used real time, rational movements. For example, if you can get to the paved path by walking north from the airstrip, then you can get back to the airstrip by walking in the opposite direction, which is to walk south from the paved path. This allows unrestricted movement, as far as possible, in the game… Thirdly, I made use of the type\_print() function in the introduction, which makes it seem like the sentences are being typed. This gives the user the feeling of the game talking to them in real time…Lastly, I also included this coronavirus test for the user, just to make it feel more real, amidst the current pandemic…To test all these features, I asked my parents (part of the user group that doesn’t know python) to play my game and to advise me on how I can improve my game such that it will serve a wider scope of users. They gave me some comments which were overally positive and also told me that it would be good if the inputs were not case sensitive. According to them, it may not be a great deal to ask the user to type in the input in lowercase but having such a feature just reduces the code’s dependence on the user. They told me that it may not feel like a big difference to the user but it sure does make the code feel more capable and reliable, to the user. I pondered over their advice for a while and quickly understood that they were right. Hence, I used the string.lower() function to automatically convert the input to lowercase, thus giving the user the freedom of choice to use any case they like. Therefore, all the inputs, other than the decoded code for the bomb of course, are not case sensitive…

***Elegance***

I used a fair bit of my time to read up about code elegance and I tried my best to make my code as elegant as possible. I referred to the following article to read up about code elegance, <https://medium.com/better-programming/how-to-make-python-programming-more-elegant-and-decent-4b5962695aa9> . In my code, I avoided confusing variable names, all my variables as simply the name of the location itself. I also used a for loop to code repetitive tasks and also stored most of my code as functions to make it seem as minimalistic as possible… Finally, I also tried to make my code as readable as possible such that it is easy to understand, implement and to troubleshoot, for whoever is looking at the code…

***Functions***

I used functions to store repetitive code which was being called several times in the code. Using functions made my code look a lot cleaner and also, the main advantage of using functions is that is there is a problem with this repetitive task, you can simply edit the function instead of editing it hundreds of times in the entire code… I also used functions to store all the directories of each game such that I only had to call 1 function in my main code, which made it a lot neater and also a lot easier to understand. Previously, when all the directories was coded in my main code, I would spend hours tracing the indent level of the if-else statements and this was a huge waste of time. I have yet to explain the biggest reason why I used functions to store the game directory, and I will be explaining this reason in the next section…

The following picture is an example of how I used functions to represent each directory…

***A screenshot of a computer

Description automatically generated***



***Challenges***

As I have stated before, the first dilemma that I was faced with was how can I code all the functions that were absolutely necessary to continue the game story. After a few days of trying over and over again, making versions after versions of code to make the functions work, I concluded that with what I know about python right now, I will not be able to solve the major problem of endless possibilities so, I decided to solve this problem by writing out my own text to end the game. Over the next couple of days, I cam up with several different drafts of stories and finally decided on a few to use in my code. I tried my utmost best to integrate them into the flow of the story to make it seem as smooth-sailing as possible…The second and biggest challenge that I was faced with was that every time the user entered a wrong input, the code would reset its progress all the way back to the first location. With my code majorly focused on user friendliness, this was a huge downside to my game and it was extremely frustrating for me as I could not move on with my story until I had solved that problem. It was really that one thing that was getting on my nerves every single time I sat down to write my code. The complexity of my original code just made my frustration increase…

My original code:

***A screenshot of a computer screen

Description automatically generated***

If, elif and else statements were everywhere and I would sit for hours just to trace and find out where each location was. In the end, I decided to consult Mr Eugene Lim to try and find a solution to this problem. He advised me to code each location as a separate function so that I can call the function whenever the player enters a wrong input. His kind advice changed the entire course of my code literally made it so much easier to understand and to trouble shoot. After having rewritten my entire code, I found it a lot cleaner and also, the structure was a lot more say “constant”. Every location had the same structure. All I had to edit was the name of the function, the story text and what each input would lead to. This very change motivated me to create the 2 other games too. Literally, I was coding directories in minutes and entire games in just a matter of a few hours. The thought of making 2 more games also crossed my mind but I unfortunately couldn’t find any more story texts to code (free at the least, there were several other apps but they all costed a fair bit of money☹…), so I stopped at 3…

***Three Takeaways of this Project***

1. Always save your code and make versions of it if you’re trying something new, lest something goes wrong and you cannot fix it…
2. I learnt how to easily structure my code and to troubleshoot extra-long codes, which is a very important skill for coders to have…
3. I learnt more about user friendliness and playability. I learnt to wear the shoes of my user and to think about how he/she would feel and what he/she would be thinking as they play my game, some of the challenges they will face etc. This really helped me to code my game, suitable for my target audience…

***Solutions***

***Horror in the Pacific***

***A close up of text on a white background

Description automatically generated***

Note: All movements can be reversed too. Example: if you walk north from swamp trail, you will reach airstrip and if you walk south from paved path, you will also reach airstrip etc.

***Madness at Hobbsgate***

***A close up of text on a white background

Description automatically generated***

***Horror in the Darkness***

***A close up of text on a white background

Description automatically generated***

***Test Cases***

If no input is entered, the system automatically asks the user to enter another input…

***A screenshot of a computer

Description automatically generated***

If a invalid input is entered, the system automatically asks the user for another input…

***A screenshot of a computer

Description automatically generated***

Inputs are not case sensitive…

***A screenshot of a computer

Description automatically generated***



If the user enters an invalid input, he is automatically asked for another input…

***A screenshot of a computer

Description automatically generated***

If no input is entered the text is displayed again and the user is once again asked to enter an input…

***A screenshot of a computer

Description automatically generated***

Inputs are not case sensitive and the ask to continue statement accepts both ‘y’ and ‘yes’ for convenience purposes…

***A screenshot of a computer

Description automatically generated***



If the user enters a wrong move, he is told that he can’t walk in that direction. The directory is printed out again and he is asked for another input…

***A screenshot of a computer

Description automatically generated***

Help function for the user’s convenience…

***A screenshot of a computer

Description automatically generated***

The directory is rational and realistic; walk north from airstrip gets you to the paved path and walk south from the paved path leads you back to the airstrip

***A screenshot of a computer

Description automatically generated***

***Hope you liked my submission! ☺***