CS 419 ST/SOFTWARE PROJECTS

Final Report:

HellScape

Christopher King Joseph Fuerst Osbaldo Esquivel

Introduction

As Team Polka, for our project we chose to create an adventure game called HellScape that would run on the command line where the user would descend through the various levels of hell (and other levels) inspired by Dante's *Inferno*. Since all group members were familiar with Python, we chose this as the language to implement the command line game. The first step in creating this game was to collaborate and come up with the actual lore for the game. After discussing different ideas including being trapped on a spaceship, being stranded on a planet, and a haunted house, we decided on a game where the user travels through hell and is also being pursued by the Grim Reaper. This adds an additional layer to the lore of the game and leads to a better user experience as they have to not only solve puzzles along the way, but they also have to interact with the Reaper. Using the Curses library, we were able to include ASCII graphics as required which also enhances the user experience.

The world this game takes place in is comprised of 15 levels with various features and items the user will encounter. Additionally, the user will be able to interact with these items and features and they will be able to store items and get rid of them as needed.

For basic gameplay and the standard ending, the user can expect to take well over 20 minutes to complete this game, but the first time a user plays, it may take much longer.

User Experience

To run the game, the user just needs to type: "python play.py" from within the Game folder. The first time the user runs the game, the size specifications will be shown on screen. Each subsequent running of the program will start the game.

The actual concept of the game includes navigating from room to room and exploring the world around you, using a list of commands that are available by typing "help." Some of these command verbs include the following:

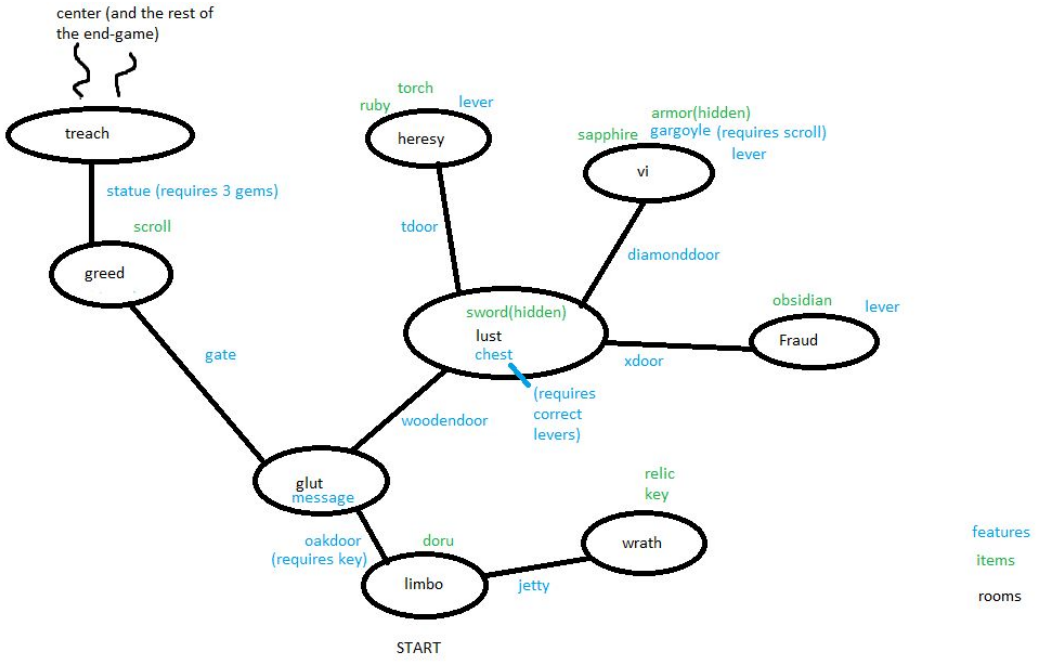
- Look

- Look at (item)
- Look (at a feature)
- Look inventory
- Pick up (item) and the alternate form pickup (item)
- Drop (item)
- Go (to a room)
- Go (to a feature)
- Use (feature)
- Use (item)
- Use (item) on (feature)
- Use (item) on (item) - limited use
- Inventory
- Search (room)
- History (of rooms/features visited)
- Pull (feature)
- Open (feature)

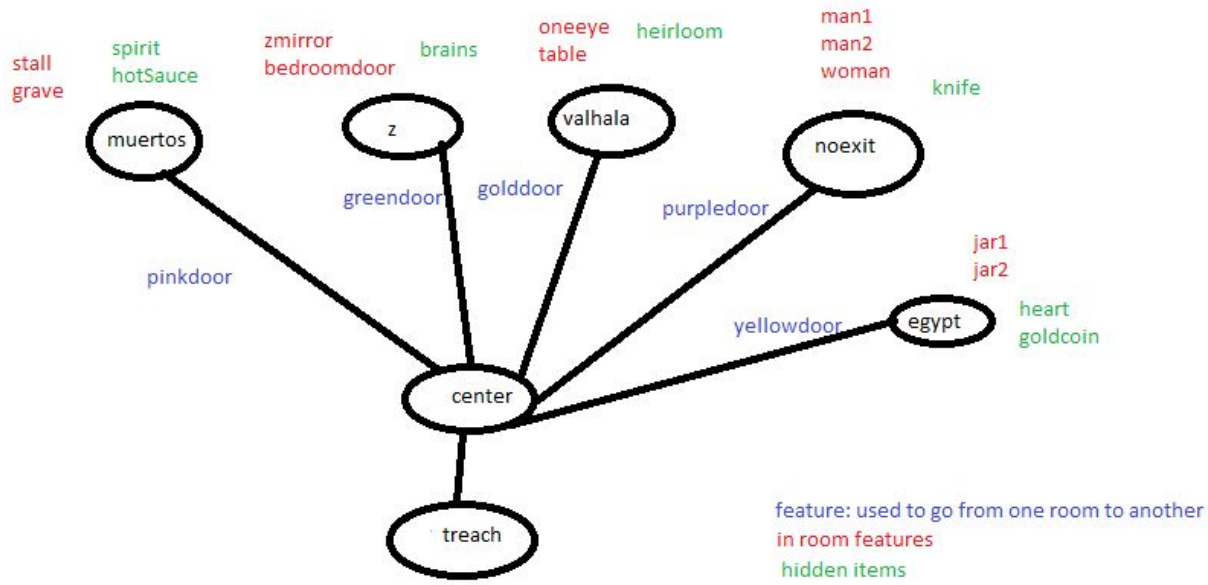
By using the Reaper, we add a sense of pressure to the user as they have to deal with it at certain points in the game. This also forces the user to pay attention to items available that they can potentially pick up to keep in their inventory. These items may be used later on for puzzles or for interactions with the Reaper although it may not be necessary to complete puzzles to advance in the game (this is not relayed to the user, though).

As mentioned, the first time a user plays this game, it may take much longer than 20 minutes to reach the end as the user will have to learn commands and how to navigate from room to room using items. For added complexity, we included different endings so a user may find a new room (or rooms) upon subsequent gameplay. However, one ending in particular is labeled the good ending as we consider it the ideal way to end the game.

Earlygame Structure:



Endgame Structure:



Software/Systems

A listing of which software libraries, languages, APIs, development tools, servers, and other systems you used to create the software, and deeper discussions of any of those you want to talk more about.

We used the python curses library to handle all of our user interface. We made five windows for displaying different types of data- names, pictures, descriptions, error messages, and user input. We stored the room, item, and feature information in text files, used python to read those files, and then fed the information into curses methods to display info on the screen. We used curses in addition to python's sleep functionality in order to make animations for the reaper and for the 'workingbrain' event.

We used the Big Huge Thesaurus API in order to allow users more flexibility in entering verbs. The way we utilized this is to create a list of lists representing lists of synonyms for each of our verbs. When the user submits input, we first check if the verb used is one of our hardcoded ones or a synonym. If it is, we perform the requisite action. If not, we make a call to the Big Huge Thesaurus API to find synonyms for the word the user entered. We then check the synonyms of the user input against all of the hardcoded verbs and *their* synonyms. If we find a match, we append the word to the list of the synonym that matched, and perform the requisite action. (Note: with a free account we are limited to 1000 calls to the API a day. If that limit is exceeded, the synonym functionality will cease to work and "not valid command" will be displayed for any verb not in our hardcoded list. Each user input makes at most one call to the API, so this probably won't be an issue, although we did go over once in testing.)

Team Member Contributions

Chris:

- Wrote all of the code for curses- creating windows, adding strings, changing colors, refreshing, user input, etc.
- Wrote functions for reading .txt name, description, and pic files and displaying the contents in the windows.
- Set up Big Huge Thesaurus API account and created our app thesaurus functionalty in thesaurus.py and incorporated it's use in function 'interpret'
- Wrote functions to display messages, errors, and animations ('specialMessage') to the screen
- Wrote <name>short.txt files for all rooms and added functionality to remember rooms visited and display short text as appropriate.
- Created our two animated events- reaper and workingbrain
- Wrote 'inventory' and 'history' verb functions

- Wrote code for handling special events from room center onwards- items appearing when user looks at a certain feature, player using items on features, player acquiring all items needed for good ending (see good ending walkthrough for complete list of special events.)
- Created the images, names, and pics for rooms ‘center’, ‘v’, ‘muertos’, ‘valhala’, ‘noexit’, ‘egypt’, ‘pinkending’, ‘greenending’, ‘goldending’, ‘purpleending’, ‘yellowending’, and ‘goodending’ as well as all items and features in those rooms.

Joe:

- Designed and implemented base text parsing and gameplay engine, including format of interpret and fightInterpret methods for interpreting input.
- Wrote base handling methods for look, use, pickup, drop, search, to handle player, item and feature interactions.
- Designed player, item, and feature classes and subclasses.
- Created Grim Reaper enemy and designed and implemented enemy AI and behavior in relation to the player, included creation of “battle state” when the reaper initiates battle with player.
- Developed door feature to facilitate user movement to and through rooms.
- Designed the functionality of different item types such as “weapon”, “defense”, “heal”, “key”, “gem” etc.
-
- Implemented early game (pre-center room) design and layout.
- Made puzzles and new feature subclasses to facilitate that, such as the chest/lever puzzle, the gem/statue puzzle, and the scroll/gargoyle puzzle.
- Created images and description files for all early game features, as well as some early game items.
- Created image and description files for some rooms, including treachery, heresy, violence, and fraud.
- Tested extensively for crashes and gameplay bugs
- Designed and helped implement gameplay changes like limited weapon uses, inventory space, and player health.

David:

- Assisted in designing the room object which is used throughout the code
- Assisted in designing the item object used in the code
- Created several room descriptions including limbo, lust, greed, wrath, and gluttony
- Created ASCII art for the rooms listed
- Created room short descriptions for listed rooms
- Created a function to read files that was ultimately not used as a different function was used to work with the game
- Assisted in developing the lore of the game and based part of the lore on Dante’s *Inferno* and part of it on the XBox 360 game with the same name

- Assisted in completing the final report
- Tested gameplay prior to submitting for mid-point check-in for issues like maximizing the terminal screen to see gameplay and checking for bugs

Walkthrough

Begin the game and name your character.
Type: “look” to look at where you are
Type “search” to search the room for items and features
Type “pickup doru” to pick up the spear.

Note: If at any time the reaper finds you, you can either run, which may fail, attack it with a weapon (“use doru” is the command you want here, but watch out, because it breaks after 4 uses) or if you have something to repel it, it will run off before attacking you. If you get attacked too much, you will die and get a game over, so do be cautious. Also, use the above maps to get a better feel for where you are at.

Note about puzzles: There are a few puzzles in the first part of the game that are not necessary to complete, but offer additional rewards to the player such as items. If you want to try them out, first look at the message feature that you find in one of the rooms, and make sure the levers in heresy, violence, and fraud correspond to the arrows on the message. (the doors to these rooms have symbols on them that correlate to each arrow.) use “pull lever” to pull a lever down or up, and once you have the right combination, the chest in lust will open up when you type “open chest” revealing a sword to replace your doru if it breaks. Also the scroll that you “pickup” in greed will heal you if you are hurt by typing “use scroll” . However, if you use the scroll on the gargoyle in Violence (“use scroll on gargoyle”) it will crumble away revealing armor that repels the reaper.

You may type “look doru” to look at it or any other thing in your inventory or room.
There is an oak door in this room, but it is locked. If you type “use oakdoor”, it will tell you so.
Type “go jetty” then “use jetty” to take the boat to a new room.
You should now be in Wrath. Type “search”. Type “pickup key” to pick up the key in this room, as it goes to the oak door in the other room. You may want to pick up some other items in here as well. They may be useful later.
Return to the first room (“use jetty”) In order to unlock the door type “use key on oakdoor” and the door will unlock. At this point, you can type “use oakdoor” to get into the next room. Now you are in gluttony. There are three doors here, and also a message on the wall. Use “look message” to read that, and then “use woodendoor”. Ignore the gate for now. This next room connects to three new rooms each with an interesting door. Look at the doors and notice they correlate to the message in the previous room. Type the following commands in this order to pick up the three gems that you will need later:

1. Use tdoor
2. Pickup ruby
3. Use tdoor
4. Use diamonddoor
5. Pickup sapphire(you may have to “drop” an item in your inventory, but you can come back for them later. They will remain where you dropped them)
6. Use diamonddoor
7. Use xdoor
8. pickup obsidian

Now with the three gems in your inventory:

9. Go glut
10. Use gate
11. Use ruby on statue
12. Use sapphire on statue
13. Use obsidian on statue
14. Use statue

Now you are in the bottom level of hell, and if you use the evildoor, there is no going back. So if you want to figure out how some of the puzzles of the previous portion that we did not cover work, you may want to go back and do that now.

15. Use evildoor

Now you are at the center.

From center:

At this point, the user can inspect various afterlifes to spend eternity in (codenamed ‘pink’, ‘green’, ‘gold’, ‘purple’, and ‘yellow’). There is also a hidden “good ending”.

The player can then type use <color>door (ie use yellowdoor) to inspect the given afterlifes. Each afterlife has a different theme and has features the player can inspect to know more about that afterlife. There are also hidden items in the rooms that are needed to achieve the good ending.

Once the player has decided which afterlife they want to go to, from room center type: “go <color>” to be taken to an ending. (Note: if the player is impatient or doesn’t care where they spend the rest of their afterlife, they can type “go <color>” from center before going to any or all of the possible afterlifes.)

A sample walkthrough of a regular ending:

1. use pinkdoor
2. look/go stall - *Player discovers this is a Dia De Los Muertos themed afterlife*
3. look - *player gets short description of room*
4. search- *player finds there is a ‘grave’ feature in room*
5. look/go grave

6. use pinkdoor
7. use greendoor
8. search - *player finds zmirror and bedroomdoor features in room*
9. look/go zmirror - *player discovers they are a zombie*
10. look/go bedroomdoor - *player kills apartment occupant by fright*
11. use greendoor
12. use golddoor
13. search - *player finds features named oneeye and table in room*
14. look/go oneeye -*player talks to Odin and gets to know about the afterlife*
15. look/go table- *player gets more info about Valhala*
16. use golddoor
17. use purpledoor
18. look/go man1/man2/woman -*player "talks" to inhabitants of room*
19. use purpledoor
20. use yellowdoor
21. look/go jar1/jar2 -*Player inspects what is inside contents of jars*
22. use yellowdoor
23. At this point, the user has inspected all of the afterlives, and are back at center. They then type "go <color>" and are taken to the corresponding end room.
24. User is prompted to type exit to quit program

Steps to achieve good ending from center (search and unneeded look commands left out but may be useful for player to know what to do):

1. use yellowdoor
2. look/go jar1 -*Causes heart to be dropped into room*
3. pick up heart
4. look/go jar2 - *Causes goldcoin to be dropped into room*
5. pick up goldcoin
6. use yellowdoor
7. use greendoor
8. look/go bedroomdoor - *Causes brains to be dropped into room*
9. pick up brains
10. use greendoor
11. use purpledoor
12. look/go woman - *Causes knife to be dropped into room*
13. pick up knife
14. use purpledoor
15. use golddoor
16. look/go table
17. use knife on table - *Causes heirloom to be dropped into room*
18. pick up heirloom
19. use golddoor
20. use pinkdoor

21. look/go stall
22. use goldcoin on stall -*Gives player hotSauce*
23. use hotSauce on brains (or ‘use brains on hotSauce’) -*Combine to give player workingbrain*
24. look/go stall
25. use heirloom on stall - *Gives player spirit*

At this point, the user has collected the heart, workingbrain, and spirit, and will be taken to the goodending room.

Closing

Overall, the game should provide the user with at least 20 minutes of gameplay in an *Inferno*-inspired world. The Reaper chasing the user and the ability to interact with items and features in the various levels adds complexity and a sense of adventure as the user attempts to reach the final level. Additional endings other than the “good” ending allow for a different adventure with subsequent gameplay so users will be drawn to playing over and over to discover new rooms, items, and features.