CLI RPG creator

Anyone can make an RPG! Kind of. It’s more like interactive fiction.

This is a program that parses text files containing character dialogue into lines which can be progressively displayed in the CLI like a simple text-only RPG.

(If this turns out well, there may be a GUI version. It’ll keep to the spirit of being easy to use, any dummy will be able to learn it in 30mins. And without knowing how to code at all. But it’ll run like a text/visual novel RPG, with no fighting :( )

Targets those who want to make a simple text-only dialogue-based RPG with no visuals, perhaps as a partial prototype for a full game, or just for fun. It can be used to obtain feedback on the storyline before spending time on other things like art assets and actual game coding. Includes VN features like branching.

NFRs:

* Should be no-frills, have as few dependencies as possible.
* Easy-to-use interface. Don’t need to add any new code, don’t even need to plug things together. Just fill in the dialogue and branch locations in text files.
* The user should not need to modify the code too much. The main input from the user should be the text files with the scripted dialogues, which will need to be formatted correctly.

Features

BASIC

The most basic dialogue parsing.

Parsing of text into character dialogues which can be displayed in sequence.

* User puts text file with dialogue in correct format into folder in the repo
* When the game starts, the program reads the file line by line, displaying the next line when the user presses enter
* Each line is prefixed with the character’s name (person who is talking)
* Example input format:
* [character A] Hello. How are you?
* [character B] I am fine. What about you?

Storing of commonly used character names in shortened form

* User can store commonly used character names in a separate file and map them to a single character (A,B,C,...) or a shortened acronym
* This is to save typing time when scripting the dialogues
* Also, in case a character’s name is changed, the user does not need to manually replace all the fields in all the text files
* Can implement as a hash map

Custom prefixes

* User can set up custom prefixes i.e. in the game, instead of printing the default [character name] followed by the dialogue, instead print another string, or add some () to the string to indicate actions, etc.
* Example input format:
* /sound/ Knock knock.
* Which prints:
* \*\*Knock knock.\*\*

BRANCHING

Deals with allowing branching storylines that diverge based on certain user inputs.

Prompting for player response

* Gets player response to decide on the branch to go to
* This feature can also be used for any other function, such as asking for confirmation of a chapter switch
* Example input format:
* [character name] Will you accept the candy?
* (ask) Enter yes/no. <— this part can be left empty if there’s nothing to prompt.

Branching involving just a few lines before convergence back to the main branch

* When the player’s response causes the dialogue to branch by just a few lines before becoming standard again
* Example input format:
* (branch) /a yes /r branch 1 /a no /r branch 2 /a idk /r branch 3
* branch 1
* [character name] dialogue
* .... more dialogue for branch 1
* branch 2
* ...dialogue for branch 2
* branch 3
* ...dialogue for branch 3

Branching involving massive divergence from the main branch

When the player’s response determines a big section of the upcoming dialogue, it makes sense to exit the current text file and load a new file instead. Unless the current file is main.txt, the branch should return to the file it diverged from upon reaching the end of the file.

Can be implemented by tracking the line number (not ideal as editing can screw up stuff), or searching for the unique jump ID in the file... can find ways to directly store the pointer if possible

Example input format:

(jump) branch\_A.txt

Possibly (jump ID1) branch\_A.txt

CHAPTER TRANSITION

Features related to the division of the RPG story into chapters.

Chapters

Can be organized into folders, with each folder having a main.txt

Changing chapters

At the end of the file for a chapter, the program’s default behavior should be to return to the selection menu (if it exists), or directly exit the program

If the user wishes to jump to the next chapter instead, this should be specified at the end of the text file for the current chapter

Example input:

(chapter) folder\_chapter2

Chapter selection menu

Allows the player to select a chapter to start from instead of starting from the very beginning

This probably requires the user to modify the actual code... but just some constants and file names. It’s easier that way.

SAVE AND LOAD

Deals with saving player progression and loading it upon initialization

Save and load chapter progression

Save and load file progression

Reset all progress

USER INTERFACE

Additional UI features not integral to the main story.

Start menu

* A message shown when the game starts
* For ease of use, this can also be done by reading from a text file instead of editing within the code
* Can prompt for options like chapter select, load progress, exit

Going back to start menu from the midst of a chapter/file

* Type exit instead of enter anytime

SOUND EFFECTS

Not sure if doable at the moment, KIV

Refer to <https://stackoverflow.com/questions/26305/how-can-i-play-sound-in-java>

DEBUGGING MODE

Allows the user to easily test-play the game.

Options to temporarily enable all chapters, or jump to a specific chapter

Basically just package what you would normally have to do by modifying the code into more convenient commands

Can be implemented as an argument when starting the executable

OTHER FEATURES

Other features related to ease of use, improving UI cosmetics, etc.

Italicized, bold text?

Ignoring of tabs (allows indentation of the input dialogue to better visualize branches)

Typewriter kind of effect (text appears character by character instead of the whole line at once), should have adjustable speed (try <https://stackoverflow.com/questions/28740706/display-characters-with-a-delay-between-each-in-java>)

BUILD RELATED ISSUES

Save txt files as binaries before giving to player (need to prevent them from simply clicking and reading the dialogues O: )

Change the file reader to read binaries instead

You could also scramble the text ? Encrypt ? Although not really necessary