Escape the Train

Our game is organized into various library and header files and one main file. We have three libraries--setup.c, location.c, and object.c.--and three header files to go along with them--setup.h, location.h, and object.h. The setup libraries contain void functions for printing the title, printing the backstory, and retrieving the player's name. The setup.h file contains prototypes of these functions so that they can be included in main.c. Similarly, object.c includes a very crucial function, pickup(), that takes in an object type and places it in the player's inventory. The prototype for this function is included in object.h as is the struct definition of the object type. Within this struct are three types: char const *, int location, and int status. The char contains the name of the object, location contains a 0 or a 1 randomly generated to determine if the object will be on the left or right of a train car, and status contains a 0 or 1 depending on if the object has been picked up. Finally, the location.c file is by far the most complicated of all our libraries. It includes printLocation(), advance(), prompt(), combat(), and forward(). Prompt() accepts a location type, asks what side of the train the user wants to search, and returns a 0 or 1 depending on their answer. Forward() takes in the next location and sets it to the current location. Advance() asks the user if they want to try the key they found on the door and either recurses or calls forward() depending on the response. printLocation() simply prints what room the user is in. Finally, the combat() function uses random number generation to determine which enemy will be fought, if the user can flee, and if and how much damage is done. The location.h header contains location.c function prototypes as well as the location struct which contains information about the name, description, and object within the location. All header functions are included in main so that every function and type can be fetched and used. Main.c includes a while loop for iterating through the train cars. Within this loop, calls to prompt() and pickup() are made. Following the conclusion of the while loop, the combat function is called. After its termination, the game ends with the player either beating the boss or dying. The credits are then displayed.

The game consists of three rooms with a final area at the end of the game. Each room and their descriptions are stored in an array. You may advance through the rooms once the pickup() function is called and the obj.status is set to 1 which doubles as our inventory system as once an obj.status is set to 1, it remains like so for the remainder of the game. Each room has a key that must be found either on the left or right side of the carts. By using the rand() function we were able to randomly generate a value which determines what side of the cart the key is on. A player is prompted each time they are to select a side of the cart to search as well as each time that they are to advance towards the next cart. This is all run within a for loop which repeats the same cycle of events whilst changing the details every time the location array is traversed.

Escape the Train is a text-based adventure game that has a fairly rigid structure with some room for variation within. The basic layout of the game is: You start on a train and move through the train cars by choosing to search right or life. By finding the key in each room, you are then permitted to advance between train cars. The keys in each cart are randomly placed on either the right or left side of the carts. Once you have finally escaped the train by finding the gold key, you are then left on the outside of the last cart. Outside you are matched with a randomly selected enemy which is either a goblin or a minotaur and have to fight or flee them in

turn based combat. Depending on a randomly generated value, the player may or may not be able to flee the monster. The damage that the player and monster both output is also generated by random values and there are three endings based on if you defeat the monster, are defeated, or if you and the monster both die.

One bug is that you cannot respond in capital letters. Doing so would just generate a “Sorry I did not understand…” response but it will not terminate the game. Additionally, the text art for the monsters sometimes generates incorrectly and messes up a couple of lines for their display in the terminal but otherwise the art is still understandable.