CS 111 Final Project Self Assessment

Your group should fill out one copy of this form and include it with your assignment when you turn it in.

Group

Who's in your group?

1. Royce Mettler

Goals

Say a few words about what you wanted the game to be like. Note that if you just wanted to write some code so you could get a good grade on the project, it's fine to admit that.

I definitely wanted it to be at least a little bit fun to play. I wanted it to be really interactive, which I think I did to some extent. Every item has specific text for if you pick it up, take it, and drop it, and some items even have text that changes depending on what you do to it. I definitely wanted it to be larger though, but I ran out of time to complete it. I'm the type of person that could spend hundreds of hours on this project, so I had to cut myself off eventually. But I feel like it's pretty good.

Lessons learned

What went right?

Pretty much everything went right. I spent about the first half of the project figuring out how to put every text in the game inside a text box just solely for aesthetic purposes and then the other half actually adding stuff to the game. I think it was a pretty fun challenge because I had to really dig into how the code that was already in the file worked in order to not break anything when adding text boxes. My new types and fields didn't always work on the first try, but after experimenting with them, they all pretty much worked eventually.

What went wrong?

There was only really one thing that didn't go right, and that was that I wanted to add a short intro cutscene using ascii art. However, for some reason, the (sleep) function wasn't working correctly with the ascii art. Instead of displaying a full ascii image and then sleeping for two seconds and then displaying another ascii image, it would seemingly load the first image over a period of two seconds in a series of chunks which wasn't what I wanted. After a ton of experimenting, I eventually scrapped the idea.

What do you wish you knew when you started?

I don't really know honestly because there was a ton of stuff that I had to figure out and look up while I was coding the project. It certainly would have been a lot easier if I had known some of the stuff I had to look up before I started, but it was really fun figuring it out, so I really can't say I wish I had known anything.

Annoying grading bookkeeping

Types

What are the types you added, and what are they for?

1. NPC: A non-player character in the game. I could have maybe implemented this by just using "person" but I wanted the NPC to have functions that I didn't want the player to have, such as talking to the player.
2. Bed: A bed that you can sleep in. If you (sleepin (the bed)), it plays a short text cutscene using (sleep). Pretty basic.
3. Box: Like a prop, but it can hold stuff. There's nothing in the game that's actually a box, rather box was just a name for a prop that can hold stuff. I needed to make a new type for this because I didn't want every box to have "box" in the name; for example, I have a coat rack and a bookshelf that can both hold items but aren't boxes. Therefore I needed the whole "noun" business that prop has.
4. Hay: Hay that the player can sift through to spawn hidden treasures into their inventory. Implementing hay was pretty hard because I had to figure out how to have sifting through the hay initialize new things in the world that weren't there before.
5. Chest: A chest that can only be opened with a specific key.
6. Key: A key that only opens specific chests. For chests and keys I used an ID system, so if the chest has the same ID as the key, it opens.
7. Flashlight: A flashlight that lights up a dark room. If you try to go into a dark room without a flashlight, text will display that says "this room is too dark" and you need to type (back) to exit. If you do have a flashlight that's turned on in your inventory, you can enter these rooms and actually see what's inside.

Fields

What are the fields you added, what types did you add them to, and what are they for?

1. Yours?: Boolean. Yours is a field for all objects that describes if the object is yours. If it is, its description will be "Your object" rather than "a/an object"
2. The?: Boolean. A field for all objects that denotes if the object should be described as "the object" rather than "a/an object." For instance, there's an NPC named the constable, and I wanted him to be "the constable" rather than "a constable."
3. Specific?: Boolean. A field for all objects that describes if they should have a custom text when examined. Rather than just stating the description made up of the adjectives and the noun, if an object is specific? It would state this description followed by a custom text
4. Examine-text: String. A field for all objects that is the specific text to be displayed if an object is specific?.

5. Noun-to-print on room: String. The noun-to-print field from “prop” but I put it on room so I could have rooms that don’t end in “room” like for example the town square.
6. Go-text: String. A field for rooms that is the text to display when you go to a new room.
7. Dark?: Boolean. A field for rooms that tells if the room is too dark and requires a flashlight to see or not.
8. Take-text: string. A field for things that is the text to print when you take this thing.
9. Drop-text: string. A field for things that is the text to print when you drop this thing.
10. Talking?: boolean. A field for NPCs that states if you are talking to them or not.
11. Current: number. A field for NPCs that is the current dialogue option you are on.
12. D1, D2, D3, D4, D5: strings. Five fields for NPCs that are the dialogue options they cycle through.
13. Sleep-text-1, sleep-text-2, sleep-text-3: strings. Three fields for beds that are the strings that are displayed in the text cutscene when you sleep in a bed.
14. Noun-to-print on box: The noun-to-print field from “prop” but now on a box.
15. Treasure: (listof procedure). Treasure is probably the most complicated field I introduced. It is a field for hays that includes a list of a bunch of make-object procedures that are all added to your inventory when you sift through the hay.
16. Chestid: number. An ID for chests. Keys with this same ID can open this chest.
17. Open?: boolean. For chests. Whether or not this chest is open.
18. Opened-examine-text: string. The new examine-text for this chest when it is open.
19. Closed-examine-text: string. The new examine-text for this chest after it has been opened and closed. This can be set to just the original examine-text usually.
20. Keyid: number. An ID for keys.
21. On?: boolean. a field for flashlights that tells whether this flashlight is on or not.

Procedures

What are the procedures you added or significantly modified from their original form, and what are they for?

1. (back): void. A procedure that moves you back to the room you just came from if it’s too dark to see anything in a room. It throws an exception if the room you’re in isn’t dark.
2. Take: thing -> void. I changed take to display some text confirming you took the item and to also display the item’s take-text.
3. Drop: thing -> void. I changed drop to display some text confirming you dropped the item and to also display the item’s drop-text.
4. Put: thing container -> void. I changed put to display some text confirming you put the item in the container and also to display the item’s drop-text.
5. Titlescreenimage: string. An ascii art image to display upon starting the game.
6. Startuptext: string. A string to display upon starting the game, kind of like splash text.
7. Start: void. Runs start-game and a modified version of (look) to display some helpful commands the user might want to try so they have some idea of what to do.
8. Add-your: (listof string) -> (listof string): Adds “your” to the beginning of a list of strings. Similar to add-a-or-an.
9. Add-the: (listof string) -> (listof string): Adds “the” to the beginning of a list of strings.
10. Makesubstring: string -> string. Takes a string of multiple lines separated by \n and makes a new string out of just the last line, unless there are no \ns and in that case just returns the original string.

11. Substrings: string -> (listof string): Takes a string of multiple lines separated by \n and uses makesubstring recursively to generate a list of all of the individual lines in the string.
12. Longest: (listof string) -> string: Returns the longest string in a listof strings.
13. Halfdifference: number number -> number. Returns half the difference between two numbers, rounding down if the number ends in 0.5. I use halfdifference in the process of centering text within a textbox.
14. Centerone: string string -> string. Takes a small string and a big string and makes the small string the same length as the big string by adding space characters onto each side of the string. Therefore, if you were to display the big string and then the small string, the small string would appear centered below the big string.
15. Textbox: string -> void. Prints an ornamental text box around a string and centers the lines of the string in this box. Uses the previous five functions.
16. Textboxonlist: list -> void. The same as textbox but takes a list rather than just a string.
17. Imgbox: string -> void. A different style of textbox. I named it imgbox because I use it on the startup ascii image.
18. Dialoguebox: npc string -> void. A custom textbox for NPC dialogue. It looks like a little guy is saying the text, kind of.

Methods

What are the methods you added or significantly modified from their original form, what types were they added to, and what are they for? Note that if you have three different methods for the same generic procedure, list each one separately.

1. Examine: object -> string. I modified examine to include the examine-text if an object is specific?.
2. Description-word-list: object -> listof string. I modified description-word-list to include “your” or “the” if your? or the? are true for an object.
3. Descriptionthe o: object -> string. A new procedure to disregard the a/an on an object and print a description that starts with the. It still prints your if the object is yours? Though. I use this mainly for when you go through a door so it says “you go through the door” rather than “you go through a door.”
4. Lookdesc: object -> string. A modified version of look to specifically say “you are in”.
5. Golookdesc: object -> string. A modified version of look to specifically say “you go to”
6. Capital-description: object -> string. A procedure that capitalizes the first letter of the description of an object. Looking back on this procedure, there are probably better ways to do it using basic racket operations, but I don’t wanna go back and change it now.
7. Prepare-to-remove!: container thing -> void. I put in a ton of exceptions depending on if I didn’t want stuff removed.
8. Prepare-to-add!: container thing -> void. I put in a ton of exceptions depending on if I didn’t want stuff added.
9. Describe-contents: container -> list. I modified this to use capital-description.
10. Fancy-describe-contents: container -> list. It’s just describe-contents but it puts ornamental - signs before and after each element of the list.
11. Contents: container -> void. A procedure that prints the contents of a container. It throws a couple exceptions if you try to check the contents of stuff you can’t store stuff in such as doors, an exception if you try to check the contents of a closed chest, and not an

exception but just a regular message if you try to check the contents of a room that's too dark to see.

12. Noun: room -> string. The noun of the room.
13. Canyousee?: room -> boolean. A procedure that checks whether you can see in a room. If the room is dark and you either don't have a flashlight or have a flashlight but it's off, this will return false.
14. Prepare-to-move!: thing container -> void. I put a ton of exceptions for stuff you can't move such as doors or people.
15. Talk: NPC -> void. The way you initiate and partake in dialogue with an NPC. It will display the next dialogue option in the cycle. If you are already on the fifth one, it will keep displaying the fifth one until you stop talking to the NPC and then if you start talking again, it will go back to the first one.
16. Stop-talking: NPC -> void. The way you stop talking to an NPC.
17. Sleepin: Bed -> void. Plays a short text cutscene when you sleep in a bed.
18. Sift: hay -> void. Sifts through hay and adds the treasure to your inventory.
19. Open: chest key -> void. Opens a chest with a key, or throws an exception if the key is incorrect or if the chest is already open.
20. Close: chest key -> void. Closes a chest with a key, or throws an exception if the key is wrong or if the chest is already closed.
21. Switch: flashlight -> void. Switches the flashlight from off to on or from on to off. If you're turning on the flashlight in a dark room, also (look)s around the room.

Total stuff ~~we~~ I built

Write the total number of items listed above.