Ahmad El-Bobou

12-05-2018

Fall 2018

CS 162

Final Project Reflection Doc

Design

In the game, you will play as a frazzled 162 student who has a tight deadline that they need to fulfill. They've waited last minute to finish their final project and now have to resolve a number of issues in order to submit their project. Each issue must be resolved by a certain space.The container inventory takes the form of a checklist. Each time you resolve an issue, your checklist gets one more issue added to it. When your checklist has 11 issues. You have resolved them all. The time is based in interaction with spaces. Whether you interact with spaces where your issues can be resolved or not, your time will go down as a result.

The basics classes I will have are Space, NoSpace, TransitionalSpace, ActionSpace, Issue, Game. The Space is an abstract class which the other space classes are based on. The NoSpace class represents the Out of Bounds of the map. The TransitionalSpaces are where the player moves to other spaces. If you interact with these, your time is wasted. The ActionSpaces are where the issues can be resolved. The Issue class is made to hold the string that describes the issue and a integer that gives the issue an ID. Each action space can solve a certain ID number. These spaces will be made to compare this number when the player uses the interact action when in an action space.

The basic actions a user will have during each step of the game are to move, to interact with the space, to display the checklist, or to quit. If a person chooses to move to a space that isn't displayed on the board, they go into a no space, lose time, and are resent back into the first space, the desktop. The spaces of the game are as follows: Desktop, browser, CS book, canvas, Stack Overflow, VIM, Slack, and PIazza. The transitional spaces are the desktop and the browser and the others are action spaces.

Testing

Test	Input	Expected outcome	Actual outcome
Input validation functions reused and already tested previously			
Ensure displayMap() displays the map appropriately	Run displayMap	Runs and looks nice	Runs and looks nice
Ensure the quit call quits the game	Select 4 from the menu	Game ends	Game ends
Ensure the move function moves the player in the right direction	Moves the player in all four directions	No errors or crashes and player lands in the same spot as the first one.	No errors or crashes and player lands in the same spot as the first one.
Ensure that when the player goes out of bounds they are reset to the starting position	Move player down twice to fall off the board	Player enters noSpace and is reset to desktop	Player enters noSpace and is reset to desktop
Ensure the printChecklist() only prints issues that are resolved	Resolve three issue, then print Resolve one, the print	3 are displayed 1 is displayed	3 are displayed 1 is displayed
Ensure interact() works with the correct space types	Try on transitional space Try on action space	Get custom trans space msg Get custom act space msg	Get custom trans space msg Get custom act space msg
Ensure game recognizes when the	Play the game and solve all the issues	Get winning message	Get winning message

player wins			
Ensure the game quits when player runs out of time	Lose time by going out of bounds	Game ends and tells player they are out of time	Game ends and tells player they are out of time
Ensure issue numbers are matched with the correct space to be resolved	Solve each issue in the correct space	Win the game	Win the game
Check for memory leaks	Run valgrind	0 errors and no lost memory	0 errors and no lost memory
Compile and run on filp		Works	Works

Reflection

When I started thinking about the type of project I wanted to do, I thought it would be funny to do something meta. I really enjoyed this project because I really felt like it gave me the freedom to come up with my own ideas with just a few requirements. The other projects really felt like making the project for someone else who is bad at explaining. I did run into an issue. When I was planning. I thought it would be possible to check for class types of object. I was going to do this to check if the player was interacting with was an action or transitional space. I was unable to find a way that worked. I ended up just checking the names of the spaces because those were hardcoded by me. Speaking of which, lots of hardcoding went into this game. I think that I can expect that going forward for more creating assignments. It felt strange because text is not really something you can easily modularize, but I guess it’s just something to get used to.