Assignment Journal

Title: Assignment Journal #4

Halie Favron

Student ID: 3673820

COMP 306

November 29, 2024

Table of Contents

Research (See references for citations):

Read Alice in Wonderland armed with highlighter and tab markers

Listened to the Alice in Wonderland audio book

Read wikipedia about Colossal caver adventure

Watched several youtube videos about cave adventure

Tried playing cave adventure (its really hard)

Design Decisions/ Errors and their Solutions

Action Files

Design Decisions:

Storing actions in string/ enum maps

I originally attempted to store the actions as a vector of enums, this worked much better and looked a lot cleaner.

Breaking down core functions for parsing actions from files

I originally wrote this as one giant function, it was very complicated and long so I broke it down. I think this is a much better approach and makes it easier to read.

The searchMap free function

This enabled a lot of code reuse in the control class.

Challenges/ Errors and their Solutions:

I spent the most time writing and editing the core functions for parsing actions, I ran into a lot of errors defining the difference between a category, label, and the specific action words. It just took a lot of time and debugging with print statements to work out the issues.

Control Files

Design Decisions:

Parsing user input

I think the way I structured the way that user input is used in my program is interesting. I personally really like it, it makes it so the user can enter words in any order, which gives them the freedom to structure their sentences however they want. I like that the game class has a vector of integers that correspond the the input categories ie. Keywords, Verbs, Items etc. I am sure there are better ways to go about this, but I am very proud that I came up with this on my own and it works pretty well.

Challenges/ Errors and their Solutions:

The way I structured parseUserInput unfortunately the program reads each word individually so if they enter go north east, the program reads this as go east. I attempted to work out a way to fix this but in the end I ran out of time and decided to include the instruction that words must be conjoined with a hyphen to be read together ie go north-east.

Game Object Files

Design Decisions:

I decided to create GameObject as a base class for the common attributes, belonging to the Items, Locations, and Character classes. These include name, short description, long description.

Item Files

Design Decisions:

Derived from GameObject

It manages the various aspects of in game items, ie item types, respawns, points, etc.

Challenges/ Errors and their Solutions:

The biggest issue I ran into while coding these files was how to enable certain items to respawn and others not to. Fixing this required a multi faced approach. First I needed two attributes, one that kept tract of IF an item should respawn in general, the other to determine whether or not to respawn an item at a given time. It also required modifying things in the Game class and the FileReader free functions

Location Files

Design Decisions:

Derived from GameObject

It manages the various aspects of in game locations, ie. exits, hints, and whether or not a location has previously been entered.

Challenges/ Errors and their Solutions:

I originally had two vectors entrances and exits, but in my game the exits and entrances were virtually the same thing. Therefor I eliminated the entrance vector for simplicity. This worked much better, and made the class significantly less complicated.

Another issue I encountered was with my getExit function the only way I could think of to return an Exit object safely was to throw an out of range exception if the exit wasn't found. This works great, but if I had more time I would rework it, possibly return an index instead. Which I did try to do but I ran into some errors so I ended up keeping my original design instead.

Exit Files

Design Decisions:

The Exit class manages the exits belonging to game locations, ie . directions, keys etc

I decided that the name of the exit would correspond to the location which the exit leads to.

I really like the way the the Exit class stores the direction of the exit.

Challenges/ Errors and their Solutions:

Originally I attempted to store a reference of the location which the exit leads to as a Location instance attribute of the exit. This caused many issues. One was that the exit locations could not be loaded until after the game locations were loaded with required additional function calls and confusion in the loadGame function of the game class. It also created issues with the getExit function, and the enterExit function. I ended up just storing the name of the location, then searching the vector of game locations for the correct new location.

Character Files

Design Decisions:

Derived from GameObject

It manages the various aspects of the characters, ie. friendship score, hit points, attack damage, and it also keeps track of gifts and whether or not they are liked by the character.

Alice Class:

Design Decisions:

The Alice class manages the player, ie. scores, hit points, and attack damage

Challenges/ Errors and their Solutions:

I originally intended to have Alice manage the player and treasure inventory but I ran into issues and ended up making them belong to the Game class instead and I ended up writing Inventory as its own class.

Inventory Class:

Design Decisions:

Manages an vector of items, enables the ability to add, remove, search, and validate items.

Challenges/ Errors and their Solutions:

I needed to do a bit of research in order to get some of my functions to work, for example std::sort, I could not figure out why it wasn't working, finally I realized I was passing compareByItemType() rather than compareByItemType. I read geekssforgeeks comparator.

Another compilation error I ran into here was in my removeFromInventory, I was trying to code the erase line from memory and switched the order of erase and find. I ended up having to look it up, geeksforgeeks How to Remove an Element from a Vector

File Reader Files

Design Decisions:

These free functions were originally a part of the GameObject class but I separated them from it.

I was able to enable code reuse by using template functions and having parseField parse common fields, then calling getSpecializedAttribute which parsed fields specific to the object type.

Challenges/ Errors and their Solutions:

I refactored these functions many many times. In the beginning each GameObject was coded on its own. ie I had separate functions for Items, Characters, and Locations. In the end I used template functions and template specialization, which I learned about from geeksforgeeks Template Specialization. I think this is a much simpler approach.

I was getting errors and I had a very hard time working out what the problem was eventually I learned about explicit instantiation from cppreference and I was able to get the program working again.

Utility Files

Design Decisions:

I decided to have separate files to hold generic free utility functions that could be used by any of the other files as needed.

Game Files

Design Decisions:

The Game class is the most complicated class. It manages and checks the players actions, it manages the locations in the game, and how the user interacts with the characters and items in the game.

I attempted to simplify and reuse as much code as possible, but it is still a very complex class. If I had more time one area I would improve on would be to move as many functions as possible into their corresponding classes. ie. move pickUpItem to the Items class. I attempted to do this but because these even functions are so reliant on other game objects such as location and player inventory, I ended up returning to my previous design.

One thing I simplified that I really like is the overloaded matches functions, this removed a lot of extra code that wasn't being used. Before I added those, I was checking each element in the currentAction vector for matching input, even if the value was null.

I ended up really liking the way that the checking functions removed a lot of code from the location functions, for example, checkPickUpItem allows the player to pick up any item in the current location, I only needed to code extra print statements if I wanted something specific to be printed when the player picks up an item.

The same with checkEnter and enterExit, I love the way that no matter the location the player can check any direction and receive messages whether or not the location is valid and each direction does not need to be coded in the location function.

Challenges/ Errors and their Solutions:

This was by far the most challenging and error prone Class.

The check functions required a lot of testing and refactoring in order to properly validate input and provide the player with correct feedback.

The commonActionChecks function caused a lot of errors. I really wanted a function that could check all of the functions that were common to all of the locations but because some needed customization, ie the function for talking to characters needed to check which character the player wanted to talk to. So in the end I ended up splitting up the functions that didn't require customization and the ones that did.

enterExit took a lot of refactoring because of the way I had originally set up the location objects, with the exit holding a reference to the next location. Once I worked that out it ended up being pretty simple but there were a lot of errors along the way.

Main Files

Design Decisions:

The main function serves as an entry point into the program. Main.cpp also contains the test plan for the entire project. It is a very short file and only contains the creation of a Game object and a call to the gameplay function.

Testing

When designing the test cases, I attempted to consolidate as many test cases as possible due to the sheer number of possible tests I wanted to do in order to ensure the game functioned as desired. Therefore I grouped similar tests under the same header ex. Bad Case (Invalid Gifting Actions) holds possible invalid gifting actions. Some invalid actions have descriptions enclosed in square brackets([ ]) if I felt they needed further clarification. But I tried to make it as concise as possible. I also only included one or two lines of output due to the length of many of the output statements. Example: for Valid Fight, I wrote "The fight begins…" rather than copying the entire fight sequence output.

Issues found during testing

While testing the biggest issue I ran into was linking errors, I spent a lot of time adjusting where header files were declares. Other than that, I did a lot of tweaking to things like NPC fight stats.

Reflection

This is by far the most difficult project I have undertaken so far. I feel like I learned a lot about C++ and various ways to use it. I learned a lot about linking errors and pointers. A big lesson I learned on this project is that there is always a different way to write things. I think I wasted a lot of time refactoring and rewriting things that were already working. Another helpful thing I learned was how to plan a project of this size and how to manage it and keep it organized.

Code References:

cplusplus.com. (n.d.). map::find function. https://cplusplus.com/reference/map/map/find/

cplusplus.com. (n.d.). std::ifstream::failure. https://cplusplus.com/reference/ios/ios/exceptions/

cplusplus.com. (n.d.). Vector out-of-range exception discussion. https://cplusplus.com/forum/beginner/170060/

cppreference.com. (n.d.). Explicit instantiation for supported types. https://en.cppreference.com/w/cpp/language/class_template

cppreference.com. (n.d.). out_of_range exception in C++. https://en.cppreference.com/w/cpp/error/out_of_range

Curious Rambler. (2024, September 14). Why was the Mock Turtle so sad? https://curiousrambler.com/curious-history-of-the-mock-turtle/

GeeksforGeeks. (2024, January 3). Comparator in C++. https://www.geeksforgeeks.org/comparator-in-cpp/

GeeksforGeeks. (2020, May 28). Exception::what() in C++ with examples. https://www.geeksforgeeks.org/exceptionwhat-in-c-with-examples/

GeeksforGeeks. (2021, June 23).exit() codes in C/C++ with examples. https://www.geeksforgeeks.org/exit-codes-in-c-c-with-examples/

GeeksforGeeks. (2024, May 23). How to add a timed delay in C++. https://www.geeksforgeeks.org/how-to-add-timed-delay-in-cpp/

GeeksforGeeks. (2024, May 23). How to generate a random number in a range in C++. https://www.geeksforgeeks.org/how-to-generate-random-number-in-range-in-cpp/

GeeksforGeeks. (2019, August 28). ios manipulators: boolalpha function in C++. https://www.geeksforgeeks.org/ios-manipulators-boolalpha-function-in-c/

GeeksforGeeks. (2024, October 11). Map associative containers – The C++ Standard Template Library (STL). https://www.geeksforgeeks.org/map-associative-containers-the-c-standard-template-library-stl/

GeeksforGeeks. (2025, February 3). map::find() function in C++ STL. https://www.geeksforgeeks.org/map-find-function-in-c-stl/

GeeksforGeeks. (2025, January 11). static_cast in C++. https://www.geeksforgeeks.org/static_cast-in-cpp/

GeeksforGeeks. (2022, August 26). Template specialization in C++. https://www.geeksforgeeks.org/template-specialization-c/

Code References Continued:

GeeksforGeeks. (2024, November 4). Transform to lowercase in C++. https://www.geeksforgeeks.org/how-to-convert-std-string-to-lower-case-in-cpp/

GeeksforGeeks. (2025, January 16). Vector erase in C++ STL. https://www.geeksforgeeks.org/vector-erase-in-cpp-stl/

Research References:

Adventure. (n.d.). In Grack. https://grack.com/demos/adventure/

Carroll, L. (2016). Alice in Wonderland [Audiobook]. Audible Studios.

Carroll, L. (2023). Alice in wonderland: The original 1865 edition with complete illustrations by Sir John Tenniel: A classic novel of Lewis Carroll. Independently published.

GameDemos. (2020, June 15). Colossal Cave Adventure – Full walkthrough [Video]. YouTube. https://www.youtube.com/watch?v=QzJbNUnfA3A

RetroGaming. (2019, September 3). Exploring Colossal Cave Adventure – Retro gaming review [Video]. YouTube. https://www.youtube.com/watch?v=1Svp9R1zz9g

Wikipedia contributors. (n.d.). Colossal cave adventure. In Wikipedia, The Free Encyclopedia. https://en.wikipedia.org/wiki/Colossal_Cave_Adventure

Time Log:

Date: December 15
Hours Spent: 6
Activities Undertaken: Started implementing the Game Object class.

Date: December 16
Hours Spent: 5
Activities Undertaken: Started implementing the character class.

Date: December 17
Hours Spent: 4
Activities Undertaken: Expanded game objects and field specialization

Date: December 17
Hours Spent: 3
Activities Undertaken: Starting GameObject file refactoring.

Date: December 18
Hours Spent: 8
Activities Undertaken: Refactored GameObject files and removed comment blocks.

Date: December 18
Hours Spent: 6
Activities Undertaken: Worked on locations files.

Date: December 19
Hours Spent: 10
Activities Undertaken: Added ability to load exits and entrances to location objects.

Date: December 19
Hours Spent: 2
Activities Undertaken: Cleaned up comments and code for GameObject and Location files.

Date: December 20
Hours Spent: 5
Activities Undertaken: Implemented direction/next-location parsing (debugging print issues).

Date: December 21
Hours Spent: 5
Activities Undertaken: Major changes to GameObjectSet structure.

Date: December 21
Hours Spent: 3
Activities Undertaken: Removed GameObjectSet and created free functions

Date: December 22
Hours Spent: 5
Activities Undertaken: Fixed various classes and the Game class.

Date: December 22
Hours Spent: 3
Activities Undertaken: Removed comments and cleaned up code

Date: December 23
Hours Spent: 4
Activities Undertaken: Made changes to location exits.

Date: December 23
Hours Spent: 5
Activities Undertaken: Added ability to load exits to location objects. Updated location file; removed extra comments.

Date: December 26
Hours Spent: 2
Activities Undertaken: Set exits refactor.

Date: December 26
Hours Spent: 3
Activities Undertaken: Refactoring locations files and utility functions.

Date: December 27
Hours Spent: 8
Activities Undertaken: Major updates; actions files implemented.

Date: December 27
Hours Spent: 3
Activities Undertaken: Rewrote the readFile function (implemented for verb).

Date: December 28
Hours Spent: 5
Activities Undertaken: Rewrote readFile, parseInputFunction, currentAction, and getUserInput.

Date: December 28
Hours Spent: 6
Activities Undertaken: Switch to map data structure.

Date: December 29
Hours Spent: 2
Activities Undertaken: Reorganizing ActionSet.

Date: December 29
Hours Spent: 1
Activities Undertaken: header comments added.

Date: December 30
Hours Spent: 5
Activities Undertaken: Refactoring Actions class and moving input functions.

Date: December 30
Hours Spent: 6
Activities Undertaken: General troubleshooting.

Date: December 31
Hours Spent: 4
Activities Undertaken: General fixes; code now stable again.

Date: December 31
Hours Spent: 3
Activities Undertaken: Split Control from Actions classes.

Date: January 1
Hours Spent: 3
Activities Undertaken: Encountered 4 errors while creating iUI for Game class; exit issues.

Date: January 1
Hours Spent: 3
Activities Undertaken: Main class linker error fixed; cleaning up code and adding comments.

Date: January 2
Hours Spent: 3
Activities Undertaken: Restructuring Actions class to pass current action to Control.

Date: January 2
Hours Spent: 3
Activities Undertaken: Added comments to Action files; and fixed enums.

Date: January 3
Hours Spent: 5
Activities Undertaken: Added matches functions; started building gameplay in GameClass.

Date: January 3
Hours Spent: 4
Activities Undertaken: Started adding content to text files; updating Action files simultaneously.

Date: January 4
Hours Spent: 3
Activities Undertaken: Began implementing an Inventory class.

Date: January 5
Hours Spent: 5
Activities Undertaken: Continued work on the inventory class and gift-giving to characters.

Date: January 5
Hours Spent: 6
Activities Undertaken: Finished gift-giving; building minimum viable product(MVP) for testing.

Date: January 6
Hours Spent: 4
Activities Undertaken: Finished basic MVP: pick up item, give item, unlock door, enter location.

Date: January 6
Hours Spent: 3
Activities Undertaken: Updating exit and location classes.

Date: January 7
Hours Spent: 5
Activities Undertaken: Implemented door search vectors; next: door direction as int.

Date: January 8
Hours Spent: 7
Activities Undertaken: Fixed enterDoor and useKey.

Date: January 8
Hours Spent: 3
Activities Undertaken: Cleaned up code in GameClass, Character, Locations, and Items.

Date: January 9
Hours Spent: 6
Activities Undertaken: Organized and added comments; started reorganizing enums.

Date: January 10
Hours Spent: 1
Activities Undertaken: Finished organizing enums; updated classes accordingly.

Date: January 10
Hours Spent: 2
Activities Undertaken: Moving useKey, storeTreasure, and giveGift to GameClass.

Date: January 10
Hours Spent: 5
Activities Undertaken: Fixed logic/ errors in GameClass.

Date: January 11
Hours Spent: 3
Activities Undertaken: Created multiple keys, introduced a test class for repeated input.

Date: January 11
Hours Spent: 4
Activities Undertaken: Fixed treasure/gift bugs; reorganized GameClass files.

Date: January 12
Hours Spent: 4
Activities Undertaken: Finished organizing GameClass files and organize item files.

Date: January 12
Hours Spent: 2
Activities Undertaken: Cleaned up Inventory aggressively.

Date: January 13
Hours Spent: 2
Activities Undertaken: Renaming Inventory in GameClass to playerInventory.

Date: January 13
Hours Spent: 1
Activities Undertaken: Renamed vectors and updated/ fixed useTreasure.

Date: January 14
Hours Spent: 1
Activities Undertaken: Moved itemType enum and changed “gift” verb

Date: January 14
Hours Spent: 2
Activities Undertaken: Successfully removed unspecified key from inventory.

Date: January 15
Hours Spent: 2
Activities Undertaken: Adjusting getExitDoor exception handling.

Date: January 15
Hours Spent: 6
Activities Undertaken: Handling nonexistent doors; removed getDoor & doesDoorExist.

Date: January 16
Hours Spent: 3
Activities Undertaken: Fixed exceptions; removed getDoorIndex and doesDoorExist.

Date: January 16
Hours Spent: 2
Activities Undertaken: Changed enterDoor to use currentLocation.

Date: January 17
Hours Spent: 10
Activities Undertaken: Encountered errors after changes. Trouble shooting

Date: January 18
Hours Spent: 2
Activities Undertaken: Exploring load game pointers.

Date: January 18
Hours Spent: 1
Activities Undertaken: Added check for already-picked-up items.

Date: January 19
Hours Spent: 10
Activities Undertaken: Implemented "go X direction" and seeInventory/explore, temporarily broke the game.

Date: January 20
Hours Spent: 3
Activities Undertaken: Fixed game, troubleshooting getCurrentLocation pointer fixes.
Suspect nextLocation(Door) is not a pointer.

Date: January 20
Hours Spent: 4
Activities Undertaken: Troubleshooting throws bad_alloc at setEntrances/exits in Location; editing vector search.

Date: January 21
Hours Spent: 7
Activities Undertaken: Still working on bad_alloc; planning to overhauling Door/Location/FileReader for simpler door data.

Date: January 21
Hours Spent: 3
Activities Undertaken: Rewriting Door to store just name/direction; adding a checkExit function.

Date: January 22
Hours Spent: 5
Activities Undertaken: Changed enterDoor implementation and rewrote enterLocation.

Date: January 22
Hours Spent: 3
Activities Undertaken: Cleaned up Location/Door files.

Date: January 23
Hours Spent: 2
Activities Undertaken: Cleaned up Game file code.

Date: January 23
Hours Spent: 4
Activities Undertaken: Working on checkUnlockDoor; adding which key unlocks the door.

Date: January 24
Hours Spent: 1
Activities Undertaken: Cleaning up comments.

Date: January 24
Hours Spent: 3
Activities Undertaken: Refactoring check-action functions.

Date: January 25
Hours Spent: 6
Activities Undertaken: Refactored and organized GameClass.

Date: January 26
Hours Spent: 6
Activities Undertaken: Made changes and expanded locations.

Date: January 26
Hours Spent: 3
Activities Undertaken: Converting Input class to "control class" and moving help/quit into GameClass.

Date: January 28
Hours Spent: 8
Activities Undertaken: Cleaned up files and added comments.

Date: January 30
Hours Spent: 3
Activities Undertaken: Attempting to restructure end-game for simpler common actions.

Date: January 31
Hours Spent: 4
Activities Undertaken: Trying new structure for check functions.

Date: January 31
Hours Spent: 2
Activities Undertaken: Organizing GameClass.cpp and renamed to Game.cpp

Date: February 1
Hours Spent: 10
Activities Undertaken: Added test cases in main.

Date: February 1
Hours Spent: 5
Activities Undertaken: Finished header blocks for all source code.

Date: February 2
Hours Spent: 6
Activities Undertaken: Added header blocks for comments.

Date: February 2
Hours Spent: 2
Activities Undertaken: Added examine item function.

Date: February 3
Hours Spent: 4
Activities Undertaken: Added descriptions for all game items and matched enums/items/action files.

Date: February 4
Hours Spent: 2
Activities Undertaken: Amended item descriptions; updated action references.

Date: February 4
Hours Spent: 4
Activities Undertaken: Edited/ organized item descriptions (fixed alignment with enums).

Date: February 4
Hours Spent: 2
Activities Undertaken: Tweaked door checks and final item references.

Date: February 5
Hours Spent: 1
Activities Undertaken: Added “examine item” logic; refining checks.

Date: February 6
Hours Spent: 7
Activities Undertaken: Wrote and edited descriptions for characters

Date: February 6
Hours Spent: 6
Activities Undertaken: Wrote and edited descriptions for locations/ exits

Date: February 7
Hours Spent: 7
Activities Undertaken: Re-read/edited/ spellchecked all descriptions for all game objects.

Date: February 8
Hours Spent: 5
Activities Undertaken: Re-organized game class

Date: February 9
Hours Spent: 4
Activities Undertaken: Built out enterLocation functions

Date: February 11
Hours Spent: 10
Activities Undertaken: Finished up game play

Date: February 12
Hours Spent: 5
Activities Undertaken: Polished game/ made minor changes

Date: February 13
Hours Spent: 7
Activities Undertaken: Polished game/ made minor changes

Date: February 14
Hours Spent: 5
Activities Undertaken: Worked on balancing gameplay (Character fight stats, treasure score etc.)

Date: February 15
Hours Spent: 6
Activities Undertaken: Updated test cases

Date: February 15
Hours Spent: 3
Activities Undertaken: Tested game

Date: February 16
Hours Spent: 5
Activities Undertaken: Tested game