Design

Idea: You are on a broken spaceship and have to fix 3 problems in order to take off. Your air supply is limited (due to the ship being broken) starting off with 100% each time you move to another room your air supply decreases by 2%. You must go from room to room and examine each one for damage. If there is damage a description of the damage will be displayed as well as the required item to fix it and the location of that item. Go to that location search the room for items to pick up and then pick up that item. Go back to the room where you had your original problem and use that item to fix the problem. Once you use the item to fix the problem the item is removed from your inventory. Your inventory is only 2 spaces so you can't go around the ship picking up everything. If your inventory is full and you don't have the required item to fix the problem in a particular room you will be given the option to drop an item of your choice. If you drop the wrong item. You will have to go back to the room where you found that item and pick up another one. Player can have two of the same item in their inventory as there was no specification against this in the rubric.

Space class: This class acts as the base class for the 6 spaces in the game, Bridge, Cargo Bay, Common Area, Engine Bay, Living Area and Outside. Included functions are to get the name of the room the player is currently in as well as a description of the room. Description is variable based on a bool variable which depends on if the player has solved the problem in the room yet. So instead of "x is broken" player sees "x is fixed". The ability to examine the room and see if a problem exists in the room. A description of the problem. An item that is required to solve the problem. An item which is laying in the room which the player can add to their inventory. This will be the abstract class with a pure virtual destructor for the space class.

Bridge:
Damaged: Yes, Item to pick up: No
Derived from Space class
Derived from the space class. Class includes description and name of area as well as the ability to search the room for items and check the room for damage.

Common Area:(Required item in Engine Bay)
Derived from Space class
Damaged: Yes, Item to pick up: No
Functions: Name, Damage Description, Name of item required to fix damage, function that sets bool to true once damage is fixed, ability to search room for item.

Living Area:
Damaged: No, Item to pick up: Yes
Derived from Space class
Derived from the space class. Class includes description and name of area as well as the ability to search the room for items and check the room for damage.

Cargo Bay: (Required item to fix room in Living Area)
Damaged: Yes, Item to pick up: Yes
Derived from Space class
Functions: Name, Damage Description, Name of item required to fix damage, function that sets bool to true once damage is fixed, ability to search room for item.

Engine Bay:
(Required item to fix room in Cargo Bay)
Damaged: Yes, Item to pick up: Yes
Derived from Space class
Functions: Name, Damage Description, Name of item required to fix damage, function that sets bool to true once damage is fixed, ability to search room for item.

Outside:
Damaged: Yes, Item to pick up: No
Derived from Space class
Derived from the space class. Class includes description and name of area as well as the ability to search the room for items and check the room for damage.

Game Class:
The game is controlled from this class. Here the variables for the playes air supply (required to beat game in certain amount of time), players inventory, and bool variables for the 3 game problems which when all equal true indicate that the player has won the game. Here pointers are set up which allow player to move throughout the game and what they point to. Functions include a menu, player input, the ability to search a room, pick up an item and put it in your inventory solve a problem (fix a room) and show your inventory. A map is also displayed so the player has some idea where to go from where they are.

Validation:
Checks to ensure that player input is a whole number from the choice provided. Anything that is not a whole number from the specified range will result in another prompt for the player to make another entry.

Main
Creates instances of each class and then destroys them. All game play functions handled in game class.

TESTING

Test	Description	Expected Result
Play or quit	Decision tree based on if the user wants to play or quit	If play take to main menu if quit, quit program
Menu and player status	Menu displays with player status	Player status updates after movement. Includes current room player is in, inventory displayed, and remaining air.
Inventory add	Add item to inventory	Item is displayed in inventory and inventory size increases by 1.
Inventory full	Tell player and give choice to drop item	Ask player which item they would like to drop and then remove that item from inventory.
Room fix	Fix room correctly	If player has correct item in their inventory fix room and remove item from their inventory. If not tell them they don't.
Air supply	Air supply decrements	Air supply decrements as player changes rooms.
Die	Move 51 times	Player dies since oxygen reached 0.
win	Fix all 3 problems	Display a congratulatory message and exit game.

Reflection

The most difficult process of this game was the planning phase again. While I was able to effectively plan there seemed to be too many factors to account for. I finally looked at the rubric and tried to create a program exactly to those specifications. The specifications never said anything about the game being good or fun but rather what the game must do. With that in mind I simply jumped in creating the ability to track players location and move throughout a game space since that seemed to be the most fundamental requirement of the game. From there I went back to the rubric and more closely looked at it and implemented the correct requirements such as the 6 different spaces, making the Space class abstract, creating a player inventory etc. Once that was done I started to create more of a “game” so to speak. I did this since there just seemed to be an infinite amount of variables to take into account during the initial planning phase. Such as having an item removed from the players inventory once they fix a problem. I didn’t think of this until very late in the game and thankfully that was an easy fix. The rubric didn’t say anything about items being removed once they are used so I don’t believe I’d be graded down for something like that however I wanted to create a game that was at least semi realistic. There seems to be no end to the little details that I could put into the game to further refine it though. Which made this program difficult to complete in I wasn’t sure what I was being graded on whether it was an actual playable game or if I simply met all the requirements in the rubric. This question was answered by Noah who stated that it was simply the requirements however this wasn’t until I had finished the program.

Ultimately while the project was very difficult to start it was ultimately very fun to program and test. I particularly enjoyed coming up with a very rough concept for game play and then really refining it so that the user has a much easier time playing the game. However I used other people to do this. It doesn’t really help when you built the game and know where everything is. But by taking the feedback from friends and family such as adding the location, air supply, and current items in inventor to the top of the menu rather than a separate function really did help to make the game much more enjoyable. This current version of the game I think will serve as a great starting point for a major project which I would like to see how far I can take during my time off this winter.