CMSC 202 Spring 2020

Project 2 – Subnautica

Assignment: Project 2 – Subnautica

Due Date: Thursday, March 12th at 8:59pm

Value: 80 points

1. Overview

In this project, you will:

- Practice basic C++ syntax including branching structures
- Write classes and instantiate those classes using a constructor
- Use arrays to hold objects
- Use simple file input
- Practice breaking projects into multiple files
- Use a makefile to compile a project

2. Background

Subnautica is an open-world adventure game that was initially released Unknown Worlds Entertainment in 2014. The main character of the game (our Diver) starts after crash landing on a mostly ocean covered planet named 4546B. The player is wearing a simple scuba suit initially and wanders around the planet exploring areas and gathering resources to build new items. For this project, you will be required to create a text-based version of this game. You may see a graphical version of the game online here: https://www.youtube.com/watch?v=BtP8r8nRfko

In this implementation of Subnautica, you will be implementing the role of the Diver. For our purposes, the Diver can search for basic materials (such as lead or titanium), combine basic material into advanced material, and then combine advanced material into unique diving equipment. The goal is to build equipment to allow us to exceed the 1000-meter mark for our diving depth.

3. Assignment Description

Initially, you will need to read in a list of materials and recipes from a file and load them into an array of Materials. The list of materials is static, and you

can assume that the size can be stored in a constant (PROJ2_SIZE). The diver does not start with anything but can start searching for raw materials. Anytime a raw material is found, it is added to the diver's collection of items. Then when a specific recipe is met (two pieces of quartz becomes one piece of glass, for example), the diver can try and merge two pieces of raw material to make an advanced material. Then two items (raw or advanced) can be combined to build a unique item that increases the diver's depth. The list of all available combinations is available below in section 9 of this document.

4. Requirements:

This is a list of the requirements of this application. For this project, you will be provided with header files to start you in the right direction. For you to earn all the points, however, you will need to meet all the defined requirements.

- You must follow the coding standard as defined in the CMSC 202 coding standards (found on Blackboard under course materials). This includes comments as required.
- The project must be turned in on time by the deadline listed above.
- The project must be completed in C++. You may not use any libraries or data structures that we have not learned in class. Libraries we have learned include **`<iostream>`**, **`<fstream>`**, **`<iomanip>`**, ~~**`<vector>`**~~, **`<cmath>`**, **`<ctime>`**, **`<cstdlib>`**, and **`<string>`**. You may not use vectors – everything must be implemented in arrays. You should only use **`namespace std`**.
- Using the provided files, **`Material.h`**, **`Diver.h`**, **`Game.h`**, **`makefile`**, **`proj2_data.txt`** and **`proj2.cpp`**, create Subnautica. You can copy the files from my directory in **`/afs/umbc.edu/users/j/d/jdixon/pub/cs202/proj2`**.
- To copy them, navigate to your project 2 folder and type:

  ```
  cp /afs/umbc.edu/users/j/d/jdixon/pub/cs202/proj2/*.* .
  cp /afs/umbc.edu/users/j/d/jdixon/pub/cs202/proj2/makefile .
  ```

  Makefiles must be copied separately as they are not transferred using the *.* wildcards.
- You **must** use the provided header files (**`Material.h`**, **`Diver.h`** and **`Game.h`**). You may only add HELPER functions and global constants to these files. Do not add member variables to any class.

- All user input must be validated. For example, if a menu allows for 1, 2, or 3 to be entered and the user enters a 4, it will re-prompt the user. However, the user is expected to always enter the correct data type. i.e. If the user is asked to enter an integer, they will. If they are asked to enter a character, they will. You do not need to worry about checking for correct data types.
- There is a single input files for this project named, "**`proj2_data.txt`**". The file name can be stored as a constant. The file is already provided in Prof. Dixon's course folder on GL.
- For this project, pointers are not to be used, however, pass-by-reference is required.
- The player's name can have a space (use **`getline`**). Additionally, the names of the materials may have a space.
- There are three types of material: 1. Raw material that can be randomly found by searching. You can have unlimited of each raw material. 2. Advanced materials that can be created by merging two raw materials. You can have unlimited of each advanced material. 3. Unique materials that can be created by merging two advanced materials or an advanced material and a raw material. You can only have ONE of each type of unique material.
- Have a main menu that asks if the user wants to:
    - Display diver's materials and their current quantities in a numbered list.
    - Search for Raw Materials
        - Randomly finds one of the raw materials from the list of materials. The material must have the type of "raw" and should not have any required materials. Increases the quantity in the diver's inventory.
    - Merge Materials
        - Asks which materials you would like to combine. Checks a few things: 1. Validates that the numbers entered are inside the range (1-50). 2. Checks to see if there is a recipe with those two materials. 3. Checks to see if there is enough quantity to merge those two materials. 4. Checks to make

sure that unique items are only merged ONCE. All other items can be merged an unlimited number of times.

- A recipe for an advanced or unique material is made up of two materials in any order. For example, two quartz can be merged to make a glass. Two silver ore could be merged to make a wiring kit.
- **`RequestMaterial()`** inside of the Game class is simply the function that asks the user which material they would like to merge. It does something specific if the user enters a -1 (which should display all materials) and it validates to make sure they enter a valid choice. A diver would have to have a material in order to use to in a combination. For example, in order to try and make a Standard O2 Tank, the diver would have already have Silicone Rubber and Titanium.

- Display Score
  - A diver wins the game when they can dive to a depth of 1000 meters. Each unique item increases the maximum depth. Once they reach 1000 or greater, the game ends. You may have "duplicate" **similar** items, as in, you can have both Ultra Glide Fins and Swim Charge Fins even though you could only wear one at a time in real life. You may only have one each of the unique items though, as in, you may have one Standard O2 Tank.
- Exit

5. Recommendations

You must use the provided header files (**`Material.h, Diver.h,`** and **`Game.h`**) additionally, we provided you with the **`makefile`** and the **`proj2.cpp`**.

Here are some general implementation recommendations (do not follow these verbatim – these are GENERAL suggestions):

- Read each of the header files in detail. Use paper to take notes. Each header file has tons of comments.
- Design the solution (part is already designed, so make sure you understand how it works) on PAPER.

- Start with the **`Diver.cpp`** file and code everything. The Diver class doesn't need to have any **`cout`** statements. Don't forget the constructor! Test everything incrementally.
- Start **`Game.cpp`** – start building the constructor. Incrementally build it as you are testing a function.
- Once **`Diver.cpp`** is written, start on **`Game.cpp`** – start with loading in the materials from the **`proj2_data.txt`** file. This will be tricky because you have material names with spaces and a comma delimited data file. Check out the delimiter feature in getline here: https://www.geeksforgeeks.org/getline-string-c/
- **`CombineMaterials()is`** the most difficult function to write so expect to spend some time finishing it.
- After the files are successfully loaded (and displayed) work on the main menu.

6. Sample Input and Output

For this project, the input file is very simple. The only input file is called **`proj2_data.txt`**. You can code the file name as a constant.

The columns for **`proj2_data.txt`** are as follows: material name, material type (raw, advanced, or unique), first material required to make, second material required to make, and depth boost.

Raw materials have no first and second materials to make and provide no depth boost. They can only be found by searching.

Advanced materials are an intermediate material that are made by combining two raw materials. They provide no depth boost.

Unique materials are a final material that are created by combining two materials. These unique materials can have a maximum quantity of one. Some unique materials are made up of other unique materials.

For example, Copper Ore is raw and can be found by searching (option 2 in the main menu). Copper Wire is Advanced and can be made by merging copper ore with copper ore. Finally, a radiation suit can be made by merging fiber mesh with lead. Once the radiation suit has been made, the current depth would increase by 100.

```
Copper Ore,raw,0,0,0
```

```
Diamond,raw,0,0,0
Gold,raw,0,0,0
Copper Wire,advanced,Copper Ore,Copper Ore,0
Wiring Kit,advanced,Silver Ore,Silver Ore,0
Enameled Glass,advanced,Stalker Teeth,Glass,0
Radiation Suit,unique,Fiber Mesh,Lead,100
Rebreather,unique,Wiring Kit,Fiber Mesh,100
Reinforced Dive Suit,unique,Synthetic Fibers,Diamond,100
```

The file can be downloaded from Prof. Dixon's data folder by navigating to your project 2 folder and typing the following command:

`cp /afs/umbc.edu/users/j/d/jdixon/pub/cs202/proj2/proj2_* .`

After you copy the data file, you can type "**`cat proj2_data.txt`**" and it should show you the entire data file.

In the sample output below, user input is colored blue for clarity. After compiling and running proj2, the output would look like this:

```
[jdixon@linux2 proj2]$ make run
./proj2
**Subnautica Art from GameTitle**
50 materials loaded.
What is the name of your Diver?
JD

What would you like to do?
1. Display your Diver's Materials
2. Search for Raw Materials
3. Attempt to Merge Materials
4. See Score
5. Quit
```

If you were to display Diver's Materials, it would look like this:

```
What would you like to do?
1. Display your Diver's Materials
2. Search for Raw Materials
3. Attempt to Merge Materials
4. See Score
5. Quit
1
1. Cave Sulfur 0
2. Copper Ore 0
3. Diamond 0
4. Gold 0
5. Lead 0
… **5-47 omitted for space**
48. Ultra High Capacity Tank 0
49. Beacon 0
50. Seaglide 0
What would you like to do?
1. Display your Diver's Materials
2. Search for Raw Materials
3. Attempt to Merge Materials
4. See Score
5. Quit
```

If the user would choose 2, Search for Raw Materials, then the output would look like this:

```
What would you like to do?
1. Display your Diver's Materials
2. Search for Raw Materials
3. Attempt to Merge Materials
4. See Score
```

```
5. Quit
2
Diamond Found!
What would you like to do?
1. Display your Diver's Materials
2. Search for Raw Materials
3. Attempt to Merge Materials
4. See Score
5. Quit
2
Acid Mushroom Found!
What would you like to do?
1. Display your Diver's Materials
2. Search for Raw Materials
3. Attempt to Merge Materials
4. See Score
5. Quit
2
Table Coral Sample Found!
What would you like to do?
1. Display your Diver's Materials
2. Search for Raw Materials
3. Attempt to Merge Materials
4. See Score
5. Quit
2
Silver Ore Found!
What would you like to do?
1. Display your Diver's Materials
2. Search for Raw Materials
3. Attempt to Merge Materials
4. See Score
5. Quit
```

If the user would choose 2, Attempt To Combine Materials, then choose something that we have in inventory:

```
Which materials would you like to merge?
To list known materials enter -1
9
Which materials would you like to merge?
To list known materials enter -1
9
Silver Ore combined with Silver Ore to make Wiring Kit!
Your diver has built Wiring Kit.
What would you like to do?
1. Display your Diver's Materials
2. Search for Raw Materials
3. Attempt to Merge Materials
4. See Score
5. Quit
```

Here is an example where after we used our 2 silver ore to build a wiring kit, if we go back and try to build another wiring kit, we don't have any silver ore left so the game indicates this.

```
What would you like to do?
1. Display your Diver's Materials
2. Search for Raw Materials
3. Attempt to Merge Materials
4. See Score
5. Quit
3
Which materials would you like to merge?
To list known materials enter -1
9
Which materials would you like to merge?
To list known materials enter -1
9
You do not have enough Silver Ore or Silver Ore to attempt that merge
What would you like to do?
1. Display your Diver's Materials
2. Search for Raw Materials
3. Attempt to Merge Materials
```

```
4. See Score
5. Quit
```

Now when we go back to the diver's materials (their inventory), we are showing a wiring kit but no silver (which we used to make the wiring kit).

```
What would you like to do?
1. Display your Diver's Materials
2. Search for Raw Materials
3. Attempt to Merge Materials
4. See Score
5. Quit
1
1. Cave Sulfur 1
2. Copper Ore 0
3. Diamond 2
4. Gold 0
5. Lead 0
6. Lithium 0
7. Quartz 0
8. Ruby 1
9. Silver Ore 0
10. Stalker Teeth 1
11. Titanium 0
12. Table Coral Sample 1
13. Acid Mushroom 1
**14-21 Removed for Space**
22. Battery 0
23. Computer Chip 0
24. Copper Wire 0
25. Wiring Kit 1
**26-49 Removed for Space**
50. Seaglide 0
What would you like to do?
1. Display your Diver's Materials
2. Search for Raw Materials
3. Attempt to Merge Materials
```

```
4. See Score
5. Quit
```

Here is an example where of the See Score is selected. As the unique items that boost the depth are acquired, the current depth should increase. Once the current depth meets or exceeds the maximum depth, the game ends and the player wins.

```
What would you like to do?
1. Display your Diver's Materials
2. Search for Raw Materials
3. Attempt to Merge Materials
4. See Score
5. Quit
4
***The Diver***
The Great Diver JD
Maximum Depth: 1000
Current Depth: 0
What would you like to do?
1. Display your Diver's Materials
2. Search for Raw Materials
3. Attempt to Merge Materials
4. See Score
5. Quit
```

Here are some example runs where additional input validation is being shown and where the name of the diver has a space:

```
50 materials loaded.
What is the name of your Diver?
Ellen DeGeneres

What would you like to do?
1. Display your Diver's Materials
2. Search for Raw Materials
3. Attempt to Merge Materials
4. See Score
5. Quit
```

```
6
What would you like to do?
1. Display your Diver's Materials
2. Search for Raw Materials
3. Attempt to Merge Materials
4. See Score
5. Quit
0
What would you like to do?
1. Display your Diver's Materials
2. Search for Raw Materials
3. Attempt to Merge Materials
4. See Score
5. Quit
3
Which materials would you like to merge?
To list known materials enter -1
0
Which materials would you like to merge?
To list known materials enter -1
100
Which materials would you like to merge?
To list known materials enter -1
-1
1. Cave Sulfur 0
2. Copper Ore 0
3. Diamond 0
**3-50 removed for space**
```

There is a longer run of the game available as **`proj2_sample1.txt`** with the other files.

7. Compiling and Running

We have provided you with a sample **`makefile`** which should help you compile all of the classes and the program itself.

Once you have compiled using the provided **makefile**, enter the command **make run** or **./proj2** to run your program. If your executable is not proj2, you will lose points. It should look like the sample output provided above.

8. Completing your Project

When you have completed your project, you can copy it into the submission folder. You can copy your files into the submission folder as many times as you like (before the due date). We will only grade what is in your submission folder.

For this project, you should submit the following files to the **proj2** subdirectory:

Diver.cpp, Diver.h

Game.cpp, Game.h

Material.h, proj2.cpp

For this project, you should only modify the header files to add helper functions but do not add any additional variables (such as **Diver.h or Game.h)**. Almost all edits should be in the classes **.cpp** files (**Diver.cpp** and **Game.cpp**).

You do not need to submit the **makefile**.

As you should have already set up your symbolic link for this class, you can just copy your files listed above to the submission folder.

a. cd to your project 2 folder. An example might be cd
   ~/202/projects/proj2

b. **cp Material.h Diver.h Diver.cpp Game.h Game.cpp proj2.cpp ~/cs202proj/proj2**

You can check to make sure that your files were successfully copied over to the submission directory by entering the command

ls ~/cs202proj/proj2

Make sure that the required files are submitted by the deadline. If the copy command provided does not work, it is your responsibility to figure out what is wrong and that all required files have been submitted.

You can check that your program compiles and runs in the **proj2** directory, but please clean up any **.o** and executable files. Again, do not develop your

code in this directory and you should not have the only copy of your program here. Uploading of any **`.gch`** files will result in a severe penalty.

IMPORTANT: If you want to submit the project late (after the due date), you will need to copy your files to the appropriate late folder. If you can no longer copy the files into the proj2 folder, it is because the due date has passed. You should be able to see your proj2 files but you can no longer edit or copy the files in to your proj2 folder. (They will be read only)

- If it is 0-24 hours late, copy your files to **`~/cs202proj/proj2-late1`**
- If it is 24-48 hours late, copy your files to **`~/cs202proj/proj2-late2`**
- If it is after 48 hours late, it is too late to be submitted.

9. List of Possible Materials

Below is a list of all possible materials that can be created using merge.

Name	Type	Requirement 1	Requirement 2	Depth
Cave Sulfur	raw	0	0	0
Copper Ore	raw	0	0	0
Diamond	raw	0	0	0
Gold	raw	0	0	0
Lead	raw	0	0	0
Lithium	raw	0	0	0
Quartz	raw	0	0	0
Ruby	raw	0	0	0
Silver Ore	raw	0	0	0
Stalker Teeth	raw	0	0	0
Titanium	raw	0	0	0
Table Coral Sample	raw	0	0	0
Acid Mushroom	raw	0	0	0
Gel Sack	raw	0	0	0
Creepvine Seed Cluster	raw	0	0	0
Creepvine Sample	raw	0	0	0
Blood Oil	raw	0	0	0
Aerogel	advanced	Gel Sack	Ruby	0
Benzene	advanced	Blood Oil	Blood Oil	0
Synthetic Fibers	advanced	Benzene	Fiber Mesh	0
Advanced Wiring Kit	advanced	Computer Chip	Copper Wire	0
Battery	advanced	Copper Wire	Acid Mushroom	0
Computer Chip	advanced	Table Coral Sample	Wiring Kit	0
Copper Wire	advanced	Copper Ore	Copper Ore	0
Wiring Kit	advanced	Silver Ore	Silver Ore	0
Enameled Glass	advanced	Stalker Teeth	Glass	0
Fiber Mesh	advanced	Creepvine Sample	Creepvine Sample	0
Glass	advanced	Quartz	Quartz	0
Lubricant	advanced	Creepvine Seed Cluster	0	0
Plasteel Ingot	advanced	Titanium Ingot	Lithium	0
Silicone Rubber	advanced	Creepvine Seed Cluster	Creepvine Seed Cluster	0
Titanium Ingot	advanced	Titanium	Titanium	0
Light Stick	unique	Battery	Glass	25
Pathfinder Tool	unique	Creepvine Seed Cluster	Copper Wire	50
Repair Tool	unique	Cave Sulfur	Silicone Rubber	25
Compass	unique	Wiring Kit	Copper Wire	50
Fins	unique	Silicone Rubber	Silicone Rubber	100
Floating Air Pump	unique	Fiber Mesh	Titanium	50
High Capacity O2 Tank	unique	Standard O? Tank	Lubricant	200
Lightweight High Capacity Tank	unique	High Capacity O2 Tank	Titanium Ingot	100
Radiation Suit	unique	Fiber Mesh	Lead	100
Rebreather	unique	Wiring Kit	Fiber Mesh	100
Reinforced Dive Suit	unique	Synthetic Fibers	Diamond	100
Standard O? Tank	unique	Silicone Rubber	Titanium	100
Stillsuit	unique	Synthetic Fibers	Aerogel	100
Swim Charge Fins	unique	Advanced Wiring Kit	Ultra Glide Fins	100
Ultra Glide Fins	unique	Fins	Synthetic Fibers	200
Ultra High Capacity Tank	unique	Lightweight High Capacity Tank	Plasteel Ingot	200
Beacon	unique	Copper Ore	Gold	50
Seaglide	unique	Advanced Wiring Kit	Battery	200