CMSC 202 Fall 2019

Project 4 – UMBC Adventure

**Assignment:** Project 4 – UMBC Adventure

**Due Date:** Tuesday, November 12th at 8:59pm

**Value:** 80 points

# Overview

In this project, you will:

* Practice creating classes with inheritance
* Working with pointers as they pertain to classes
* Using polymorphism
* Working with a large number of classes

# Background

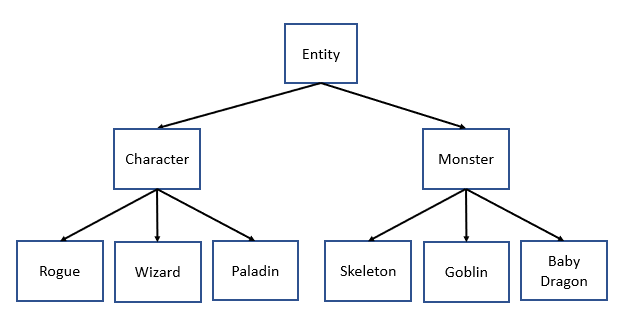
A text game or text-based game is a video game that uses text characters instead of the graphics of modern-based games. The earliest examples of text-based games were introduced in the mid-1970s in games such as Colossal Cave Adventure. They remained popular throughout the 1980s especially as the popularity of Dungeons and Dragons expanded.

Historically, text-based games were popular because they were typically easier to write and required less processing power than their graphics-based counterparts. Even in 2019, people continue playing MUDs (multi-user dungeon) and exploring interactive fiction using text-based interfaces. Many beginning programmers still create these types of games to familiarize themselves with a programming language which is where our project begins.

# Assignment Description

Your assignment is to develop a simple adventure game where a character can adventure through a dungeon and interact with a variety of monsters. Data files for a variety of dungeons will be provided in a standard format.

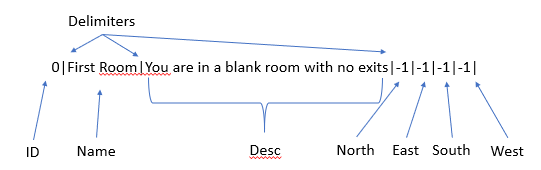
As this project is focused on inheritance and polymorphism, there is an inheritance chain as defined in the figure below.



In addition to these classes, there is a room class that defines the characteristics for each room and a game class the manages the game itself. Specific information about the amount of damage that each character or monster can do is included in the table below:



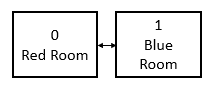
The provided map files are formatted like this:



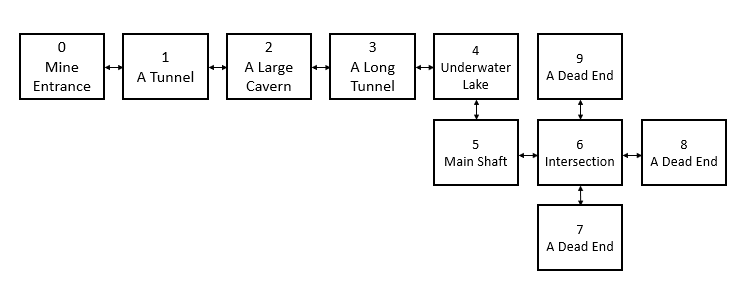
If we look at the first row in the map 1 file, we are saying the following:

1. Column 1 – Room ID (should be an integer)
2. Column 2 – Room Name (string)
3. Column 3 – Room Desc (string)
4. Column 4 – Room ID of room to the north (-1 is no room)
5. Column 5 – Room ID of room to the east (-1 is no room)
6. Column 6 – Room ID of room to the south (-1 is no room)
7. Column 7 – Room ID of room to the west (-1 is no room)

If you are having a tough time visualizing the rooms structures, here is an example of map1\_data.txt in an image:



And here is the larger map from map2\_data.txt:



# Requirements:

This is a list of the requirements of this application. For this project, it is up to you exactly how you want to implement it. 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 standard (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>, <cmath>, <cstdlib>, <vector>, <ctime.h>,** and **<string>**. You should only use **namespace std**.
* You may use any of the conversion functions such as **stoi** or **atoi**.
* Using the provided files, **Room.h**, **Character.h**, **Monster.h**, **Paladin.h, Rogue.h, Wizard.h, BabyDragon.h, Goblin.h, Skeleton.h** and **proj4.cpp**, create an adventure game. You can copy the files from my directory in **/afs/umbc.edu/users/j/d/jdixon/pub/cs202/proj4**. To copy them, navigate to your project 2 folder and type:

**cp /afs/umbc.edu/users/j/d/jdixon/pub/cs202/proj4/\* .**

* You may **NOT** modify the headers files.
* There are ten required classes in this project: Game, Room, Character, Monster, Paladin, Rogue, Wizard, BabyDragon, Goblin, and Skeleton. All functions listed in their class files must be implemented completely (even if you do not use them).
* All entities (and their subclasses) and rooms must be dynamically allocated (and deallocated).
* 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.
* The name of the map file may change. We have two map files prepared for you to test your program. The name of the files are: **map1\_data.txt and** **map2\_data.txt.**

The map files get increasingly complex (2 rooms then a bunch!)

* You will be using a vector to store the map itself. Each room must be dynamically allocated (and deallocated!).

## Class Requirements

* Room – Use **Room.h** to write the **Room.cpp** file for this class. This class contains the code to manage the rooms and the movement between the rooms. All rooms must be dynamically allocated (using new). Movement must be validated to check to see if an exit exists in that direction!
* Entity – Use **Entity.h** to write the **Entity.cpp** file for this class. Entities are the base class for both characters (who are the players) and the monsters. They only have three characteristics: name, health (or hp), and level.
* Character – Use **Character.h** to write the **Character.cpp** file for this class. Characters are the “intermediate” class for the player. There are three subclasses to define named Paladin, Rogue, and Wizard. As the character does not have any purely virtual functions, you can still create a character. The only thing noteworthy between any of the subclasses is the Attack and SpecialAttack functions.
* Monster – Use **Monster.h** to write the **Monster.cpp** file for this class. Monsters are similar to characters. They have a name, health(hp), and level. As there is a purely virtual function in Monster, you cannot build a generic Monster. There are three child classes named BabyDragon, Goblin, and Skeleton with specialized attacks.
* Game – Use **Game.h** to write the **Game.cpp** file for this class. This is the longest and most complicated of the classes as it manages the entire game. After you have Room coded, you can try to **LoadMap()**. Both the character and the monsters must be dynamically allocated (using the **new** keyword). **CharacterCreation** isn’t too complex as it just creates a character of a specific type (Paladin, Rogue, or Wizard). **Move** is challenging because it must validate to make sure that the room being moved to exists.
* Movement and Monsters – when the games starts (always in room 0), there may or may not be a monster there. When a player moves from one room to another, there is a 25% chance one of the following occurs (a Baby Dragon is there, a Goblin is there, a Skeleton is there, or nothing is there, and everything is peaceful). If you leave a room and come back later, a different monster may be there. You cannot rest if a monster exists in the room. Rests and special attacks are finite based on the constants in **Game.h**.

# Sample Input and Output

For this project, input files are not too challenging. For this project, the data files can change based on the map you are going to adventure in. The standard format of the filename will be **map1\_data.txt**. Included in the main function is the command line arguments to use each of these files. When you load the project, you will be loading the file at the same time. For example, if I wanted to load the **map1\_data.txt** file, I could use **./proj4 map1\_data.txt**. Additionally, the **makefile** has all three already coded. You can use **make run1 and make run2** for the normal runs and **make val1 and make val2** for the valgrind runs.

For example, **map1\_data.txt** would look like this because it only has two room. Larger examples (map2 for example) has many more lines.

|  |
| --- |
| - 0|Red Room|You are in a mysterious dark red room. All of the walls are faintly glowing but you can't see anything of interest. There is an archway in the eastern wall.|-1|1|-1|-1|  1|Blue Room|The walls here glow a deep blue. The airs smells like burning paper. There is an archway in the west wall.|-1|-1|-1|0| |

The file can be downloaded from Prof. Dixon’s data folder by navigating to your project 1 folder and typing the following command:

**cp /afs/umbc.edu/users/j/d/jdixon/pub/cs202/proj4/\* .**

After you copy the data file, you can type “**cat map1\_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 proj4, the output would look like this:

|  |
| --- |
| [jdixon@linux6 proj4]$ make run1  ./proj4 map1\_data.txt  Loading file: map1\_data.txt  Welcome to UMBC Adventure!  Character Name: JD  Select a class  1. Rogue  2. Wizard  3. Paladin  4. No Class  1  Red Room  You are in a mysterious dark red room. All of the walls are faintly glowing  but you can't see anything of interest. There is an archway in the eastern wall.  Possible Exits: E  It is peaceful here.  What would you like to do?  1. Look  2. Move  3. Attack Monster  4. Rest  5. Check Stats  6. Quit  6  Good bye! |

Let’s look at map1 again which has two rooms. Again, any time the character goes into another room, there is a 25% chance for one of the three monsters (dragon, goblin, or skeleton) or 25% for nothing (peaceful).

|  |
| --- |
| ./proj4 map1\_data.txt  Loading file: map1\_data.txt  Welcome to UMBC Adventure!  Character Name: JD  Select a class  1. Rogue  2. Wizard  3. Paladin  4. No Class  3  Red Room  You are in a mysterious dark red room. All of the walls are faintly glowing  but you can't see anything of interest. There is an archway in the eastern wall.  Possible Exits: E  It is peaceful here.  What would you like to do?  1. Look  2. Move  3. Attack Monster  4. Rest  5. Check Stats  6. Quit  2  Which direction? (N E S W)  n  Which direction? (N E S W)  w  Which direction? (N E S W)  s  Which direction? (N E S W)  e  Blue Room  The walls here glow a deep blue. The airs smells like burning paper.  There is an archway in the west wall.  Possible Exits: W  A skeleton lumbers around the room.  What would you like to do?  1. Look  2. Move  3. Attack Monster  4. Rest  5. Check Stats  6. Quit  3  1. Normal Attack  2. Special Attack  1  JD attacks dealing 1 damage.  Skeleton deals 1 point of damage!  JD's health:14  Skeleton's health:6  1. Normal Attack  2. Special Attack  2  JD uses smite evil!  JD deals 7 damage!  Skeleton deals 1 point of damage!  You have defeated the Skeleton.  What would you like to do?  1. Look  2. Move  3. Attack Monster  4. Rest  5. Check Stats  6. Quit  5  Name: JD  HP: 13  Rests: 1  Special: 2  What would you like to do?  1. Look  2. Move  3. Attack Monster  4. Rest  5. Check Stats  6. Quit |

Here we have an example of make run2. It is a much larger map (it is a mine).

|  |
| --- |
| Loading file: map2\_data.txt  Welcome to UMBC Adventure!  Character Name: JD  Select a class  1. Rogue  2. Wizard  3. Paladin  4. No Class  3  Mine Entrance  Here is where the actual mining began, the main shaft extends east  to where it splits into many smaller shafts following veins of minerals or  valueable crystals. Cart tracks lead down into rubble, the first evidence  of old cave-ins.  Possible Exits: E  It is peaceful here.  What would you like to do?  1. Look  2. Move  3. Attack Monster  4. Rest  5. Check Stats  6. Quit  2  Which direction? (N E S W)  e  A Tunnel  The tunnel of stone is smooth and straight heading deeper into the earth.  Shelves project outward at angles, with water trickling out of some of them  adding to the dampness in the air.  Possible Exits: EW  A goblin is here picking his nose.  What would you like to do?  1. Look  2. Move  3. Attack Monster  4. Rest  5. Check Stats  6. Quit  3  1. Normal Attack  2. Special Attack  1  JD attacks dealing 4 damage.  Goblin deals 1 point of damage!  JD's health:14  Goblin's health:1  1. Normal Attack  2. Special Attack  2  JD uses smite evil!  JD deals 3 damage!  Goblin deals 1 point of damage!  You have defeated the Goblin.  What would you like to do?  1. Look  2. Move  3. Attack Monster  4. Rest  5. Check Stats  6. Quit |

Here is an example using make val1 (with valgrind):

|  |
| --- |
| valgrind ./proj4 map1\_data.txt  ==20073== Memcheck, a memory error detector  ==20073== Copyright (C) 2002-2017, and GNU GPL'd, by Julian Seward et al.  ==20073== Using Valgrind-3.15.0 and LibVEX; rerun with -h for copyright info  ==20073== Command: ./proj4 map1\_data.txt  ==20073==  Loading file: map1\_data.txt  Welcome to UMBC Adventure!  Character Name: JD  Select a class  1. Rogue  2. Wizard  3. Paladin  4. No Class  1  Red Room  You are in a mysterious dark red room. All of the walls are faintly glowing  but you can't see anything of interest. There is an archway in the eastern wall.  Possible Exits: E  A goblin is here picking his nose.  What would you like to do?  1. Look  2. Move  3. Attack Monster  4. Rest  5. Check Stats  6. Quit  3  1. Normal Attack  2. Special Attack  1  JD attacks dealing 6 damage.  Goblin deals 1 point of damage!  You have defeated the Goblin.  What would you like to do?  1. Look  2. Move  3. Attack Monster  4. Rest  5. Check Stats  6. Quit  2  Which direction? (N E S W)  E  Blue Room  The walls here glow a deep blue. The airs smells like burning paper.  There is an archway in the west wall.  Possible Exits: W  A goblin is here picking his nose.  What would you like to do?  1. Look  2. Move  3. Attack Monster  4. Rest  5. Check Stats  6. Quit  3  1. Normal Attack  2. Special Attack  2  JD performs a sneak attack!  JD deals 11 damage.  Goblin deals 1 point of damage!  You have defeated the Goblin.  What would you like to do?  1. Look  2. Move  3. Attack Monster  4. Rest  5. Check Stats  6. Quit  6  Good bye!  ==20073==  ==20073== HEAP SUMMARY:  ==20073== in use at exit: 0 bytes in 0 blocks  ==20073== total heap usage: 21 allocs, 21 frees, 84,873 bytes allocated  ==20073==  ==20073== All heap blocks were freed -- no leaks are possible  ==20073==  ==20073== For lists of detected and suppressed errors, rerun with: -s  ==20073== ERROR SUMMARY: 0 errors from 0 contexts (suppressed: 0 from 0) |

Here is an example of validating the direction (checkDirection):

|  |
| --- |
| [jdixon@linux6 proj4]$ make run1  ./proj4 map1\_data.txt  Loading file: map1\_data.txt  Welcome to UMBC Adventure!  Character Name: JD  Select a class  1. Rogue  2. Wizard  3. Paladin  4. No Class  2  Red Room  You are in a mysterious dark red room. All of the walls are faintly glowing  but you can't see anything of interest. There is an archway in the eastern wall.  Possible Exits: E  A skeleton lumbers around the room.  What would you like to do?  1. Look  2. Move  3. Attack Monster  4. Rest  5. Check Stats  6. Quit  2  Which direction? (N E S W)  n  Which direction? (N E S W)  w  Which direction? (N E S W)  s  Which direction? (N E S W)  N  Which direction? (N E S W)  W  Which direction? (N E S W)  S  Which direction? (N E S W)  E  Blue Room  The walls here glow a deep blue. The airs smells like burning paper.  There is an archway in the west wall.  Possible Exits: W  A skeleton lumbers around the room.  What would you like to do?  1. Look  2. Move  3. Attack Monster  4. Rest  5. Check Stats  6. Quit  3  1. Normal Attack  2. Special Attack  1  JD attacks dealing 6 damage.  Skeleton deals 1 point of damage!  JD's health:9  Skeleton's health:1  1. Normal Attack  2. Special Attack  2  JD casts magic missle!  JD deals 6 damage!  Skeleton deals 1 point of damage!  You have defeated the Skeleton.  What would you like to do?  1. Look  2. Move  3. Attack Monster  4. Rest  5. Check Stats  6. Quit |

# Compiling and Running

You need to write a **makefile** for this project as one will not be provided for you in order to compile and test. Make sure to create any macros to help!

Once you have compiled using your **makefile**, enter the command **make run** or **./proj4** to run your program. Make sure you have implemented that as part of your make file. If your executable is not proj4, you will lose points. It should look similar to the sample output provided above. Some of the simulation is based on randomness so the output will vary.

Because we are using a significant amount of dynamic memory for this project, you are required to manage any memory leaks that might be created. Remember, in general, for each item that is dynamically created, it should be deleted using a destructor.

One way to test to make sure that you have successfully removed any of the memory leaks is to use the **valgrind** command.

Since this project makes extensive use of dynamic memory, it is important that you test your program for memory leaks using **valgrind**:

**valgrind ./proj4 map1\_data.txt**

Note: If you accidently use **valgrind make run1**, you may end up with some memory that is still reachable. Do not test this – test using the command above where you include the input file. The makefile should include **make val1** (which is ok).

If you have no memory leaks, you should see output like the following:

|  |
| --- |
| ==5606==  ==5606== HEAP SUMMARY:  ==5606== in use at exit: 0 bytes in 0 blocks  ==5606== total heap usage: 87 allocs, 87 frees, 10,684 bytes allocated  ==5606==  ==5606== All heap blocks were freed -- no leaks are possible  ==5606==  ==5606== For counts of detected and suppressed errors, rerun with: -v  ==5606== ERROR SUMMARY: 0 errors from 0 contexts (suppressed: 6 from 6) |

The important part is “in use at exit: 0 bytes 0 blocks,” which tells me all the dynamic memory was deleted before the program exited. If you see anything other than "0 bytes 0 blocks" there is probably an error in one of your destructors. We will evaluate this as part of the grading for this project.

Additional information on **valgrind** can be found here: <http://valgrind.org/docs/manual/quick-start.html>

Once you have compiled using the provided **makefile**, enter the commands (**make run1**, **make run2**, **make val1**, or **make val2**) or **./proj4 map1\_data.txt** to run your program. If your executable is not proj4, you will lose points. It should look like the sample output provided above.

# 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 **all** (except your makefile) files to the **proj4** subdirectory. 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.

* 1. cd to your project 2 folder. An example might be cd **~/202/projects/proj4**
  2. **cp \* ~/cs202proj/proj4**

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

ls ~/cs202proj/proj4

You can check that your program compiles and runs in the **proj4** 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.

For additional information about project submissions, there is a more complete document available in Blackboard under “Course Materials” and “Project Submission.”

**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 proj4 folder, it is because the due date has passed. You should be able to see your proj4 files but you can no longer edit or copy the files in to your proj4 folder. (They will be read only)

* If it is 0-24 hours late, copy your files to **~/cs202proj/proj4-late1**
* If it is 24-48 hours late, copy your files to **~/cs202proj/proj4-late2**
* If it is after 48 hours late, it is too late to be submitted.