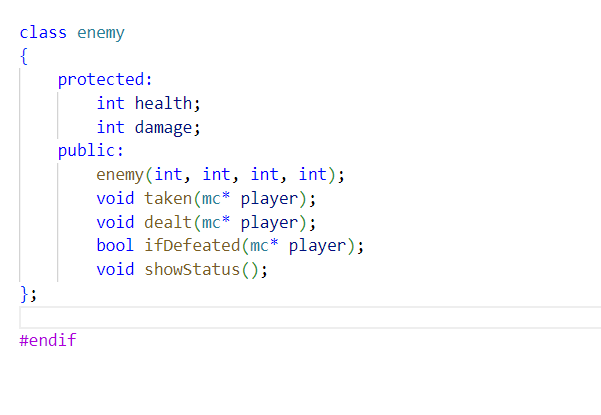
Project Documentation - EC327

Kenneth Byun, Jeff Guan, Benny Li, Carson Mooc

Initially, we were planning to use GameMaker studio as the main environment for developing the game, however after considering this decision we came to the conclusion that using a standard C++ environment would be more suited to the type of game we were trying to create and allowed us to be more efficient with the programming aspect part of the project. This would turn out to be better than trying to learn GML (GameMaker Language) as even though it has similarities to C++, there are some quirks about the language that were avoided by just using C++.

A challenge we encountered during development was the inclusion of visuals in the game. This posed an obstacle for us because of its implementation through GameMaker studio, which we decided not to use and go through the process of learning the ins and outs of GML, however upon further evaluation we decided that the last few weeks would have been better spent refining the games mechanics and functionality.

One of the core elements in the game is the “enemy” class that serves as the parent class for the various types of enemies. These consist of the subclasses “rogue knight”, “ghost”, “minotaur”, and “skull”, all of which are derived from the enemy class (in terms of members and functions) with altercations in the values for each respective “health” and “damage” value to provide the user with more gameplay variety.



Arguably the most important part of the game, the “mc” class is a collection of functions that allow other classes to change the values of private members that are the central focus of the game’s core mechanics. These include things such as user health and gold, all of which the mc class contains to provide gameplay functionality within the program. Without this class, the game would not function at all.

Another aspect of the back-end components included the error checking within the game to ensure that user input would not cause the game to function incorrectly. Using while loops allowed for repeated prompts until the input is valid and progression within the game is halted until then. There is also an error check for negative numbers within the program which takes it into account for changing some of the private members. One of the bugs that was faced was when a negative number was the cause of the program displaying the wrong type of enemy class, which needed to be addressed by the error checking of negative numbers.