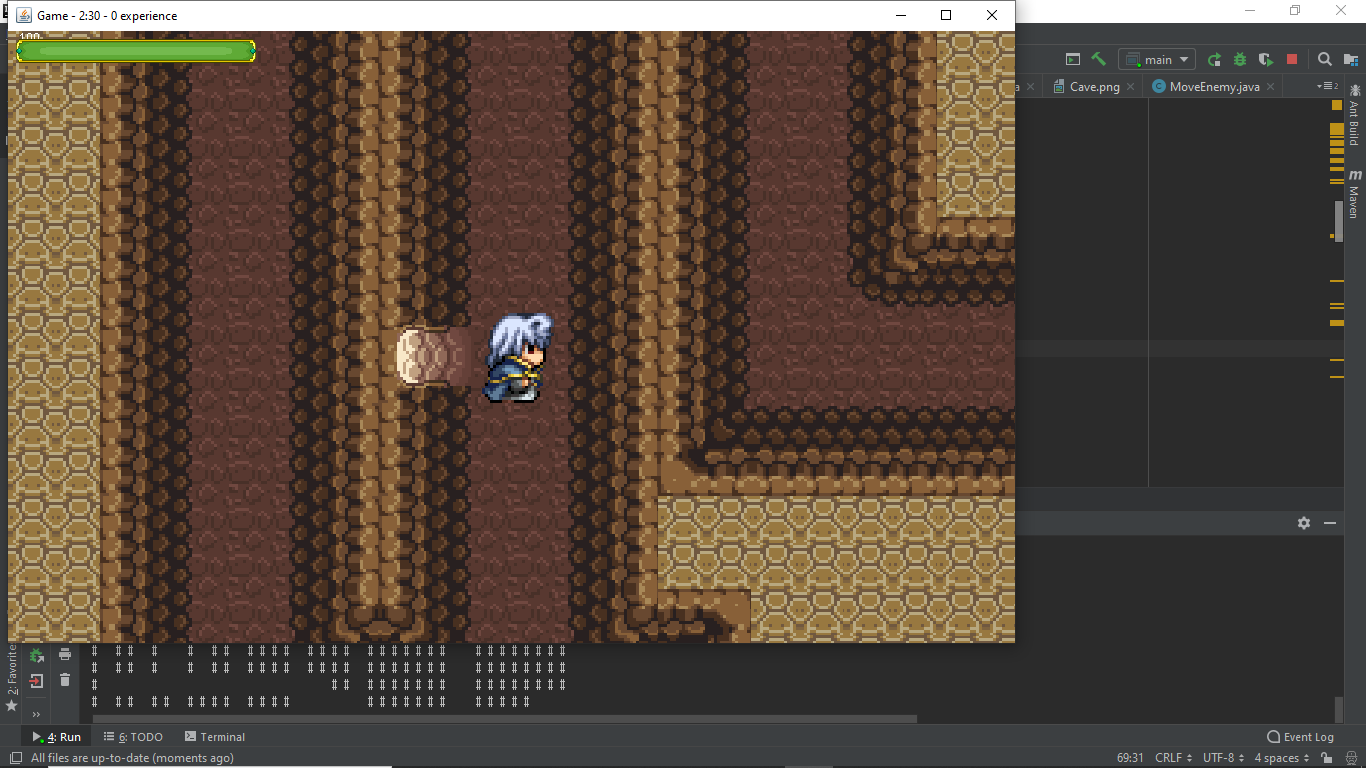
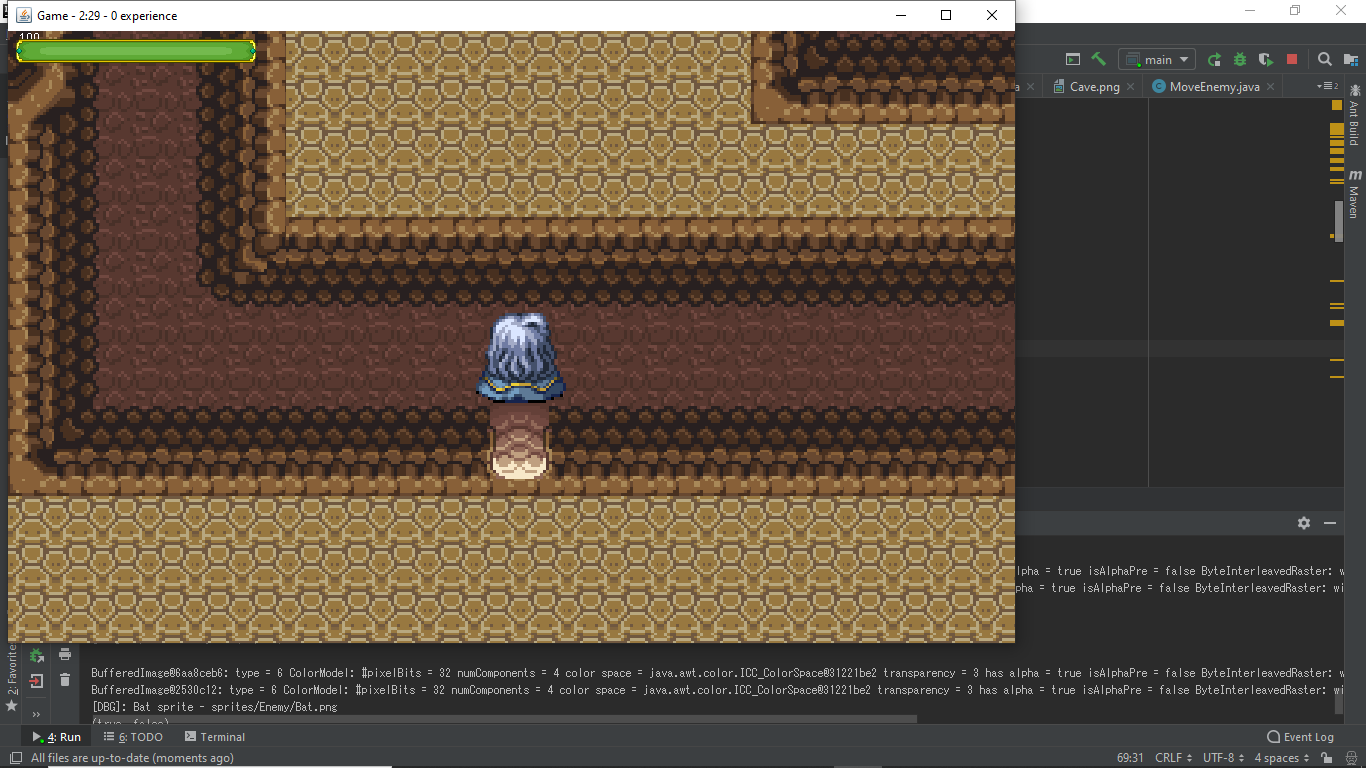
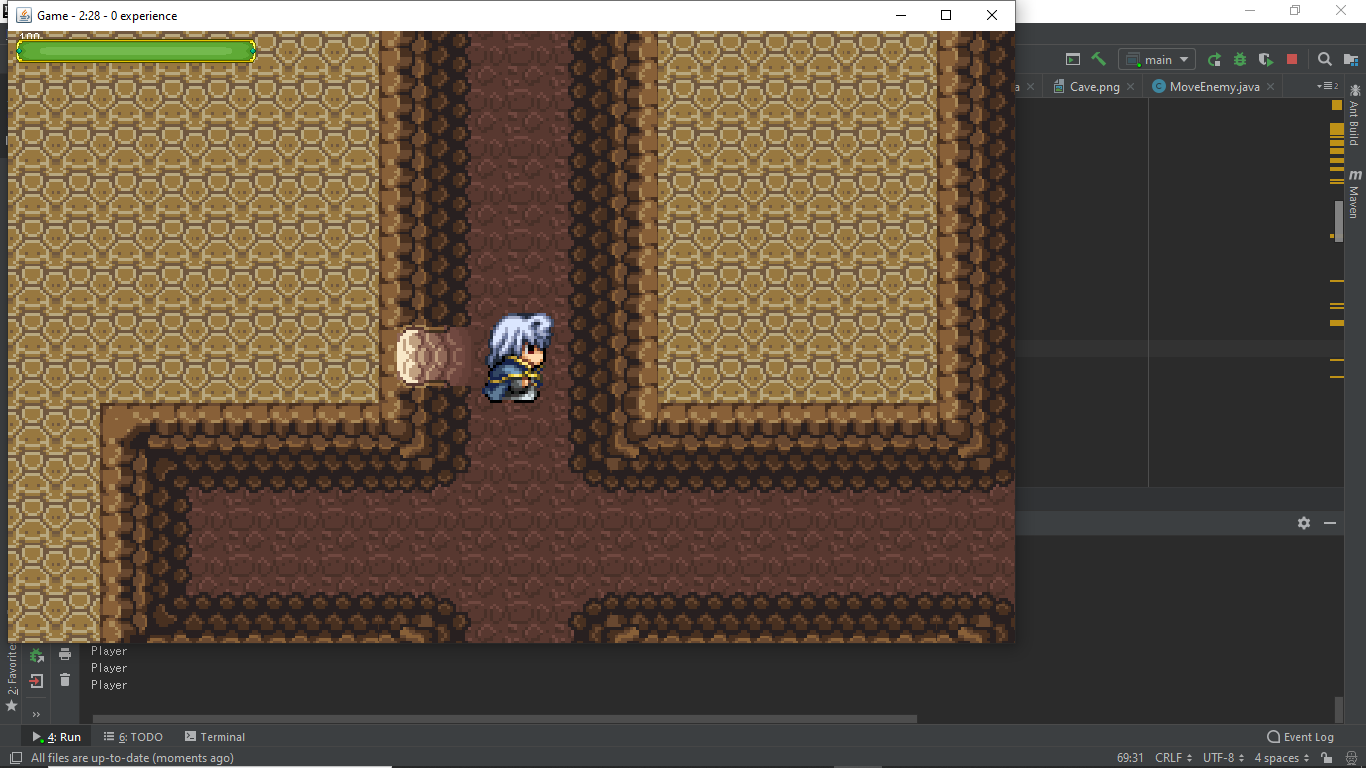
This is my game. It is a roguelike randomly-generated dungeoncrawler, requiring the player to complete each level, navigating a complex maze of tunnels to try and find an exit before the timer runs out. On each level there are a selection of enemies that they must deal with. So far, there are only bats and slimes, though I may add in others later on.

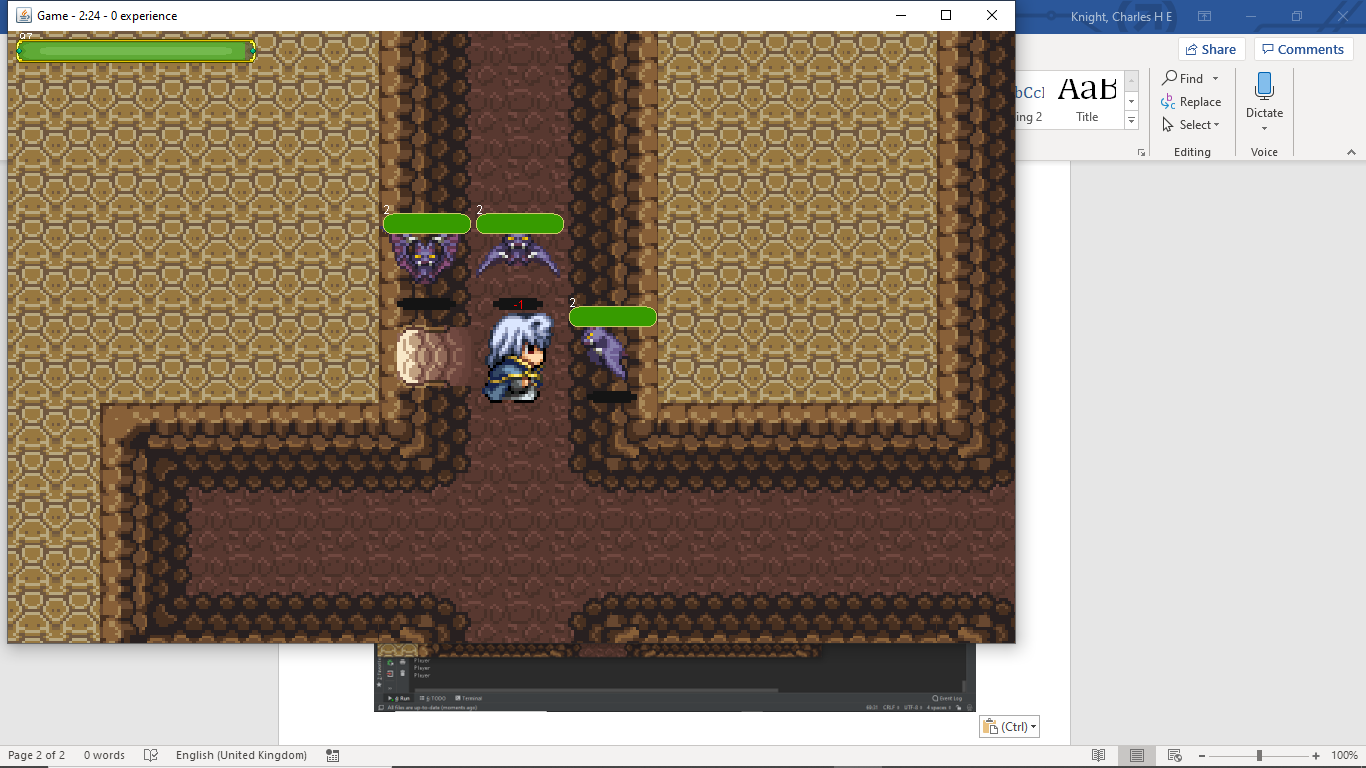
If you look closely at the title bar of the window, you’ll notice a timer and an experience counter. Experience is gained by killing enemies, and is supposed to be the score each player will increase. If the timer runs out, the game ends. Throughout these screenshots, you’ll notice both the timer and experience values change. This is normal behaviour.



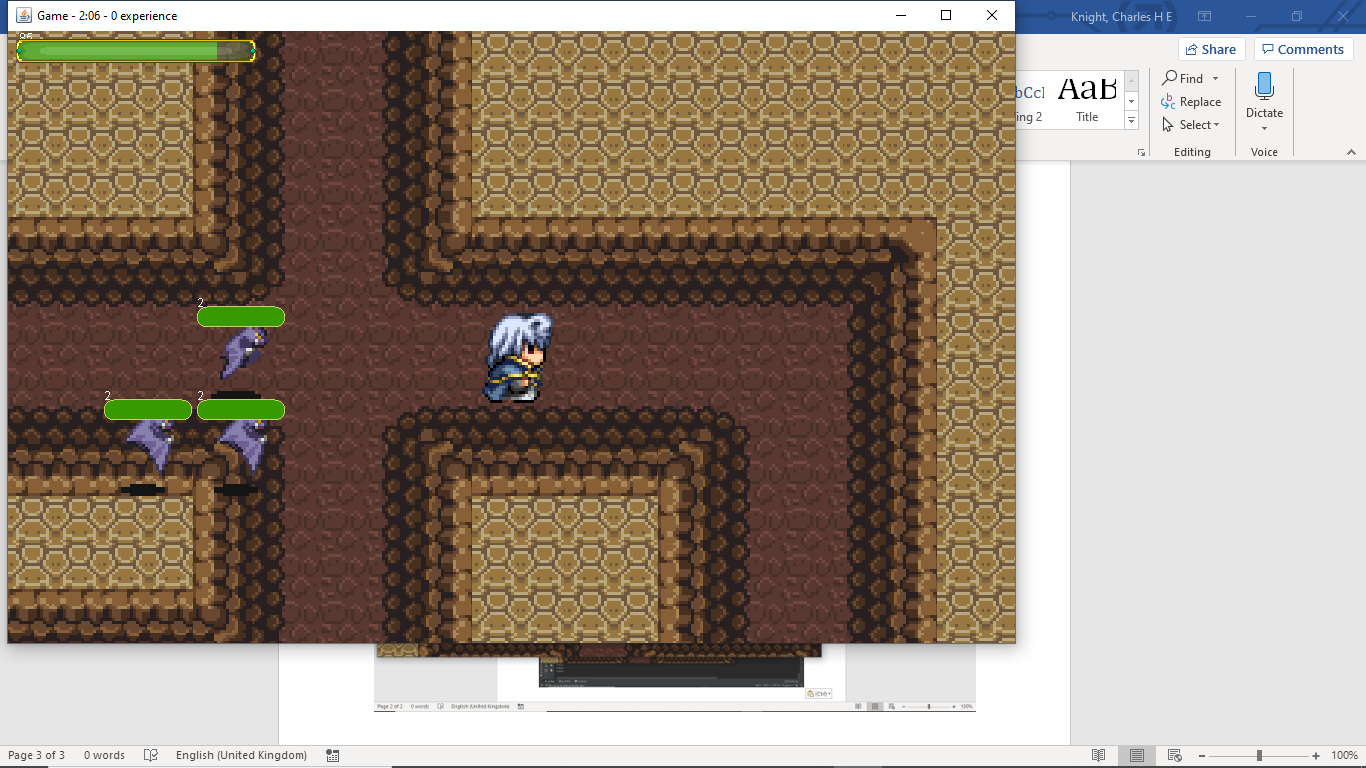




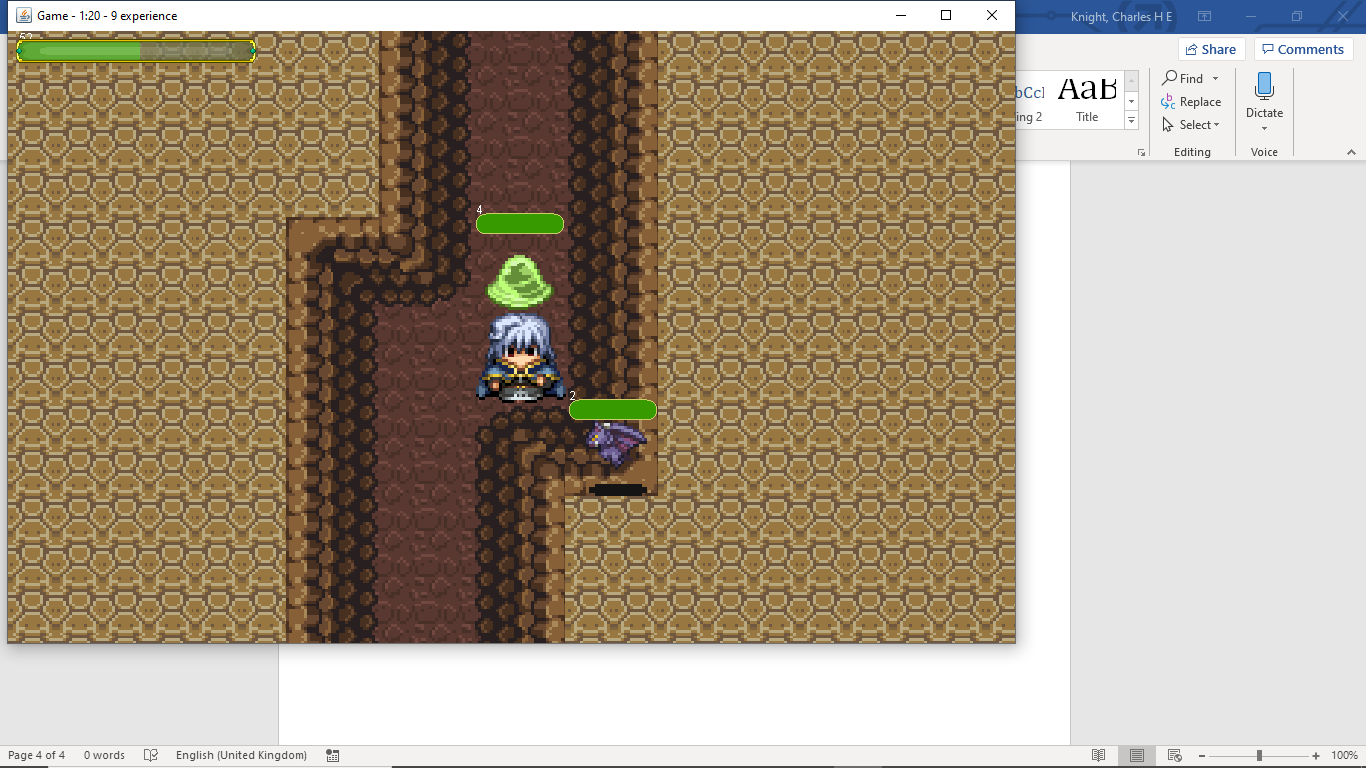
As can be seen, bats will crowd around you. Their AI is not perfect; the bat to the left is simply being idle, but it is passable.



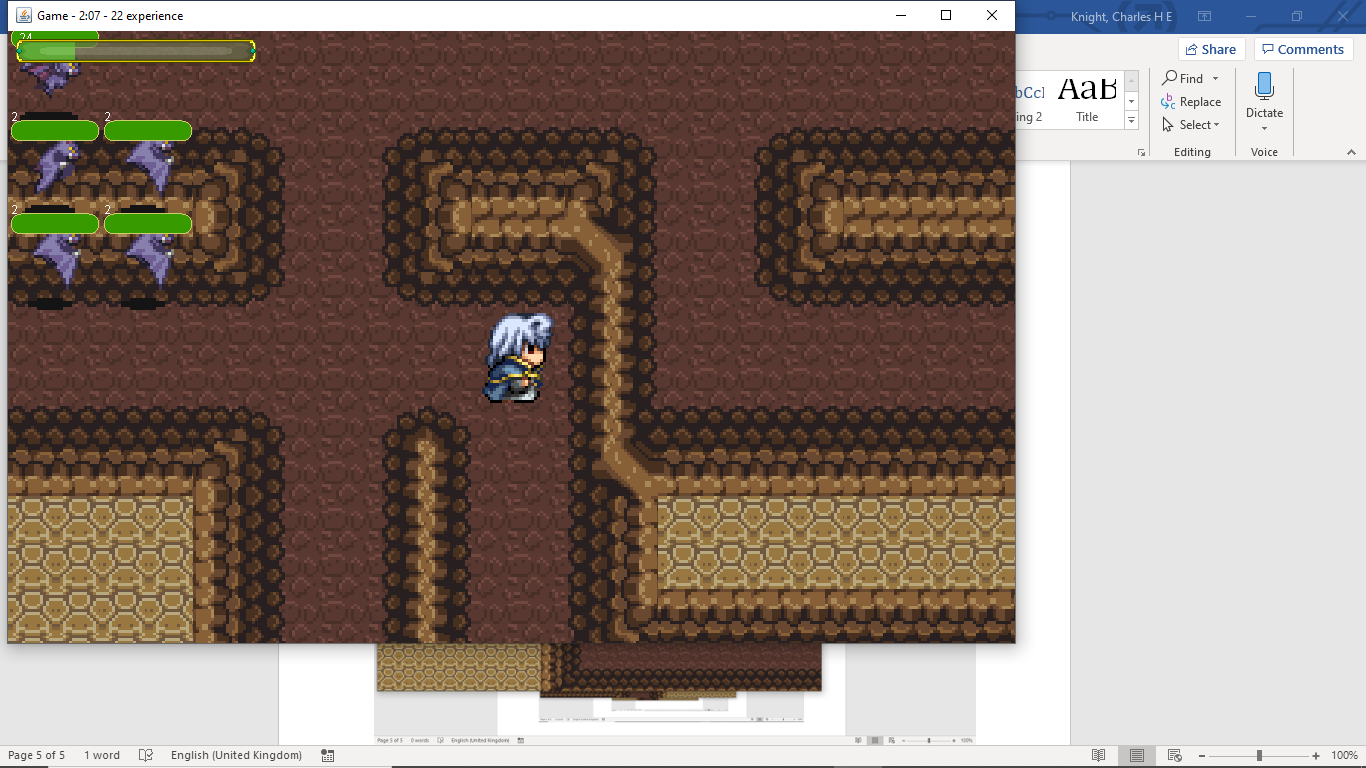
If you run, they will follow you, and though you can’t walk through walls, they can fly and so glide above them.

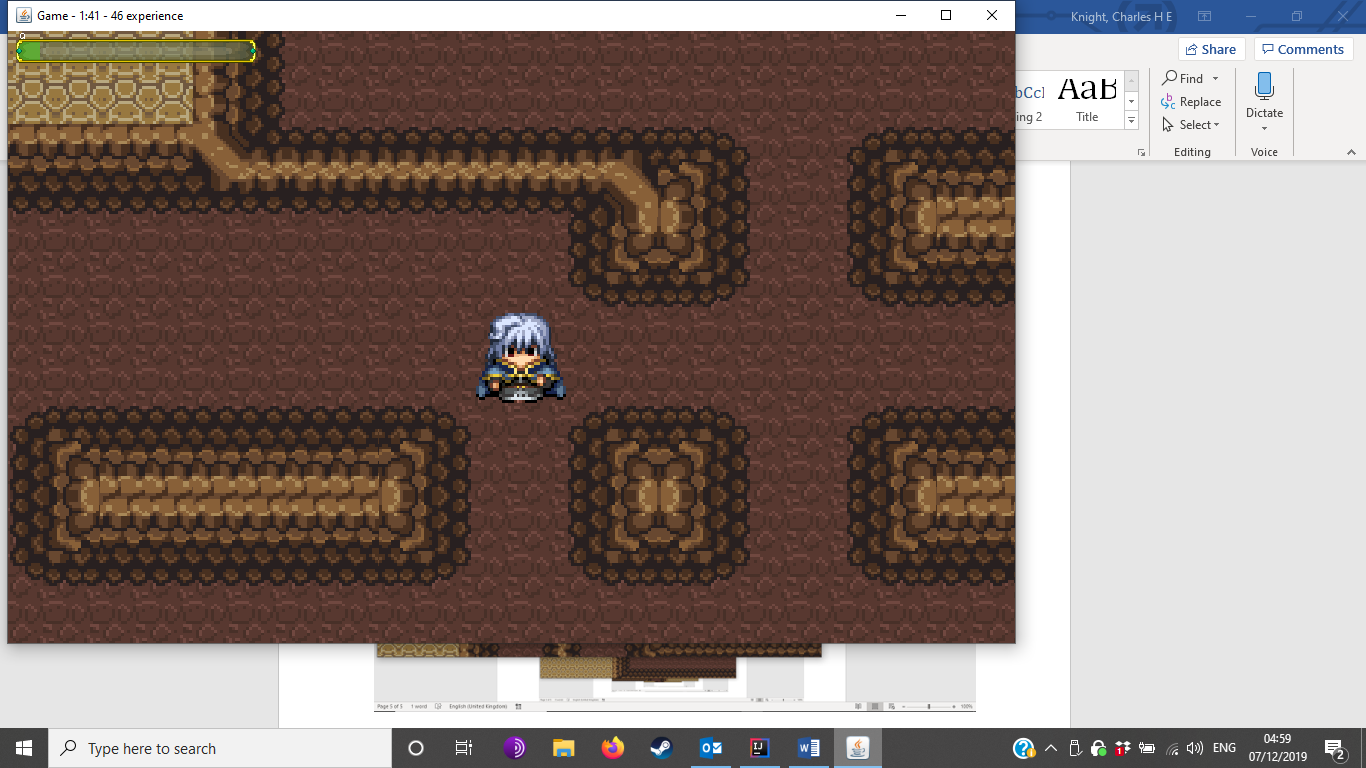


As can be seen, I have taken considerable damage, and so my healthbar is rather depleted.

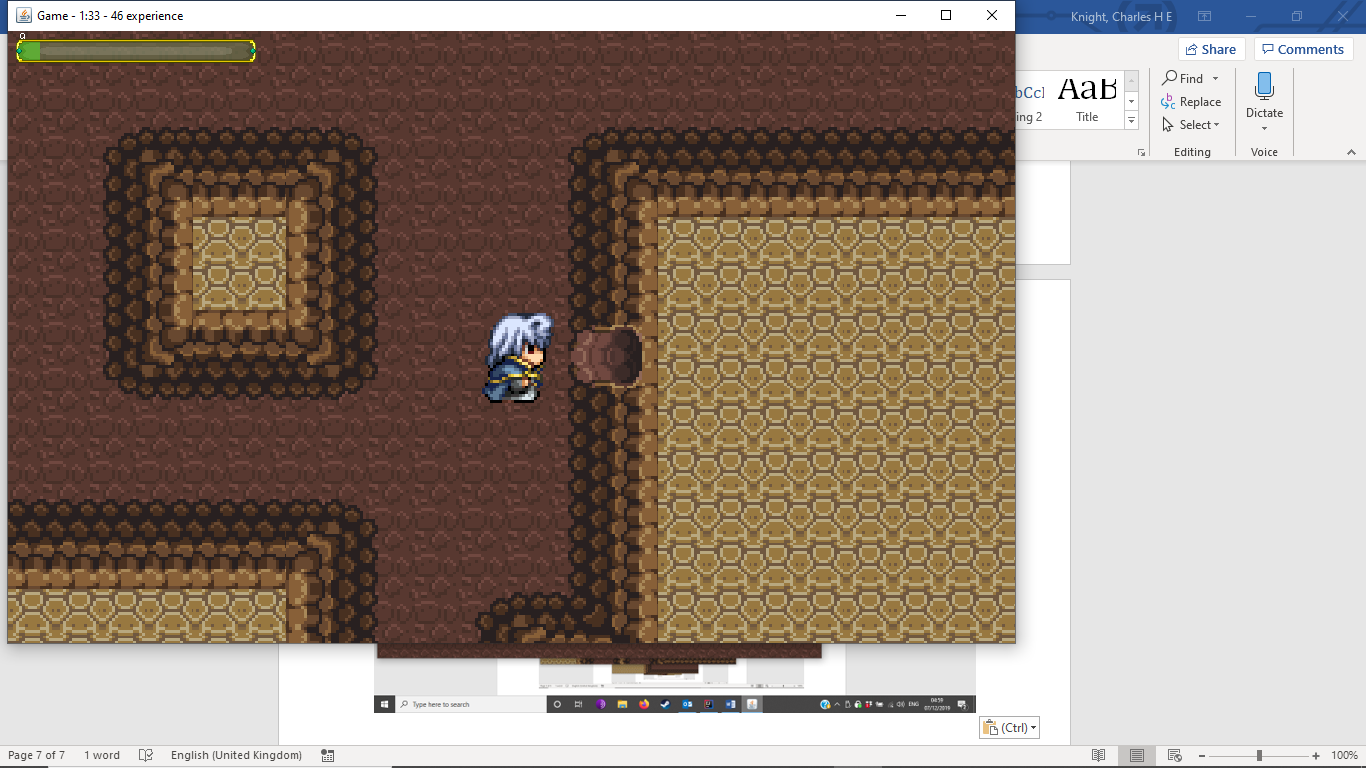


The levels autotile themselves, creating connections based on a spritesheet.



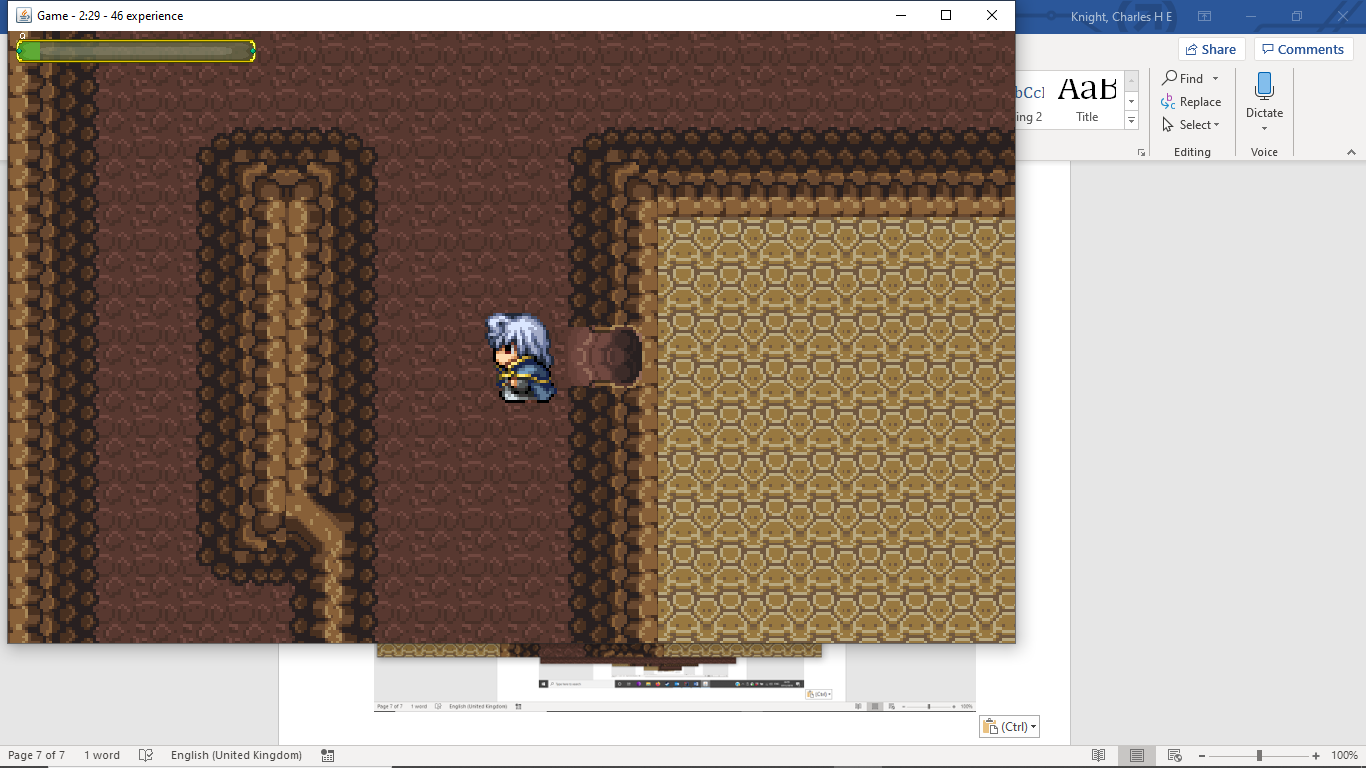


This is an exit to a level. Note the uncleared rubble in front of it.

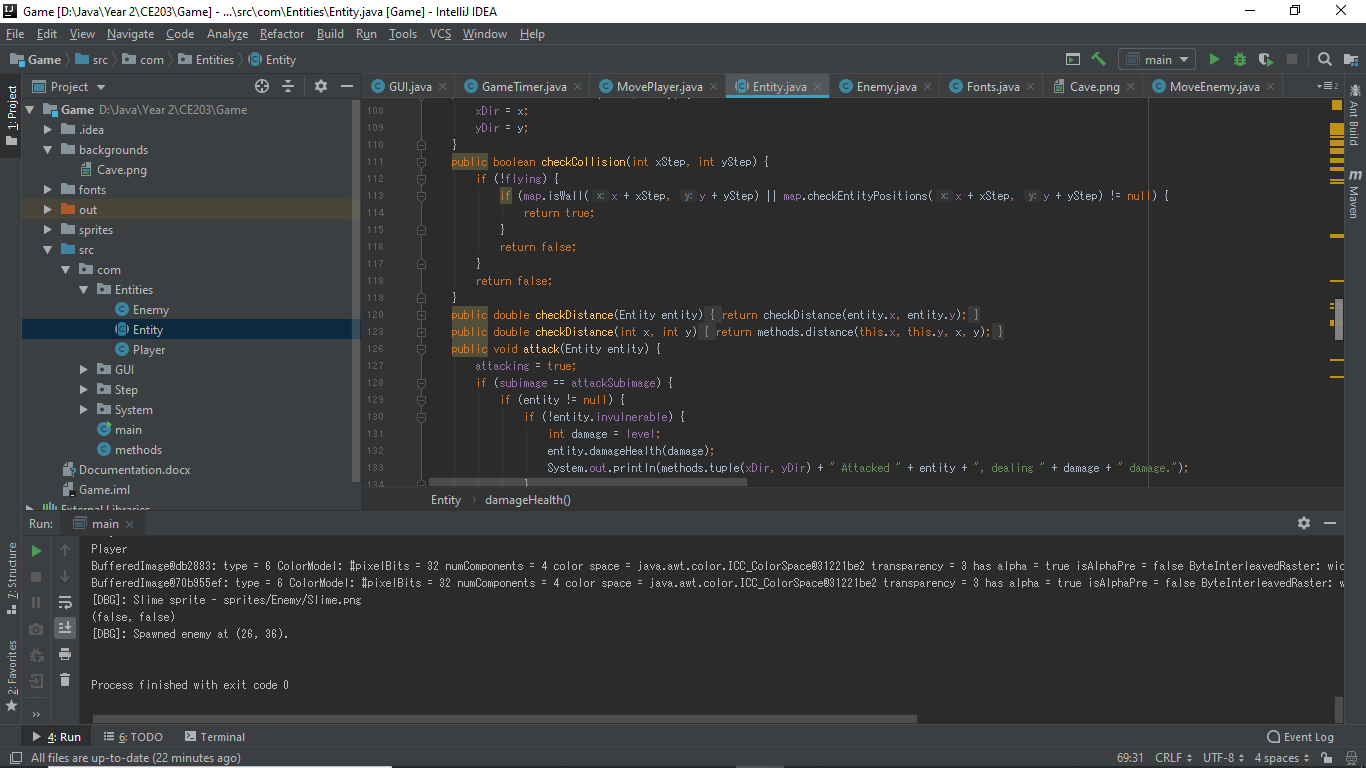


V

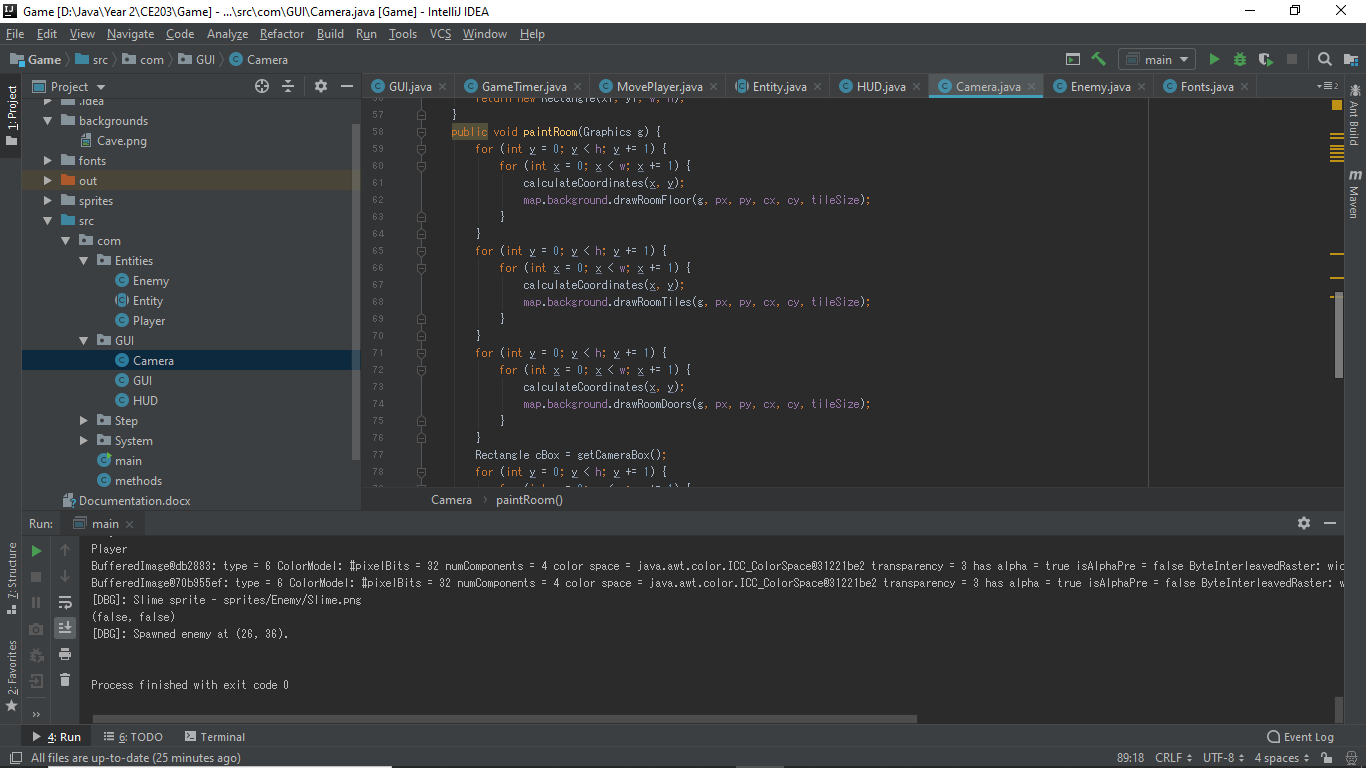
An entrance to a lower level looks like this. Note the unlit, cleared path. It is also randomly generated.



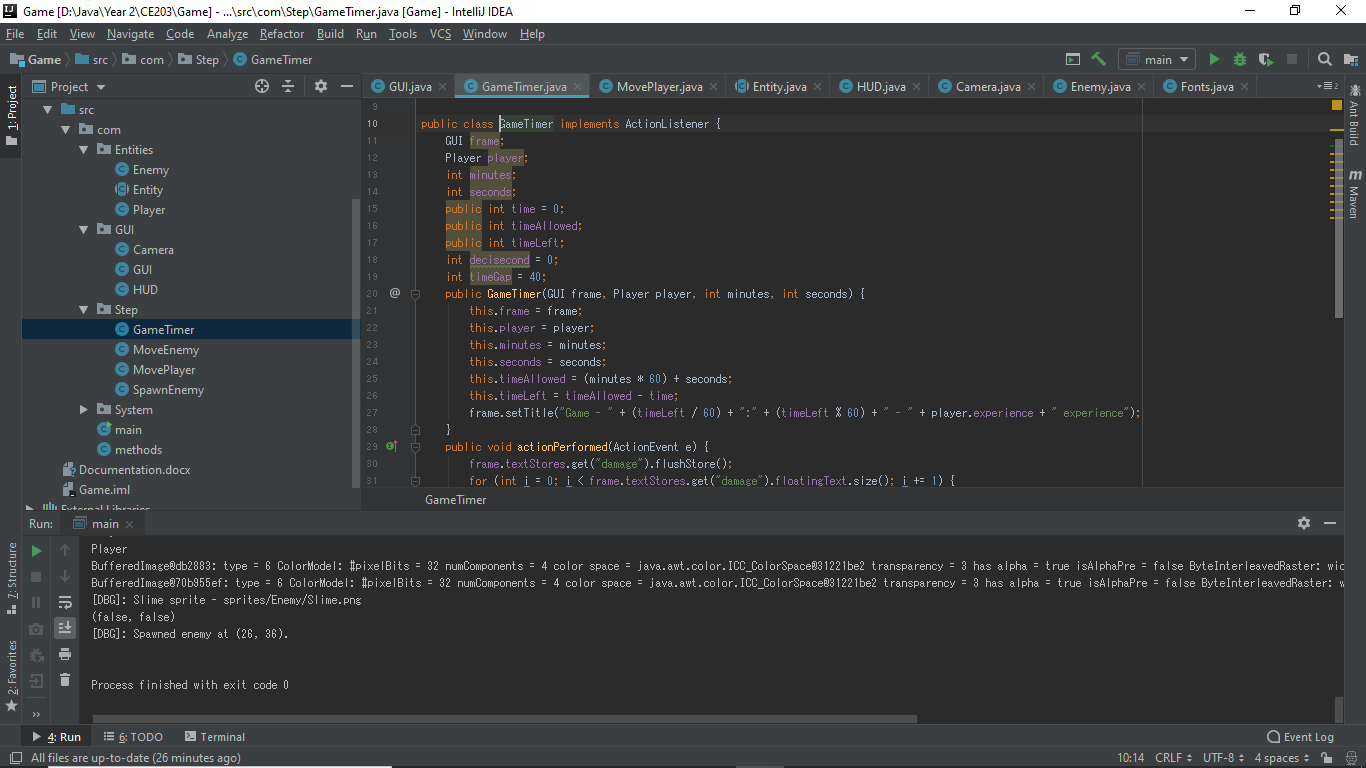
Entity objects are delineated between Players and Enemies, with the abstract class Entity containing the common logic between them.



The camera paints all objects in the room, using various functions from different classes to draw them onto the frame.



GameTimer handles the basic step functions of the game, including flushing unused text and handling the countdown clock.



The players and enemies are also moved on individual timers as well, as well as enemy spawning.

