# 

# Overview

# Automatic Configuration

The included script will compile and run the game for you.

To use the included script:

1. Open a terminal and navigate to the root of the game directory (DungeonsOfDooom)
2. Run the Play.sh script using the following command (without the quotation marks): “./Play.sh”
3. This will compile and start a server and client in GUI mode
4. If you would like a different configuration, use the manual configuration option below

# Manual Configuration

## Compilation

1. Open a terminal and navigate to the root of the game directory (DungeonsOfDooom)
2. Type the following commands (without quotation marks):
3. “mkdir out”
4. “cd src”
5. “javac -d ../out \*.java”
6. cd ..

## Playing the game

### Running the server

#### GUI Mode

1. Open a terminal and navigate to the root of the game directory (DungeonsOfDooom)
2. Enter the following command: “java -cp out Server”
3. If you would like to specify a port, type the port number at the end of the command, leaving a space before
4. A file choosing dialog will open, allowing you to select a map

Example: “java -cp out Server 40004”

In this case, 40004 is the port number of the server.

#### Command Line Mode

You can also run the server in command line mode. Follow the instructions for GUI mode, but add a ‘nogui’ argument to the end of the command.

Example: “java -cp out Server 40004 nogui”

In this case, 40004 is the port number of the server.

### Running the client

#### GUI Mode

1. Open a terminal and navigate to the root of the game directory (DungeonsOfDooom)
2. Enter the following command: “java -cp out PlayGame”

#### Textual Mode

As with the server, you can also run the client in command line mode. Append the command for GUI Mode with a ‘nogui’ argument. You will also need to specify the ip address and port number of the server as arguments.

Example: “java -cp out PlayGame nogui 127.0.0.1 40004”

In this case, 127.0.0.1 is the IP Address of the server, and 40004 is the port number.

# Analysis

## What I did right

### Fog of War

The client only receives its visible look window when it looks, and keeps track of it position relative to where it spawned. Using these, it uses the ClientMap class to build an expandable map as it moves around, creating a fog of war style map. This allows for a whole view of discovered areas, whilst preventing cheating and reducing the size of look packets. Other players are only displayed in the current look window around the player.

### AI

My AI uses a task based system that I developed. The bot class contains a stack of tasks, and gets commands from these tasks until they are done, before removing them from the stack. The base task (ExploreTask) is only complete once the game is won.

I have implemented A\* pathfinding in the ClientMap class, which the bot uses to remember where it has been. This allows the ExploreTask to explore efficiently, by finding the nearest reachable, unexplored area and getting the fastest route to that area. Pathfinding is adaptable, so if a player gets in the way, the bot will move around them.

The explore task does this exploration activity until it finds a piece of gold, at which point it adds a CollectGoldTask to the stack. If it doesn’t need gold, it will move to the nearest exit instead.

### MapPanel

My map panel has continuous, smooth transitioning between tiles, even though the player position is quantized to the nearest tile. This gives the appearance of smooth walking.

The player is also rendered facing the direction they last moved. This adds polish to the game.