Exercise 3 - Play with your friends

While making our requirements it became painfully obvious to us that we had to rewrite much of our game to correctly and more easily implement the required features. We discovered the existance of LibGDX' Scene and Stage classes while investigating how to create a menu.

In the old code (the code we handed in for the first deadline), a button, for example, was nothing more than a texture drawn somewhere on the screen with an invisible region defined on it that received click events. Of course, this is how all widgets work in theory but this would make it really cumbersome to define menus with. A second issue was that, for some reason, these invisible regions remained active when another menu was shown resulting in invisible buttons messing up the program flow. On top of this, there were even more reasons our old code was not sustainable for a multiplayer solution (a quick example that comes to mind is defining a TextField for IP addresses).

Thus, we set out for a (severe) rewrite of our code. This has resulted in a much more easy to maintain code base, that, while still being a bit messy (we had no time left to clean it up completely) is much more logical to us than the old. Our Grid and Tile objects now are Actors (a LibGDX class), which makes it very easy to draw and position them, allowing us to drop the CoordinateHandler. In fact, many handlers were dropped as the LibGDX classes took over their functionality.

We are especially happy with our abstract Screen class, that in combination with the ScreenHandler makes it very easy to browse through menus and start different games.

A downside of all this is that there is no more a clear separation between the model, the view and the controller. That is something we strived to achieve in the first version of our code. However, we are certain that our current implementation is much better than what the old would have been had we kept using it.

Another downside is that testing got harder, because most objects now required textures upon creation. Textures do not exist in the headless backend of LibGDX, making it impossible for us to test the classes. Our solution was to add a second constructor, allowing us to inject mock objects instead of textures. Again, this solution is not the best (test-specific code is never good), but we feel it is a good compromise between easy-to-maintain code and testable code.