# Code Review 3

## 1. GameConfig initialization

Consider using more strongly typed values for the constructor of the GameConfig object. For example, initialization currently takes values as such:

```
GameConfig gameConfig{name, minPlayers, maxPlayers, hasAudience,
setupPairs};
```

where minPlayers and maxPlayers are both integer values. As such, it is easy for the values to become mixed up when passed into the object initialization (e.g. switching min and max). Consider using a struct with a min and max value to avoid possible misuse:

```
GameConfig gameConfig{name, playerMinMax, hasAudience, setupPairs};
```

## 2. MessageType clarity

The MessageType enum currently has two very similar enumerations, namely: OUTPUT_MSG and OUTPUT_G_MSG.

It's not immediately clear what the difference between the two are, so perhaps consider providing more clarity in the naming scheme.

Furthermore, there is currently a typo in the class, with the two enumerations: OUPUT_G_MSG and OUPUT_SCORES.

## 3. Unnecessary ifstream.close()

In the constructor for the JsonParser class, it currently ends with a call to ifstream.close(). This line is unnecessary, as once the ifstream is out of scope it will be closed automatically. Consider removing the line from the function.

## 4. GameMessage private fields

The GameMessage class currently holds many different private variables which may be outside the scope of a game message. Consider encapsulating some of these variables into a MessageContents class for example, which may help to clarify the purpose of the class.

## 5. GameSession class design

The GameSession class seems well designed, with self-documenting function names and variable names. However, consider encapsulating the players and audience vectors into a more

strongly typed struct (e.g. a Connections struct) to reduce redundancy of having two identically typed variables.