Component Design

A game for the playmaker release of the GF platform will consist of three parts; one server-side part and two client-side parts -- one for the fixed platform and one for the mobile platform.

Each part of the game will consist of one or more source-components, and any number of binary components. (see Component Packaging.doc for more information on the different types of components).

Architecture

The GF platform requires the game component to fulfill the following interfaces:

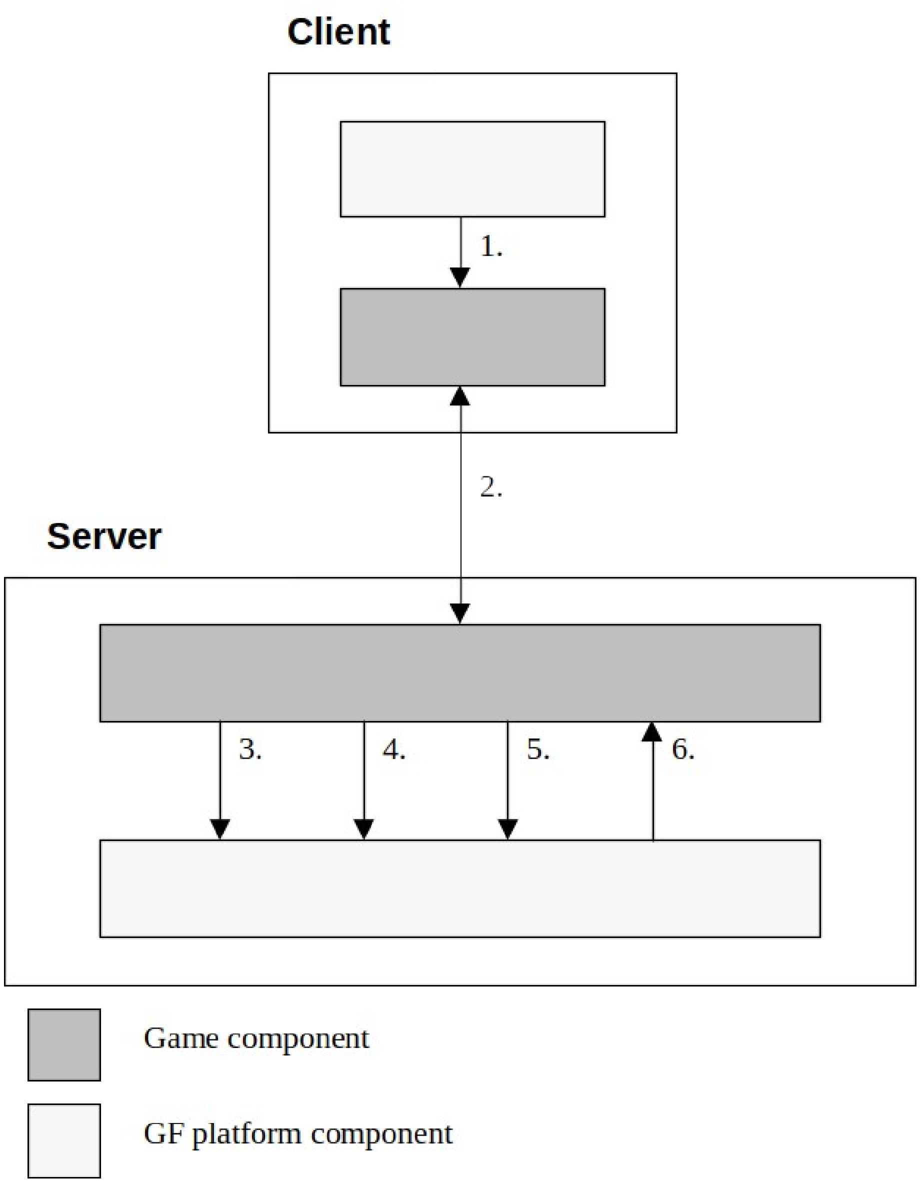

Figure 1, component architecture.

1. `com.gamefederation.playmaker.component.IComponentStateControl`

   Required, the client game components need to *implement* this interface to be controllable by the platform.

2. Game internal communication between client and corresponding server side game component.

3. `com.gamefederation.playmaker.??tick??`

   Additional, server game components can *use* the interface to send ticks during the game session.

4. `com.gamefederation.playmaker.session.ISessionUserStateControl`

   Additional, server game components can *use* the interface to control the individual users connected to the game.

5. `com.gamefederation.playmaker.session.ISessionMainStateControl`

   Additional, server game components can *use* the interface to control the flow of the game and to delete users.

6. `com.gamefederation.playmaker.component.IComponentStateControl`

   Required, the server game component need to *implement* this interface to be controllable by the platform.

Server side game components:

The server side part will *implement* the `com.gamefederation.playmaker.component.IComponentStateControl` interface to be able to be run under the GF platform. When the component is first started, init() is called, then stop(), pause() and start() are used to control the component.
For example, the game should pause when pause() is called, and resume the action when start() is called.

The server can also *use* the following interfaces:
- `com.gamefederation.playmaker.session.ISessionMainStateControl`
  Used to control the flow of the game and to delete users.
- `com.gamefederation.playmaker.session.ISessionUserStateControl`
  Used to control the individual users connected to the game.

Instances of both those interfaces are passed to the init() call to the server.

Client side game components:

The client (PC and mobile) also need to *implement* the `com.gamefederation.playmaker.component.IComponentStateControl` interface,

but the init() call will send a S*tring* and an *int* with the IP address and the port of the server side part of the game.
The client will then connect to that IP:port and use that connection to send all the internal game data. On the mobile client, this will have to be tunneled via http, but on the full client, you can use this like just a normal TCP/IP socket.

See the Java doc over the component API for an exact reference of the interfaces and how the game should implement them.