GLADDOS SERVER DOCUMENTATION

HOW CLIENTS ARE TRACKED BY THE SERVER

The way that the server tracks clients is by client id's. The server object has a private variable that keeps track of the number of client's that have connected to the server. Each time a client connects, they are assigned the current client count number, and then that count is incremented. This number will then be referred to the entire duration of their connection as their clientID. This ID will then be mapped to a Being object that represents the player.

HOW DISCONNECTIONS ARE HANDLED:

There are 4 scenarios of disconnections that need to be considered with a server and client architecture. They are as follows:

1) The client disconnection is expected by the server
    a. Occurs when the user disconnects as is intended by the program.
2) The client disconnection is unexpected by the server
    a. Occurs when the client crashes for an unknown reason or some other factor causes the server to not be informed of the crash
3) The server disconnection is expected by the client
    a. Occurs when the server shuts down as is intended by the program.
4) The server disconnection is unexpected by the client.
    a. Occurs when the server crashes or some other factors causes the client to not be informed of the crash.

How these 4 situations are handled:

The server will handle an expected disconnection from a client in the form of an event. When a client disconnects, they will send a "DisconnectEvent" event to the server. The server will process that and perform the appropriate cleanup for the client. Once it has removed the client, it will update the ServerWorldState and then send out an Update that reflects this change to all remaining connected clients.

The way the server will handle unexpected disconnections is through a maximum timeout set on each client socket. If a client exceeds this timeout limit, the server will perform disconnection cleanup on them. Or if the server can't send an update to the client, it will attempt to send it a maximum of 5 times before disconnecting the client.

The way that the client will handle an unexpected server disconnect is through the way it sends events. If a client attempts to send an event and it receives an exception, it will try to send the event a maximum of 5 times before it will assume the server was disconnected. At this point, the client will perform the appropriate client-side cleanup outlined below to disconnect and cleanup the connection to the server.

The way the client will handle an expected server disconnect is through an Update object sent by the server. When the server has to shut down for some reason, it will send out a "ServerShutdownUpdate" update. This update will inform the users the server disconnected and then perform the appropriate cleanup. The users will then handle expected server disconnection cleanup.

Cleanup Procedures for client disconnection

Server-side cleanup

1) Remove the client's ObjectOutPutStream from the Broadcaster object.
2) Stop the ClientListener thread
3) Close the socket to the client
4) Remove the ClientListener from the clients HashMap
5) Send out an update that reflects the removed client.

Client-side cleanup

1) Send DisconnectEvent event to server
2) Close the socket to the server
3) Stop the Client thread from running

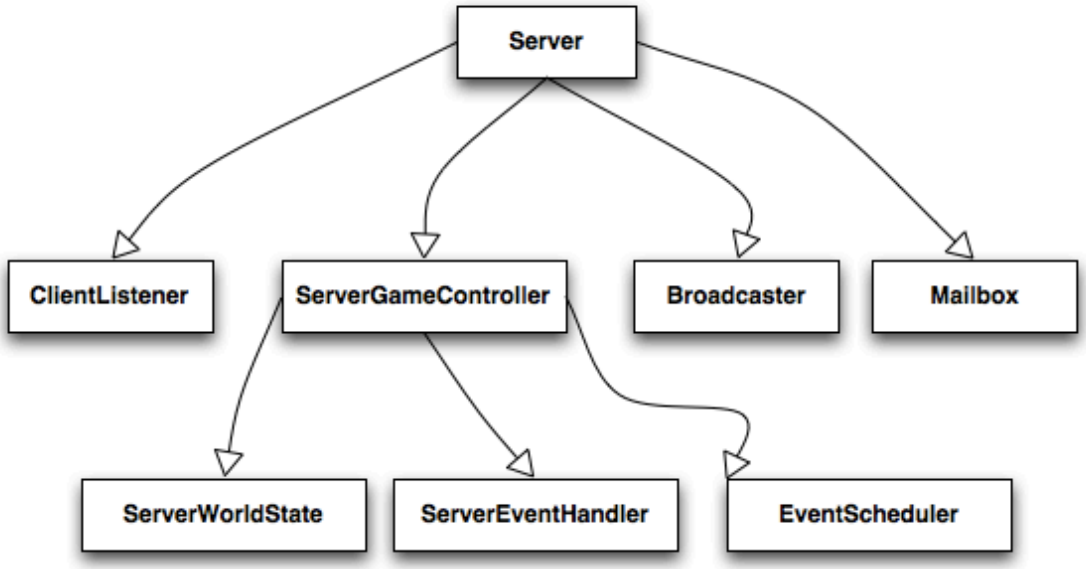
How new clients are handled:

When a new client connects to the server, the server performs the following steps:

1) Creates a ClientListener thread on the client's socket

2) Adds the client's ObjectOutputStream to the Broadcaster's list of clients.
    a. After the Broadcaster adds the client, it then immediately broadcasts a NewGameUpdate to the client from which they can create their game state.
3) Places a NewGameClient event in the mailbox to be processed by the ServerGameController in order to update all players of the new connection.
    a. When this event is executed, it will create an update with the newly connected player added in, this update will then be to broadcasted to all players.
4) Adds the ClientListener thread to a HashMap that maps the client's id to it's associated ClientListener thread.

Object Instantiation Server-Side



Object Instantiation Client-side