Project Multitris

Group 23
Marcus Dicander
Måns Olson
Tomas Alaeus
Daniel Boström
Oscar Olsson

2.2. Overall Architecture Description

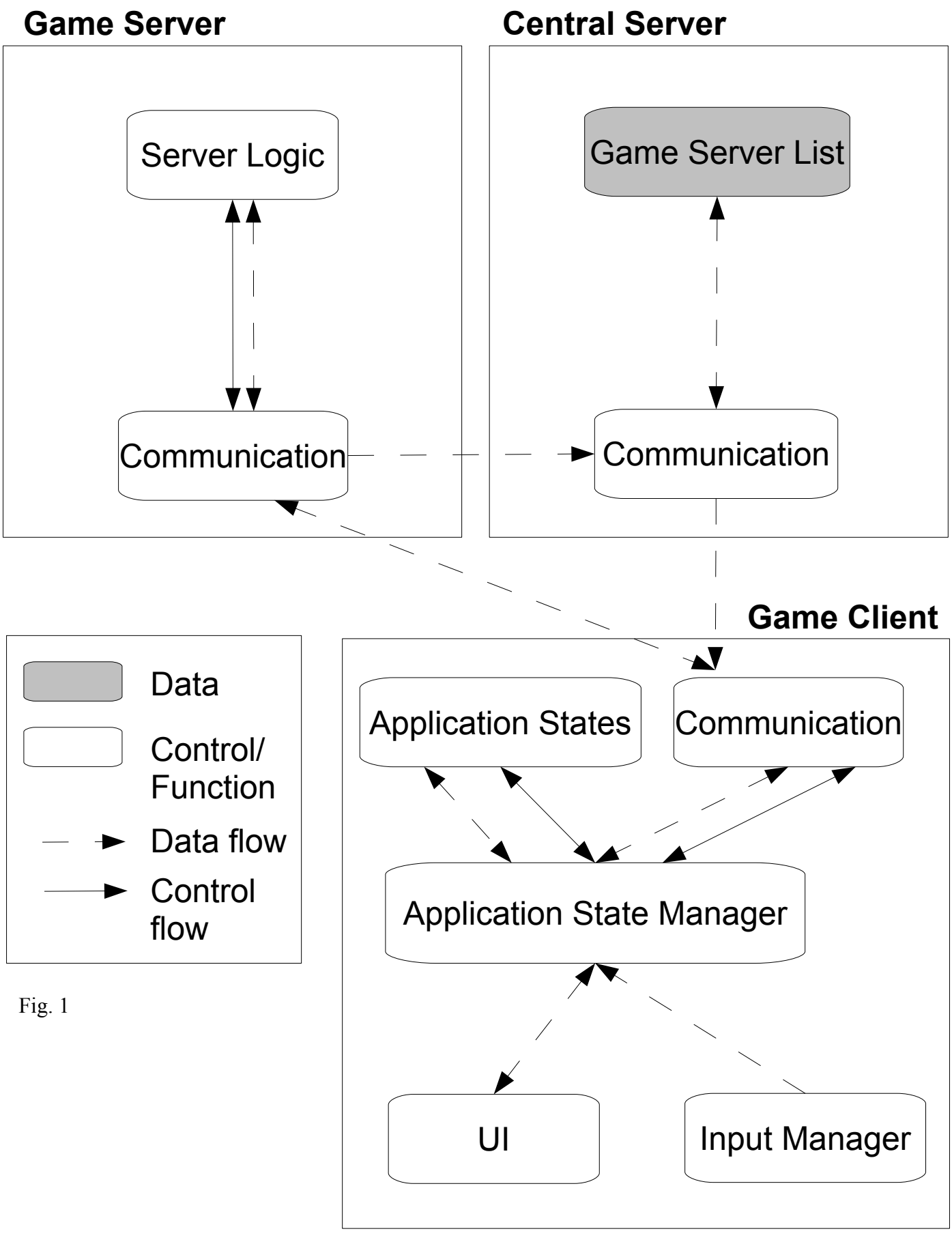

Fig. 1

Multitris is a multiplayer game system organized into a client-server architecture. Figure 1 is a Box-Line diagram that shows the overall architecture. The system consists of three major parts; the Central Server, the Game Server, and the Game Client. The Central Server keeps a list of Game Servers, which is publicly available to any Game Clients. Clients can the Central Server to find Game Servers. A Client can then request a connection to one of the Game Servers in the list.

The Game Server's task is to keep track of a Game Session. A number of Clients can connect to a Game Server to participate in a Game Session (i.e. playing together). To change the state of the game, for example by making a move, the Client sends a request to the Game Server.

Each of the three major parts consists of a number of subsystems. The Central Server has a Game Server List and a Communication interface. Game Servers that wish to be added or removed to the list can communicate this via the Communication interface. The Central Server can also notify Clients of available Game Servers via the Communication interface.

The Game Server has a Communication interface that is used to communicate with the Game Client and Central Server. Incoming requests are passed on to the Server Logic subsystem. The Server Logic gathers any action requests from the Game Clients, and then notifies each Client of all action requests made via the Communication interface. Further, the Server Logic subsystem can request the Game Server to be added or removed from the Central Server's Game List via the Communication interface.

The Game Client consists of a Communication interface, a Application State Manager, a User Interface subsystem, and an Input Manager. Player input is handled by the Input Manager, which is passed on to the Application State Manager. The Application State Manager then passes on the data to another system, depending on the input. The Client regularly recieves a list of requested actions from the Game Server, and these are then passed on to the Application State Manager which makes changes to the Application States accordingly.

2.3. Detailed Architecture

Local player action

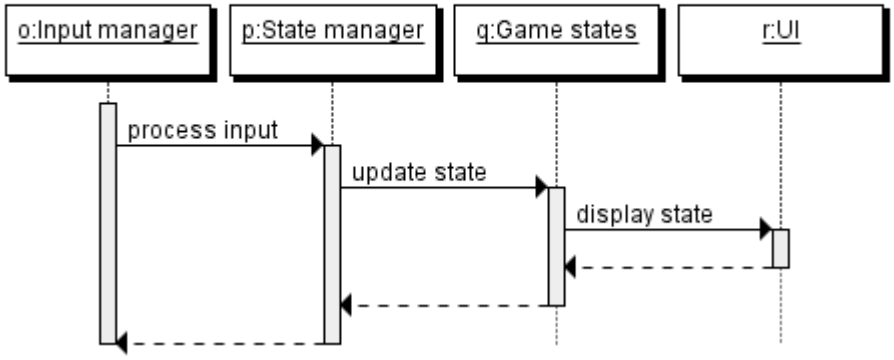

Fig. 2

Remote player action

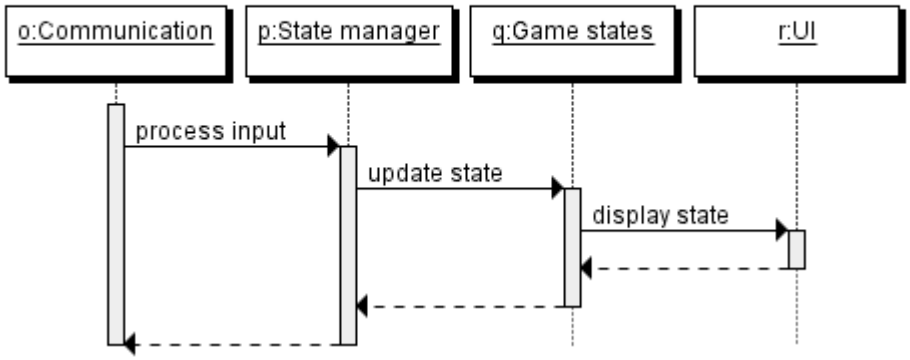

Fig. 3

Game server

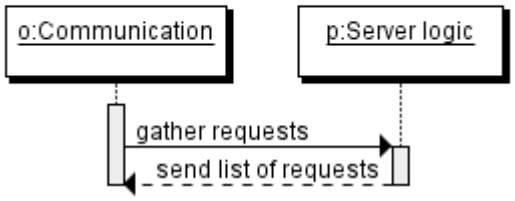

Fig. 4

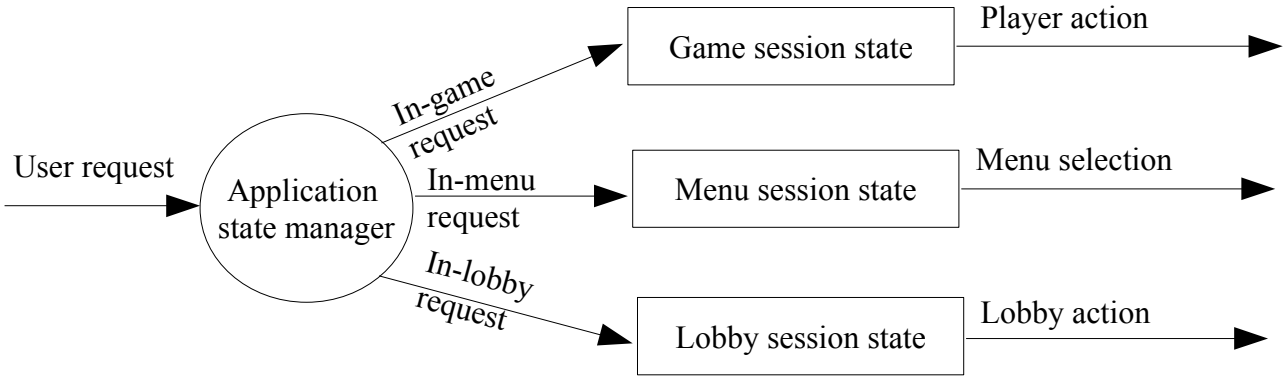

Fig. 5

Figures 2 through 4 above are sequence diagrams showing control and data flow in the system. Figure 5 is a Box-Line diagram of how decisions are made in the Application State Manager. The process is explained in detail below.

The Application State Manager's responsibility is to provide a single interface to the different Application States. Any change requested by either the User or the Game Server is routed through the Application State Manager. Depending on the type of change, the Application State Manager will pass on the request to a different Application State, which will in turn handle the request.

At all times, User input is passed on to the currently active Application State. Important Application States are the Menu State, the Lobby State, and the Game Session State. The Menu State uses User input to provide menu functionality. The Lobby State is active when the User is waiting for a Game Session to start. When it does, the Game Session State will be the active state.

Possible requests are movement requests from the User (for example navigating a menu, or moving a piece while in a Game Session), and requests to open or close a Game Server for additional users. In some cases, the requests will need to be passed on via the Communication interface to the server.