Balance of Power: a Strategic RealTime MultiPlayer Game

The Setting:

The game is set in a universe consisting of a number of planets. The basis of existence of a planet is its energy. Each planet starts with the same amount of energy, all is in balance and peaceful. What a wonderful world.

Enter the human players: each player controls a planet: planets can interact, in a supportive or aggressive way, both of which are expressed in (forceful or voluntary) exchange of energy, the behavior can hence be aggressive or supportive:

Aggressive:

- **Absorb**: a planet can absorb another planet's energy --- per time unit a certain amount can be transferred, which is subtracted from the assaulted planet, and added without loss to the aggressor.
- **Attack**: A planet can attack another planet in a single strong attack: half of the aggressor's energy is subtracted from both, the attacker and the attacked. This can entirely destroy the attacked planet, if the attacker was more than double as energetic as the victim. Being weakened from wrath, a planet having actively performed an attack cannot act in any other way for a certain amount of recovery time.

Supportive:

Both of the aggressive actions can be similarly performed in the opposite way:

- **Infusion**: A planet can generously transfer its own energy to another planet in need, same amount per time unit as in the attack.
- **Donation**: a planet can shock transfer half of its energy to another planet in a single shot. Similar to an attack, being weakened from generosity, a planet having actively performed an energy donation cannot act in any other way for a certain amount of recovery time.

Winner:

I would not necessarily define a winning state --- this game might be more interesting if you see it as a psychological experiment / political simulation. You can decide for yourself, if your goal in the game should be to keep the power in balance, or to gain as much energy as possible to destroy other planets. Play different roles to see what happens!

Death of a planet:

If the energy level of a planet falls below zero, the planet dies without chance of recovery: no actions can be performed from this planet.

Notes:

Absorb, Infusion and Shock transfer do not change the accumulated energy of the system, attack does.

The players in this game are humans. Given the actions at hand, alliances can develop, friendships can be destroyed, all is possible. Add rules as needed. The set of rules above is the minimum.

A planet under absorbing-action can neutralize the energy loss by a counter-absorbing-action. However, the enemies cannot perform any other action at that time, making themselves vulnerable to third parties.

Triggering the actions:

You select the planet to support/attack by mouse click (mouse-picker). With a target selected, the following keys should trigger the actions (if you are left handed, you might want to change this key arrangement):

- a absorb (key down: start, key-up: end)
- A attack (key down only)
- s infusion (key down: start, key-up: end)
- S donation (key down only)

Only a single action can be performed at a time. Use the key pressed/up flag to detect the start/end of an absorb-action or infusion-action.

Note: do not use an analog listener here, you don't want to transmit data continuously to the server. Only transmit start and end of the action.

Graphic representation:

- **Health (energy level) of a planet**: represent the health by color and/or transparency of a planet. If you generate textured planets, you might want to add rotation speed or anything else that comes to mind.
- **Selecting a planet**: grey arrow from the source to the target sphere
- **Absorb**: red arrow from source to target
- **Attack**: red arrow from source to target (for one second), plus particle emitter explosion on target sphere

- **Infusion**: green arrow
- **Donation**: green arrow with friendly particle explosion (let there be flowers!)
-

Gaming Framework

Add a very simple framework, e.g. the game starts when n (e.g. 5) players are connected. This assignment is not about the game-framework, but about the client/server/message structure.

Help

This game is a straightforward extension of the MinimalNetworking project, which is uploaded to blackboard. The extension consists of three tasks:

1. You need to extend the server in order to implement the game logic. Please be aware that the server is not a JMonkey application (i.e. it is not extended from SimpleApplication), but just a normal JAVA application importing some jme classes. In order to run your state machines, just generate you own mainloop.
2. You need to extend the clients in order to represent the graphics. Important: all logic is set in the server, the clients only take input, and provide graphic output.
3. You need to define your own messages to interact between client (game I/O) and server (game logic).

In this assignment you see, probably for the first time, a really strict separation between I/O routines and game logic: I/O and logic may even be physically separated, running on different machines!

One of the main design tasks is to identify the state machines in the server, but also the messages. Start thinking about the messages first, this will ease the approach to the project. Implement all messages needed first, using empty stubs (or console prints) as reaction to them. Once having the messaging down, build up the game one action at a time.

Think before you code! Writing games like this needs a lot of discipline. Quick hacks will hurt you, I promise.