

Test Scenario for ZoidCom's Integrationtest

Goal

The goal of the test is to determine how a network engine design might look like and what's necessary to integrate the network engine (and thereby the network framework) into the game engine.

Furthermore this test should uncover any fundamental problems induced by ZoidCom including any potential performance problems.

Measuring the efficiency

- CPU usage: To discover any spikes in the CPU usage, the application's CPU usage will be monitored using Window's Task Manager.

- the time for the transmission is measured in the following way for each test case: (in ms)

TestCase 1:

time 1 = endtime of step 3 – starttime of step 2

time 2 = endtime of step 4 – starttime of step 4

time 3 = endtime of step 5 – starttime of step 5

TestCase 2/3:

time = endtime of step 4 – starttime of step 4

TestCase 1: Simple X Universe Environment

1. The server creates 101 (max clients + 1) instances of the class: TestObject

```
{  
    int i;  
    string s;  
    float[3][3] f;  
}
```

This simulates the game universe.

2. Whenever a client connects, the server creates a new instance of the class: TestObject. The owner of this object is the connected client.

This simulates the player's spaceship.

3. All the players must have an initial proper representation of the universe.

4. The following loop runs for 100 iterations

- do nothing for x-1 iterations (where x is the number of the connected player)
- otherwise alter the float of the own playerObject and update the server's/client's data
- wait for 100 ms

Odd iterations (1,3,5,...) will be expected to result in immediate changes on all clients (0 ms latency), Equal iterations (2,4,6,...) will be expected to result in client changes within 2*[average lag].

This simulates the players' inputs.

5. The following loop runs for 1000 iterations

- remove one instance of TestObject
- create one instance of TestObject
- wait 5 ms

This simulates the changing environment in the game.

TestCase 2: Switching Authority Test

1. same as in TestCase 1
2. same as in TestCase 1
3. same as in TestCase 1
4. The following loop runs for X iterations (where X is the number of connected players)
 - in iteration x-1 (where x is the number of the connected player) get the ownership of 101/(X+1) objects (i.e. for two players, player 1 will get control of object 35-67 and player 2 will get the ownership of object 68-101 while the control of object 1-34 remains at the server's side).
 - alter the float value of object x (in every iteration!)
 - wait for 100 ms

TestCase 3: In-Game-Player-(Dis-)Connection Test

1. same as in TestCase 1
2. same as in TestCase 1
3. same as in TestCase 1
4. The following loop runs for X iterations (where X is the number of connected players)
 - in iteration x (where x is the number of the connected player) will disconnect
 - in iteration x+2 client no x will reconnect
 - wait for 3s