Introduction

This document is written to define the steps of using Multi-Agent Spatial Simulation (MASS) debugger to debug MASS applications. The MASS debugger supports three main features: high-tech, low-tech and interactive features. High-tech features focus on visualization and control of the holistic simulation of a MASS application. Low-tech features visualize each place/agent and show the status of the memory contents. The interactive features interact with user to control the program execution such as pause/resume, modify place/agent data.

Modules

MASS debugger contains two modules: the GUI and the backend. The GUI is used to visualize MASS application status, both a macroscopic and a microscopic view. The GUI also provides an interfaces for the user to Pause/Resume the MASS application in the backend. The MASS debugger backend is used to collect Place/Agent data from all computation nodes and send the data to the GUI. At the same time, the backend will also receive the user's commands from the GUI. The MASS debugger backend itself is a Place in terms of MASS.

Enabling MASS debugger

After all place and agent objects have been created and MASS.init() has been called, you may instantiate the debugger with a call to MASS.debugInit(placeHandle, agentHandle, portNmber). Note, the port number here is different than the port number passed to MASS.init() and must match the port number entered in the Debugger GUI. Next, to synchronize your application with the debugging GUI, you must call MASS.debugSync(). This should be placed at the start of your simulation loop. To send Place and Agent data to the GUI, a call to MASS.updateDebugger() must be made. This call should be placed at end of your simulation loop.

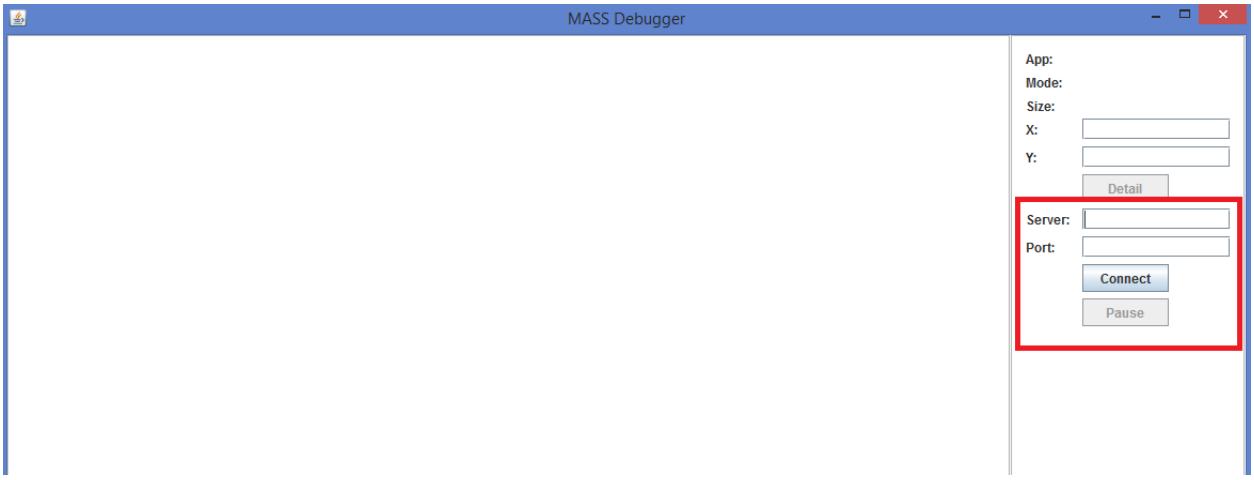
Alternatively, you may instantiate the debugger with a call to MASS.init(args, nProc, nThr, placeHandle, agentHandle, portNumber), in place of the debugInit() call.

Simple use case example:

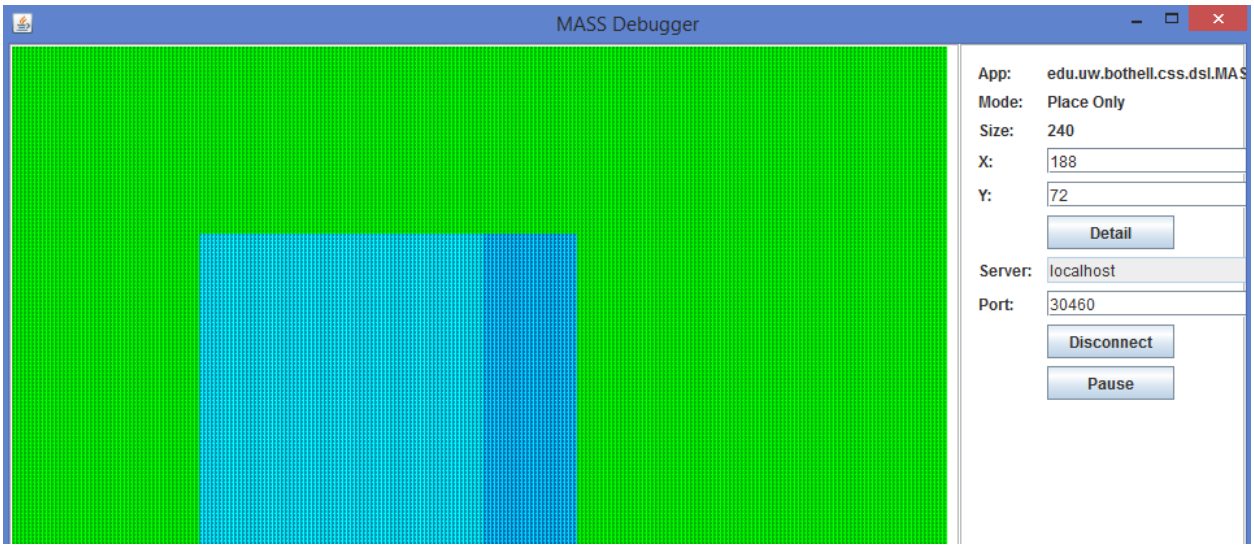
```
//initialize MASS
MASS.init();

//initialize Wave2D places
Places wave2D = new Places( WAVE_HANDLE, "Wave2DMass", new Integer(interval), size, size );
wave2D.callAll( init_ );

//initialize debugger
MASS.debugInit(WAVE_HANDLE, 0, DEBUGGING_PORT);

for ( int time = 0; time < maxTime; time++ )
{
    //waits for debugger GUI
    MASS.debugSync();

    //simulation computations
    wave2D.callAll( computeWave_, time);
    wave2D.exchangeAll( 1, exchangeWave_ );

    //updates GUI with current data
    MASS.debugUpdate();
}
MASS.finish( );
```

Running the GUI

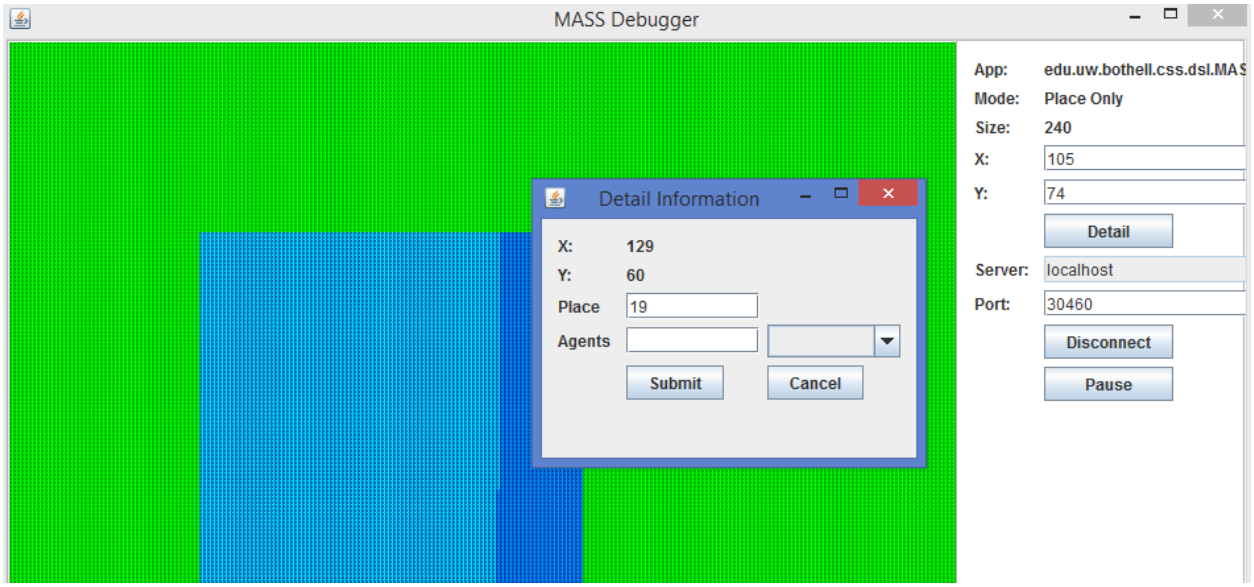
Double click to run the Debugger executable. Once running, you may enter the IP and port number of the master node of your MASS application. Afterward, click the connect button to connect the GUI to your MASS application. We recommend starting your MASS application first, debugSync() will wait for the GUI to connect.



Once a connection has been made, the simulation will begin and a visualization Place/Agent data will begin to show in the Debugger window.



Each element in the matrix may be clicked on for further information on the particular Place/Agent. The information displayed is editable at run time as well. The debugger may be paused and resumed at any time.



Next steps

3D and zoom support will be made available shortly.