Thea Render Network Rendering

Revision	Author	Reason for Change
17/01/09	Ioannis Pantazopoulos	Initial version.
05/05/12	-//-	Small update with newer screenshots.

Introduction

Network rendering is a way to remarkably improve render times by utilizing more machines to contribute in one or more renders. In our framework, there are two types of network rendering:

1. Co-operating on the same frame. All machines contribute to the current frame by sending and merging their image buffers.
2. Co-operating on different frames. In this case, each machine processes fully a frame that is part of an animation sequence.

In rendering, as in most network processing applications, the server-client scheme is used in order to distribute workload to several machines. In this scheme, one machine takes the role of splitting and distributing the work (the server) to other machines that receive these jobs, complete them and send back the results (the clients).

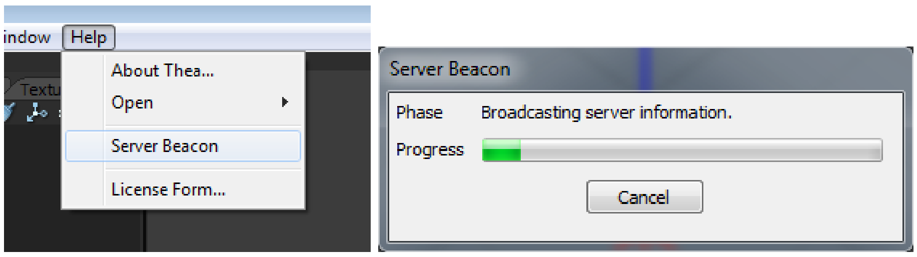
In Thea Render, there can be only one machine acting as server while there can be many client machines. The server machine can also act as a client, i.e. receiving and completing a render job itself. Because of this and the additional effort needed to communicate with client machines, the server machine is usually chosen to have high performance characteristics.

Network Rendering Setup

Let us see here how to setup the application in order to run in network rendering mode. As said in the previous paragraph, there can be many clients and one server, thus, we need to setup the application in more than one machines. Only a machine using the full application software license can run as a server. The machines that will be used as clients should have their corresponding client software licenses. Note here that if you have more than one full licenses, you can still choose one to become the server and the rest running in client mode[1].

The first thing generally done for network processing applications, is for client machines to look up and find the server machine. In order to facilitate this process, Thea Render comes with a search mechanism that will show all server machines. Nevertheless, this will only work if all machines are members of the same local area network and connected on the same router. Let us see how the server in such case can be located.

First of all, we need to run Thea Render on the server machine (in either the studio or darkroom mode). Then, from the help menu, we select the Server Beacon option. This will popup a small window indicating that the machine is broadcasting its identity (figure 1). There is nothing else needed to do on the server machine, so we keep the server broadcasting information and we move on now to client machines.

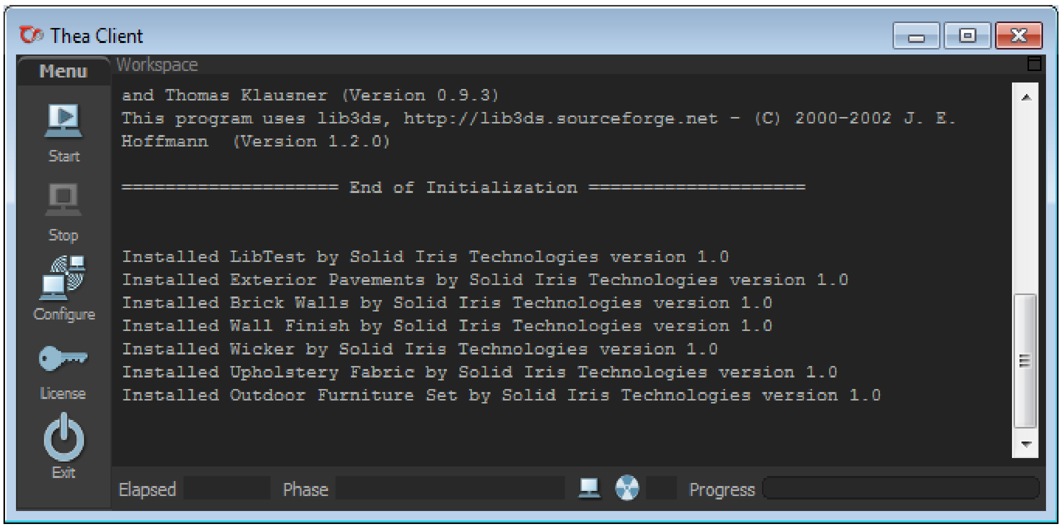


Phase Broadcasting server information.
Progress

[1] This is done by running Thea Render from command line, using the following syntax (without the quotes): "thea -client"

Figure 1. *Employing Server Beacon help tool.*

Running Thea Render on client machines, will automatically show the client user interface (figure 2). At this moment, the client is in start up mode and is not polling for any render jobs. We could click immediately the start button and make the client looking automatically for all servers broadcasting on the local area network. This way, assuming that your server and clients are on the same network, you can start immediately with network rendering without any further configuration.



```
and Thomas Klausner (Version 0.9.3)
This program uses lib3ds, http://lib3ds.sourceforge.net - (C) 2000-2002 J. E.
Hoffmann  (Version 1.2.0)

=================== End of Initialization ===================


Installed LibTest by Solid Iris Technologies version 1.0
Installed Exterior Pavements by Solid Iris Technologies version 1.0
Installed Brick Walls by Solid Iris Technologies version 1.0
Installed Wall Finish by Solid Iris Technologies version 1.0
Installed Wicker by Solid Iris Technologies version 1.0
Installed Upholstery Fabric by Solid Iris Technologies version 1.0
Installed Outdoor Furniture Set by Solid Iris Technologies version 1.0
```

Figure 2. *Running first time the client user interface.*

If on the other hand, you would like to select a particular server that your client should communicate with, you should click the Configure button. By clicking on that button, the client configuration menu pops up (figure 3). The server port is a number used to allow simultaneous communication of several applications between machines. As long as two applications do not use the same port number there is no risk of data conflict. It is recommended to use the default value (6200) or in (a rare) case of conflict with another application to try a higher value (but this will also need setting up the server beacon using that port). The overriding of threads is necessary in order to utilize certain number of threads in the client machine, not necessarily the same with the settings that come in a scene. The default value (0) corresponds to maximum efficient threads for the machine.

The upload period is the time period used to upload data to server (in minutes); for starting up, a low value may be used to verify everything is working properly but once you get accustomed to network rendering it is recommended to use a higher value (like 10-20 minutes) in order to minimize network traffic. Note that the upload period is only used by unbiased cooperative rendering, since in any other case, resulted data are uploaded as soon as the render job is finished.

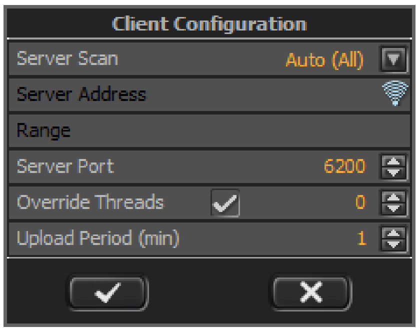
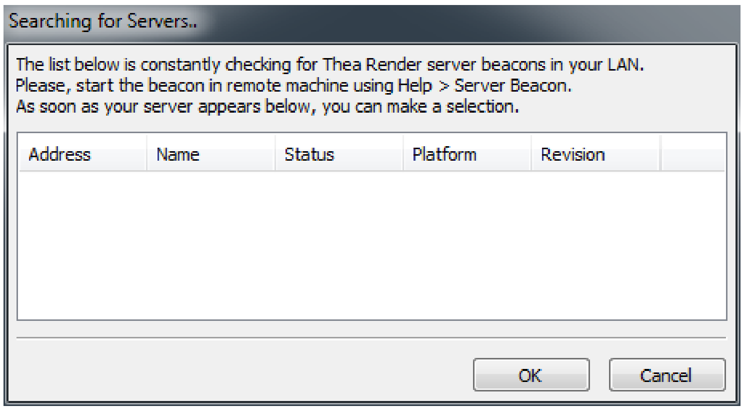


Figure 3. Client configuration menu.

We haven't discussed yet the Server Address entry - this is where the IPv4 (internet protocol version 4, for example 192.168.0.1) address should be entered. This address can be found from the network settings of the server machine but as we said before, we can locate the server easier. We already have the server machine broadcasting information at this point, so we can select "Manual" in the Server Scan entry and then click on the "sonar icon" found at the end of Server Address entry. This will bring up the server search list dialog (figure 4); this is where the application checks the local area network for server beacons[2].


Searching for Servers..

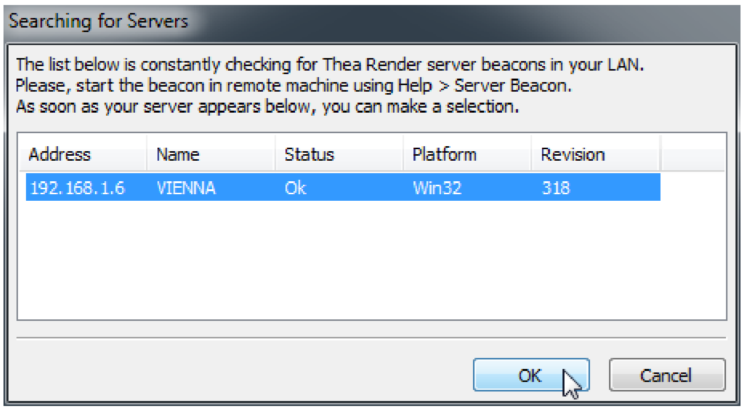
The list below is constantly checking for Thea Render server beacons in your LAN.
Please, start the beacon in remote machine using Help > Server Beacon.
As soon as your server appears below, you can make a selection.

Address	Name	Status	Platform	Revision

Figure 4. Searching for server beacons.

Once the server is found, it is displayed in the list, we can click on the corresponding line and then OK to select it (figure 5). It is important to note that automatic detection can take place only if the machines are connected on the same router. Also, installed firewall may not allow connection of the application, without even informing the user. In such case, the user should explicitly check the firewall, in case it forbids access[3].

[2] If there is a firewall installed then you may be inquired to allow network access for the application.
[3] For example, on Windows Vista OS, users should customize appropriately Bit Defender


Searching for Servers

The list below is constantly checking for Thea Render server beacons in your LAN.
Please, start the beacon in remote machine using Help > Server Beacon.
As soon as your server appears below, you can make a selection.

Address	Name	Status	Platform	Revision
192.168.1.6	VIENNA	Ok	Win32	318

Figure 5. Selecting a server from the search list.

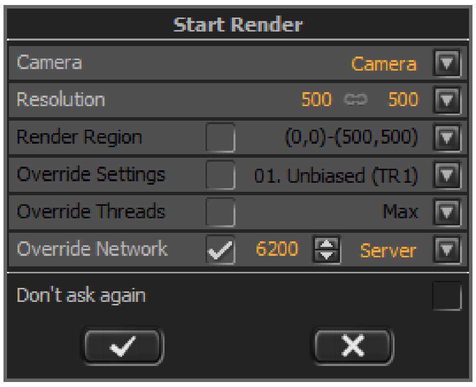
When the server is selected, we return to the client configuration dialog where the IP address has been setup and we can accept now the configuration by clicking the accept button. The application will remember our client configuration settings, so we can also exit the client application at this point.

Cooperative Rendering of Same Frame

As soon as all our clients have been setup, we can perform our first network render. The simpler network rendering mode is that of the same frame. This is typical for unbiased rendering, where many machines are working together contributing more and more passes on the same image.

There is no particular sequence in starting a network rendering, in the sense that server and clients can be started asynchronously and in any order. The most easy way to do this though, is to have the clients started and waiting for render jobs sent by the server. Even more, if you press the Start button on client interface, the application will automatically start next time the client runs. This is quite useful, since the whole procedure can be automated so that the application is launched once the operating system starts up.

Now, once the client machines have been started, they get essentially in a stand-by mode where they periodically check the server for any render jobs. As soon as the server starts running, the scene will be automatically sent to clients to be rendered. The easiest way to get a machine acting as a server is by overriding the network render settings, when the Start Render button has been pressed (figure 6).

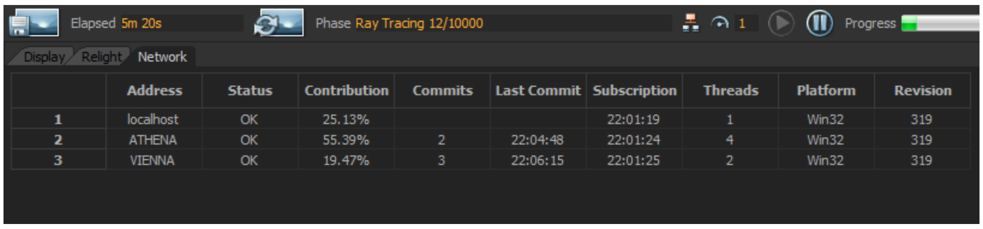

Start Render

Camera		Camera
Resolution		500 ⇔ 500
Render Region	☐	(0,0)-(500,500)
Override Settings	☐	01. Unbiased (TR1)
Override Threads	☐	Max
Override Network	✓	6200 Server

Don't ask again ☐

Figure 6. Overriding network settings to act as a server.

During rendering you can check the client status and progress in the network panel located inside darkroom interface (figure 7). This is purely for information and diagnostics of all machines working on rendering.



	Address	Status	Contribution	Commits	Last Commit	Subscription	Threads	Platform	Revision
1	localhost	OK	25.13%			22:01:19	1	Win32	319
2	ATHENA	OK	55.39%	2	22:04:48	22:01:24	4	Win32	319
3	VIENNA	OK	19.47%	3	22:06:15	22:01:25	2	Win32	319

Figure 7. The network panel info; a server and two clients in action.

Stopping a Network Render

With unbiased rendering, you can stop the render processing whenever the image looks good enough (other by stopping by maximum time or passes). When in network rendering, one can stop the whole render farm by simply clicking the stop button in server machine. This will send a termination signal to all clients which by their turn, will make their final render commit. At this point, several clients will try to connect to the server to transfer data, and this process may take some time to complete. After that, the clients return to stand-by mode for receiving new render jobs.

It is also possible to stop a client separately from the others. At the moment, this can be done only directly on the client machine though. In this case, it is recommended to press the Stop Client button in the menu bar, which will both stop rendering (and make the final commit to server) but also avoid setting the client into stand-by mode again (thus, new render jobs won't be handled).