A Client-Server System for Remote Internet Access to a Modular Visualization Environment – Abstract

In this project we present an approach to collaborative 3D Graphics processing and visualization through a Client – Server System that is used to access a Modular Visualization Environment (MVE) remotely through the Internet. The purpose of this study is to provide an efficient protocol for the communication between the client and the server, and between the server and the environment.

The Visualization Environment, which is the back-end of our system, is a modular environment. This means that its operation is based on a set of modules, which are individual components, and are responsible for various functions that are applicable to a representation of a 3D graphics object. Such functions can be:

(1) loading an object from a file or a stream,

(2) modifying the object,

(3) saving the object to a file, and

(4) rendering the resulting object.

These modules are designed in a common form, and each module has a set of parameters, inputs and outputs. The parameters define the behavior of the module, and they depend on the task the module is responsible for. The inputs of the module are arguments that are necessary for the module to run. An input to a module is usually an object that resulted of another module's execution. The outputs of a module are the resulting objects that the module has produced with its execution. Several modules connected properly with each other can form a scheme, which may perform complex computations to the objects it processes. This scheme can be executed, or it can be saved so that we can carry on modifying it in the future.

The server, which acts as a middleware between the MVE and the client, is responsible for providing the client with the data required to operate and offer its services to the remote user. The server acquires the information it needs from the MVE.

The client is the front-end of the system, and it is the part of the system, which the end user gets to interact with. The client downloads the list of available modules on the MVE, by making a proper request to the server. Then the user can develop his scheme by inserting modules into it, setting their parameters and connecting them with each other.

When the user has created a complete scheme, he may execute it and get the results. If the results contain objects for visualization, the client loads these objects and renders them. The rest of the results, which may be objects that are to be saved in a file, are kept on the server's computer, since the client does not need to posses this data. Instead of executing his scheme, the user may also save it for further processing. The scheme is saved on the server's computer, so that the user may retrieve it the next time he connects to the server, and more important, so that other users may use this scheme and maybe modify it. The client offers this feature, of saving and loading schemes, and it is achieved by making proper requests to the server. The MVE is not involved at this point.

The protocol for the communication between the client and the server, and the server and the MVE, defines the form of the data that is transmitted, the content of the data, and the moments when data needs to be transmitted. The communication is ordered as follows:

- When the client connects it requests the list of currently available modules
- The server calls MVE to get the list and sends it to the client
- When the client has created a scheme he can send it to the server for execution
- The server calls MVE to execute the scheme, and sends the results to the client
- Instead of executing the scheme, the client may send it to the server for saving
- The client may also load a scheme. It requests the list of available schemes and when the server sends this list it chooses a scheme. The server sends this scheme.

The data is always in XML format and contains the absolute information needed for the whole system to operate. Some of the parameters of the modules in the MVE have been omitted, and default values have been set in their place, so that we can reduce the complexity and the amount of the data to be transmitted.

The performance of our system has been more than satisfying. The time it takes to complete the described tasks was satisfactory, and the status of the network or the computers involved does not seem to influence the overall performance. The system performed well, even when the network was loaded with traffic, or the related computers are low on resources. In conclusion, we have designed and implemented a system that provides remote Internet access to the MVE, with the minimum communication possible, that performs well, in various states of the involving resources.