Controller:

```
Create socket connection with Server/Renderer
//Request file list from Server

while(!exit){
     Request File list
     Display file list

     Get user input

     if(user input != exit) {
          Request specified file from Renderer
          Listen to renderer
          if(rendering beginning)
               fork()
               Child:
                    Displays user controls
                    Get user input
                    Switch case(pause)
                              Send pause signal
                         case(resume)
                              Send resume signal
                         case(restart)
                              Send restart signal
               Parent:
                    while(!renderingDone){
                         Listen to socket
                         if(rendering complete signal received)
                              Rendering done = true
                    }
                    Kills the child
     }
}
Send exit signal to Renderer
Send exit signal to Server
```

Server:

```
Create socket connection with Controller/Renderer
while (!exit){
     Message = parsemessage()
     switch(messageType)
          case:10
               Get file list
               Format message
               Send message to controller
          case:20
               fork()
               Send requested file in child fork
               Send 21 message  to renderer
          case:30
               Send pause message to child
               Send 31 message to renderer
          case:32
               Send resume message to child
               Send 33 message to renderer
          case:34
               Kill child
               fork()
               Send requested file in child fork
               Send 35 message to renderer
          case:99
               Kill child
               exit = true
     Default:
          //TODO: determine size of message and transport
          protocol
}
```

Renderer:

```
Create socket connection with Server/Controller
while(!exit){
     Read incoming message
     Switch // message type
          case(request file to render)
               Request file from server
               Send rendering begins to controller
               While(!done){
                    Receive portion from server
                    Print portion
                    Check for controller signals

                    while(pause){
                         Receive controller signals
                         if(resume)
                              Pause = false
                         if(restart)
                              break;
                    }
                         if(restart)
                              Empty socket
                              Request file from server
                    if(Q value == 0)
                         done = true
               }
               Send rendering complete to controller
          case(exit)
               Exit = true
}
```

Renderer:

```
Create socket connection with Server/Controller
while(!exit){
     fork()
     Parent:
          Read incoming message from controller
          Switch // message type
               case(request file to render)
                    Request file from server
                    Send rendering begins to controller
               case(pause)
                    Request to pause
               case(resume)
                    Request to resume
               case(restart)
                    Request to restart
                    Empty socket
     Child:
          Read incoming message from Server
          while(!done)
               Receive portion from server
               Print portion
               If end of file
                    Send rendering complete to controller
                    exit

}
```