Apáczai Csere János High School

Mathematics - Informatics - Intensive English Class

System Share

Made by: Supervising Teacher:

Lippay Balazs Geváld Júlia

2019, Cluj-Napoca

User Documentation

General description

System Share is quality of life application that runs in the background and improves significantly the ability of the user to use multiple computers at the same time. It speeds up the work of the user by removing the transition phase between multiple pairs of keyboards and mice, and making the action of transferring focus to another computer seamless by connecting them on the interface level and creating a shared virtual desktop, through which they function almost completely as one single machine.

The perks of this fused virtual machine are simple: the ability to use a single mouse and keyboard, and transition between computers as if they were just normal connected display devices; and share a common clipboard amongst them.

System Share needs a simple one time setup on each computer, then starts working in the background without obscuring the work of the user in a seamless way to the effect that the user can even forget about installing it in the first place, and on the occasion of a work environment change, the settings can be adjusted through a simple interface immediately.

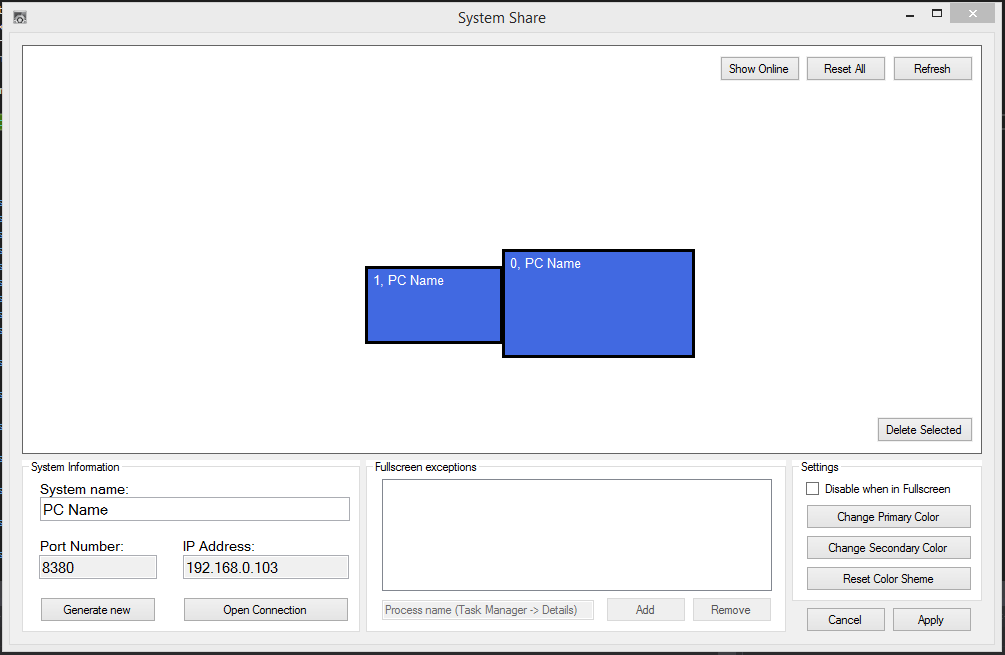
System Share was written in C# using Microsoft Visual Studio as IDE, an editor that supports almost anything that a programmer could come up with, and helps the programmer out with automatic code generation and .Net documentation pop-ups for every library and premade function. System Share was tested on windows 7 through 10 and should work on lower versions as well. For the specs of the system, anything should be fine as long as the computer can run smoothly by default.

Regarding the visual aspect of the program, it is recommended to not use any display with a resolution lower than 720p or HD, since the interface window might not fit on the screen, if something like this might happen, the user can manually adjust the settings outside the program in the %appdata% with the help of the data structure explanation in the upcoming *programmer documentation.*

.

Fist time setup and interface navigation

System Share is a twin application that relies on both sides to function, thus the first action is to setup the brains of the program, the HOST machine. A main computer should be chosen, one that the user defaults to using, one that will have the main mouse and keyboard connected.

On the chosen computer, the user should run the HOST installation application and install the program, this action will also generate desktop and start menu shortcuts, as well as adding one to the startup folder of the user. By starting the application, the interface windows will open:

On the interface window, the user can give a name to the machine or leave it on default, then press APPLY then OPEN CONNECTION and done. Optional settings can be made in addition to the previous one: the user can generate a new port number, which will always pick an unused port, or change the color of the rectangles representing the displays.

By checking the *Disable when in Fullscreen* box, the program will stop to function if another application is in the FULLSCREEN setting, preventing unwanted transitions while for example playing games. The user can add exceptions to this rule by typing in the PROCESS NAME of the program into the textbox under the list, then pressing the *Add* button, unwanted additions can be removed with the *Remove* button.

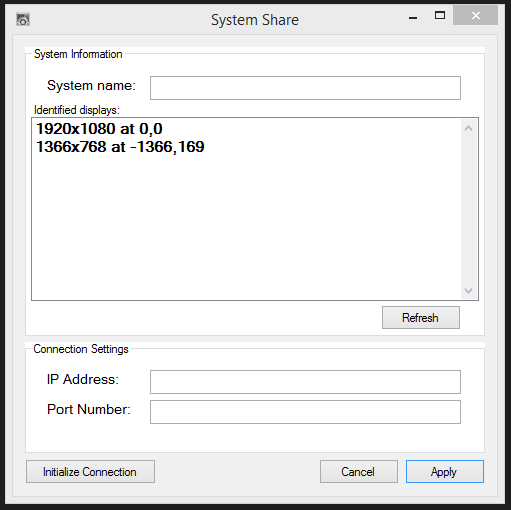
The remaining buttons will be explained after the pairing process, for which we need to set up the CLIENT machines. There can be multiple CLIENTs but only one HOST.

To set up a CLIENT, run the CLIENT installation, which will do the same as for the HOST installer, but install the CLIENT application. On starting the installed application, the interface is going to pop-up:

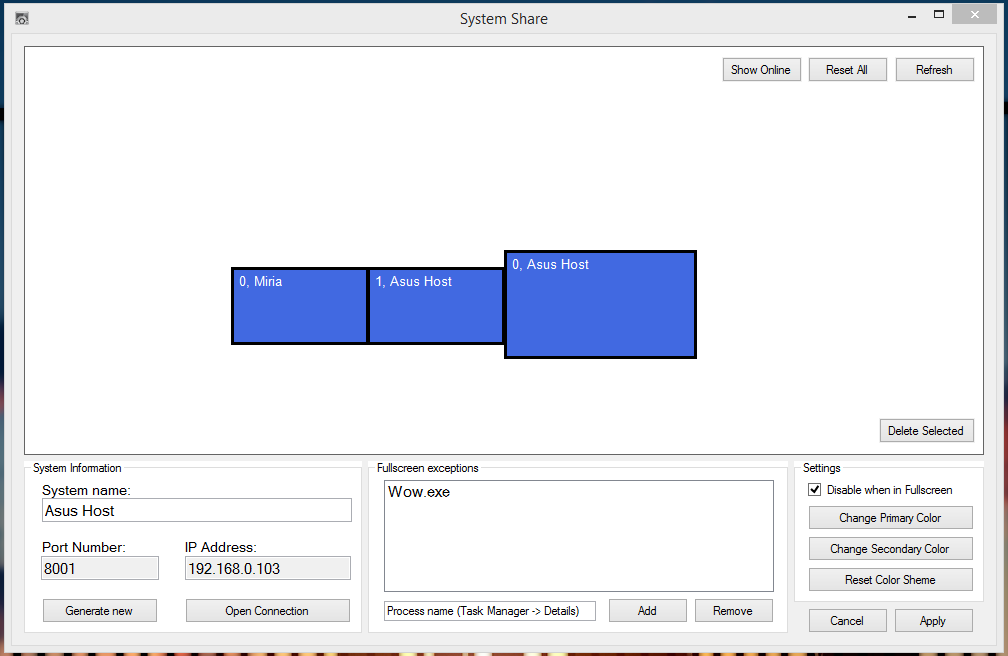
In this window, a name needs to be given to the machine. The IP address and PORT numbers are found on the HOST machine under the specific labels.

After the data has been inserted, the APPLY then the INITIALIZE CONNECTION buttons need to be pressed and the setup will be done for the CLIENT.

In case of environment change, the *Refresh* button needs to be pressed and then APPLY.



Back on the HOST machine, the CLIENT should have appeared as an extra rectangle or multiple ones, these rectangles represent the DISPLAYs of the specific computer, by any chance this did not happen, the *Refresh* button can be pressed to refresh the interface, if this did not fix the issue, the firewall will be blocking the network, contact the network manager of the institute in this situation. The rectangles can be freely dragged around with the mouse and oriented as pleased. If the orientation is changed, the APPLY button will save everything, and the CONNECTION will need to be reopened.



In the middle of the interface, the rectangles representing the DISPLAYs show on their upper left corner the index of the DISPLAY based on the machine of origination, followed by the name of the machine that the DISPLAY is associated with.

By clicking on any rectangle, the DISPLAYs of the associated machine will get selected, these can be deleted by pressing the *Delete Selected* button, this removes them from the database, and the machine will need to resend the data once connected, this is a way to remove unused machines, or reset the data of a machine in case of an environment change.

The *Reset All* button should be pressed to delete all data regarding the DISPLAYs and update the DISPLAY settings of the HOST computer.

The user can check the current alignment by pressing the *Show Online* button, which will open a new window displaying the orientation of the online DISPLAYs. The main display of the main computer will be highlighted as a reference point.

Usage of the application

System Share takes pride in the simple, smooth, and seamless experience that it provides, after the first time setup is completed, the whole program can be forgotten about.

System Share starts together with the user, on login the application is run, and the networking is done automatically without any need for the user. When a CLIENT is started up, it gets logged in, then the online DISPLAYs get rearranged automatically to fit the new parameters, and the whole setup will work as if it was one single computer, the user can drag the mouse from one computer to the next following the preset DISPLAY alignment as if the desktop would be extended to them; use the keyboard on any of the online computers following the default windows focus system: the text gets written to where the user clicked last.

The COPY and PASTE commands work as follows: the shortcut key combinations (CTRL + C / CTRL + V / CTRL + X) share a clipboard, and send the TEXT data through the network to other connected machines, and the RIGHT CLICK copy/ paste/ cut work without sending the data, and they allow the copying of non-text data onto the same computer.

In case of multiple DISPLAYs connected to the HOST machine, in case of the alignment differing from the settings set up in the original WINDOWS DISPLAY SETTINGS, the application will overwrite them, but for the smoothest experience, it is recommended to match them, since the application is purely logic based, not driver based, and the mouse will flicker if it can move on the original WONDOWS SETTINGS but not on the APPLICATION SETTINGS, the movement is impossible to be prevented, thus the mouse slightly appears for a millisecond on other screens before getting traced back to the correct position.

For easier navigation, the application can be accessed from the system tray, where double clicking the icon brings up the interface, and right clicking on the icon erects a menu in which the user can quit the application, check the connection status, open or initialize connection, or open the interface.

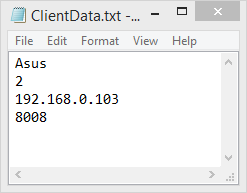
Programmer Documentation

Data Structures

Since System Share stores DATA about the user, it has different data tables and data structures. The data stored by the application is not PERSONAL DATA, it is the DISPLAY data and the application settings, thus it does not need protection, since the DISPLAY data is accessible by anyone, and the application does not care if the settings are copied or changed by the user manually.

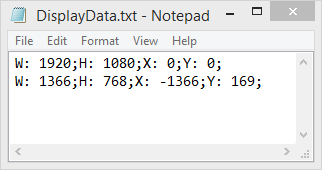
The DATA is stored in plane .TXT files in the roaming folder of the user, under a FOLDER with its name starting with “SystemShare”, this folder can be accessed by opening a file explorer and typing into the address box "%appdata%".

Client specific data: CientData.txt





Shared data structure: Client: DisplayData.txt - Host: LocalDisplayData.txt

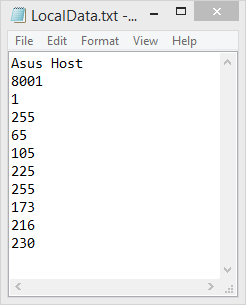


Each row represents a different DISPLAY device, each snippet of data starts with a letter followed by a colon and an empty space, and ends with a semicolon to signal the end of the snippet.

The Data Structure is as follows:

W - width, H - height, X and Y - coordinates relative to the main DISPLAY, which is (0,0)

Host specific data: LocalData.txt, DisplayData.txt, FullScreenEvasion.txt



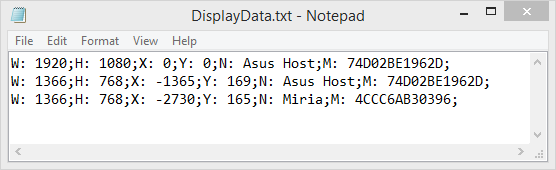
The given name of the HOST

Port Number on which to accept clients

Check-Box value (1- checked, 0- not checked)

Primary color ARGB

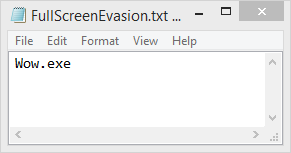
Secondary color ARGB



Each row represents a different DISPLAY device, each snippet of data starts with a letter followed by a colon and an empty space, and ends with a semicolon to signal the end of the snippet.

The Data Structure is as follows:

W - width, H - height, X and Y - coordinates relative to the main DISPLAY, which is (0,0)  
N - name of the computer, M - MAC address of the computer (unique code given by manufacturer)



Each row represents the name of the application that the program ignores when *Disable when in Fullscreen* is on.

Communication between computers

Before we begin with the logic of our application, we need to define a language through which the computers can communicate with each other. The language follows strict rules: the first letter defines the command, and the last character is a semicolon, which marks the end of the command. The CLIENTs and the HOST have different commands that they can accept, but the structure is mostly the same.

Commands towards the CLIENT:

|  |  |  |
| --- | --- | --- |
| **First letter** | **Description** | **Example** |
| m | mouse move on screen S, to(X,Y) | m0,500x500; |
| l | Left click (u-up, d-down) | lu; / ld; |
| r | Right click (u-up, d-down) | ru; / rd; |
| w | Middle click (u-up, d-down) | wu; / wd; |
| s | Scroll x number of lines | s500; / s-500; |
| k | Key (u-up, d-down), Key-Code (ASCII) | ku12; / kd12; |
| d | Data request | d; |
| q | Quit | q; |
| j | Login request | J; |
| p | Clipboard paste | p<13(length of text)>Text to paste; |
| c | Send clipboard to host | c; |
| v | Empty command, used for data flow | v; |

Commands towards the HOST:

|  |  |  |
| --- | --- | --- |
| **First letter** | **Description** | **Example** |
| l | Login Request + MAC address | l74D02BE1962D; |
| n | Name Change | n,ASUS; |
| d | DISPLAY data, Width, Height | d,1920x1080; |
| u | Data transfer end | u; |
| p | Clipboard paste | p<13(length of text)>Text to paste; |
| c | Keep up data flow, sends empty command | c; |
| ! | Empty command, signals success | !; |

Undefined commands are automatically disposed of.

The communication between the computers is done in a MASTER - SLAVE relationship: the HOST sends a command to the CLIENT, the CLIENT fulfills the command and answers with the results, which are either a "JOB DONE" (which is an empty command - !; -) or a request for data/ action (login, name change, clipboard update, etc.).

Logic behind the application

Every time the application starts, it reads the data from the .TXT files stored in %appdata%, if it does not find them or something goes unexpectedly wrong, the application opens the interface, alerting the user to insert the missing data into the interface.

If everything went well during the starting period, the application starts looking on the network for possible connections. If one is found, it does the login / register procedure.The logic for the two applications differs from this point on:

**Client Logic:**

Once the login phase is over, the client goes idle while observing the established network stream, if it finds data on it, then it starts working again, by reading the available data, processing it and returning the result. Afterwards the application enters the observing phase again.

**Host Logic:**

To understand the logic, we need to define the expression VIRTUAL MODE, which will be used in the future, and it refers to the Mouse being on a REAL (HOST) screen, or a VIRTUAL (CLIENT) screen.

OUTSIDE VIRTUAL MODE, the mouse position is updated relative to the screen it is on and we see the REAL mouse moving on the screen.

INSIDE VIRTUAL MODE, the mouse is hidden, and placed in the center of the main screen, the position of the VIRTUAL mouse is updated by calculating the distance traveled by the REAL mouse from the center of the screen, and adding the value to the VIRTUAL mouse. After every update, the REAL mouse is put back in the center.

By using these two modes, we can achieve smooth and true mouse movement in and outside our HOST machine.

Before our Logic starts working, we need to provide a solid background from our data about where each DISPLAY is and their dimensions, and we need to make sure to align each of them to be next to each other in case of an "in between" display being offline. The alignment is done using analytic geometry, and the end result is that each screen gets situated next to each other with ONE PIXEL OVERLAPPING.

After the alignment is done, the logic can start. The logic consists of again two modes, which are relative to the before mentioned ones:

OUTSIDE VIRTUAL MODE, the logic checks if the mode should be changed, by looking if the mouse is inside one of the virtual displays, which overlap the sides of the current one, and checking if the mouse is inside the bounds of the VIRTUAL DESKTOP, if not, it return the mouse to its last valid position.

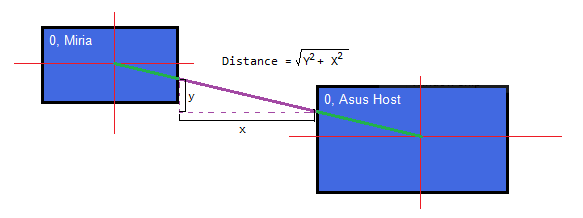
INSIDE VIRTUAL MODE, the logic checks if the mode should be changed, by looking if the VIRTUAL mouse is inside one of the REAL displays, and monitors the actions of the user by blocking them on the HOST computer and sending them to the CLIENT whose VIRTUAL screen the VIRTUAL mouse in on.

Background of the Logic / Analytic Geometry tools

CheckCollided(parameters) - checks if the given rectangle collided with any other rectangle.

Poligon(parameter) - a recursive function that arranges all colliding rectangles into polygons.

CheckDist(parameters) - aligns the given rectangle to the nearest other rectangle that belongs to a different polygon based on calculated distance. The alignment is done by walking along the line between them until they collide.



CheckDistAll() - iterates through all polygons and calls "CheckDist" on them.

ResetOnline() - resets the positions of the online DISPLAYs, used when a new computer logs in.

CheckCollided(parameter) - used when drag and dropping with mouse on the interface, makes sure the rectangles are not on top of each other.

DragAll() - used when the user clicks on an empty part of the interface, dragging every rectangle displayed on the screen.

Tools of the Logic

VMouse - a class that controls the mouse, that can set or get the position, click and scroll the mouse.

VKeyBoard - a class that control the keyboard, can press and release keys.

ClipBoard - a class that can access the clipboard and get or set it.

KeyBoardHook - a class that observes the keyboard strokes and prevents them from reaching the operating system, also sends key data to the main server.

MouseHook - detects when the user presses the left click.

FullScreenCheck - the brain behind the *Disable when in Fullscreen* setting, it checks if the foreground application is as big as the screen, then if it is, it check if the process name of that application is not System Share or is on the exception list.

VMousePosition - a class that updates the position of the VIRTUAL mouse based on the above mentioned mode the application is on.

Connection - present on both machines, the class that handles the network and the logic, the BRAIN.

MainForm - on the HOST machine, serves as the sender of the keyboard and mouse data from the machine to the server, a Fullscreen window that has a trigger to let analog data through or block it and send it.

Extra information

The differentiation between the CLIENTs is done with the unique MAC address.

The mouse blocking is done by setting the MainForm window opacity to 1% and changing between a transparent background color and a solid one, then handling the mouse-click events.

The opening of the application multiple times or opening both of them on the same machine is heavily ILL ADVISED.

In case of a fatal error, pressing CTRL+ALT+DEL frees the mouse from the center of the screen.

If any window that has administrator privileges gets focused, the application will lose the permission to control the mouse.

Possible future features

* Sharing sound from clients to host.
* Automatic drive sharing
* File transfer with copy/ paste
* Remote controll of computers with no displays using a virtual one
* The ability to drag open windows from one computer to the other on the screen
* Automatic printer sharing
* Linux/ Mac full ports
* Android/ IOS client ports

Table of Contents

[User Documentation 1](#_Toc6317816)

[General description 1](#_Toc6317817)

[Fist time setup and interface navigation 2](#_Toc6317818)

[Usage of the application 5](#_Toc6317819)

[Programmer Documentation 6](#_Toc6317820)

[Data Structures 6](#_Toc6317821)

[Communication between computers 8](#_Toc6317822)

[Logic behind the application 9](#_Toc6317823)

[Background of the Logic / Analytic Geometry tools 10](#_Toc6317824)

[Tools of the Logic 11](#_Toc6317825)

[Extra information 12](#_Toc6317826)

[Possible future features 12](#_Toc6317827)