Javascript/WebSocket Based Desktop Share

## Implementation Details

|  |  |  |
| --- | --- | --- |
| **Date** | **Author** | **Modification** |
| 8/29/2012 | Benjamin Xiao | Initial Writeup |

### Goals

The goal of this project is to create a purely web based desktop share client that requires no download on the part of the user. The client side will be using Websocket to establish a connection to the desktop share stream, Broadway JS for H.264 decoding inside Javascript, WebGL for fast YUV->RGB conversion, and Canvas for rendering video data to the browser. The server side consists of the video-based desktop share prototype ASRDemo.exe, modified to stream video data through a C++ Websocket server.

### Client Side

**WSAvcPlayer Flowchart**

Decode with Broadway JS

Render image data using Canvas

YUV->RGB Conversion

Fast Path:

WebGL Shaders

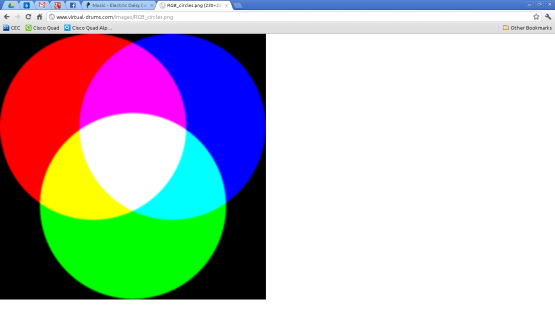
Slow Path:

Javascript algorithm

Connection to server

On NAL received

010101011001



The Javascript client code is located in wsavc.js. To instantiate the client, you need to create an object of WSAvcPlayer, which handles creating a Websocket connection, passing stream data to the Broadway JS decoder and rendering to HTML5 Canvas.

|  |  |
| --- | --- |
| Functions | Detail |
| WSAvcPlayer(*canvas, canvastype, numnals, delay*) | *canvas* is any HTML5 canvas object. *canvastype* specifies which path we use for YUV->RGB conversion and accepts “webgl” or “canvas” as parameters. |
| connect(*url)* | Connects WSAvcPlayer to a Websocket server. URL should be in the format “ws://ip:port/resource” |
| disconnect() | Disconnects from Websocket |
| decode(*data*) | Passes video data ArrayBuffer to the decoder. |
| onPictureDecodedWebGL(buffer, width, height) | YUV->RGB conversion and render function using WebGL shaders. Fast, but needs hardware and browser support. |
| onPictureDecodedCanvas(buffer, width, height) | YUV->RGB conversion and render function using a slower Javascript algorithm. |
| playStream()  Not used for ASRDemo, only Webclient Demo | Sends a REQUESTSTREAM message to Websocket server. For example if numnals is set to 1 and delay is set to 100, the message sent to server will be “REQUESTSTREAM 1NAL 100MS” |
| stopStream()  Not used for ASRDemo, only Webclient Demo | Sends a STOPSTREAM message to server. Server will stop sending data after it has received this message. |
| playNAL()  Not used for ASRDemo, only Webclient Demo | Sends a REQUEST message to server requesting one chunk. For example, if numnals is set to 1, the message sent to server will be “REQUEST 1NAL” |
| flush()  Not used for ASRDemo, only Webclient Demo | Resends last frame received from websocket to the decoder. |

Please refer to webclient\_as.html for more information on how to use WSAvcPlayer

### Server Side

The Websocket server code is located in the WSStreamer visual studio project. The server supports client connections to specific known resources and messages are sent to all clients connected to a resource. For example, if you add a resource “/as” to the server, clients can now connect to the server using “ws://ip:port/as”. To instantiate the server, you need to create an object of WSStreamer, add a hostname/ip and resource to the server, and initiate the run loop. The hostname/ip is necessary for origin checking, otherwise the server will only accept connections from localhost.

|  |  |
| --- | --- |
| Functions | Detail |
| WSStreamer(string port) | *Port* is the port number that the server will listen on |
| run() | Starts server loop in a separate thread so it does not block. |
| runAndBlock() | Starts server loop in main thread. |
| addResource() | Adds a resource that clients can connect to. |
| removeResource() | Removes a resource |
| addHostname(string host) | *Host* is the hostname or ip address that you want to add to the server. Whenever a client connects to the server using a specific hostname, it will be verified against a list of added hostnames on the server. If the hostname is not in the list, the connection will be rejected. |
| sendAll(string resource, const vector<unsignedchar>& data) | Sends a vector of binary data to *resource* |
| sendAll(string resource, const unsigned char\* data, int len) | Sends an array of binary data of length *len* to *resource* |

Please refer to ASRDemo.exe code, specifically WseVideoSourceChannel.h, for more information on how to use the server. Relevant code is surrounded by #ifdef ENABLE\_WEBSOCKET.