

Programming
Multimedia
Groupware with
Collaborary:
GroupLab Networking

Topics

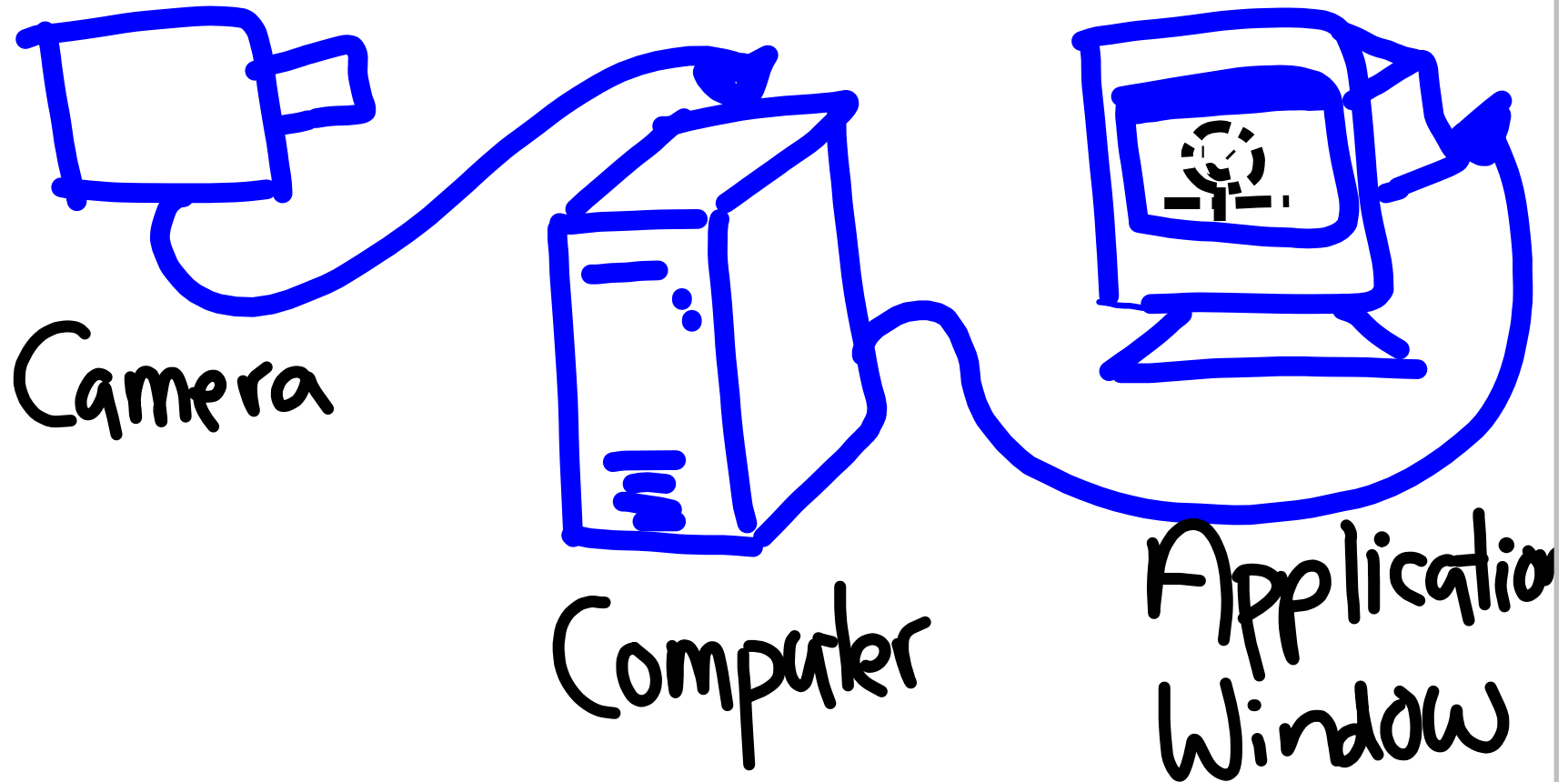
- Multimedia Capture & Render w/ Collaborary
- Distributed groupware with GroupLab. Network..
- Put the two together

Demo 1: Video Mirror

Shows:

- Getting Collabrari
- Working with C#
- Simple video capture
- Simple video rendering

Simple Video Mirror



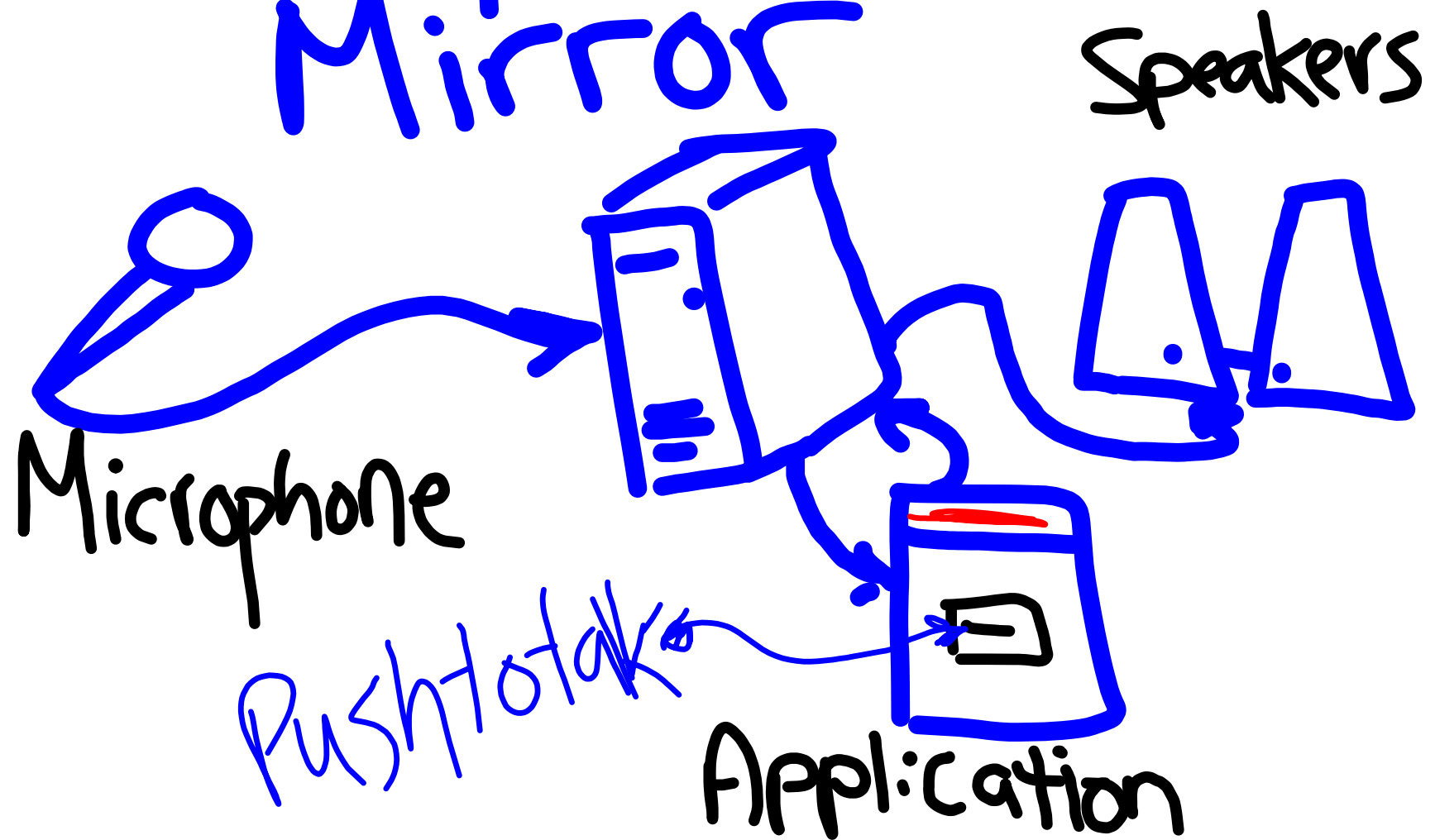


Step 1:

Download /install

<http://grouplab.cpsc.ucalgary.ca/collabrary>

Simple Audio Mirror



Demo 3

Shared Inkpad

- Demonstrates
 - Getting Started w/ GraphLab.
- Networking
- Shared Dictionary Concepts

Shared Dictionary

"Dictionary" == "hashtable"
"map"
"associative array"

String keys → values of any type

↳ look like paths in
a file system

e.g. ↓

```
d["/users/boylem/name"] = "Mike";  
d['/users/boylem/mass'] = 87.3;  
d["/users/boylem/picture"] = camera.Frame;
```

Publish/Subscribe

Producer/Consumer

Model/View

Abstraction/Presentation

Bottom Line

put data in → get notified of
new data → update GUI

Demo Architecture



Mouse Down
Mouse Move
Mouse Up

} → store in sd

Subscription. Notified → Paint

Data Model

/strokes : Vector of Guid
Stroke ids

/strokes / id / pts : vector of
Point

Could extend w/ colour,
thickness, etc.

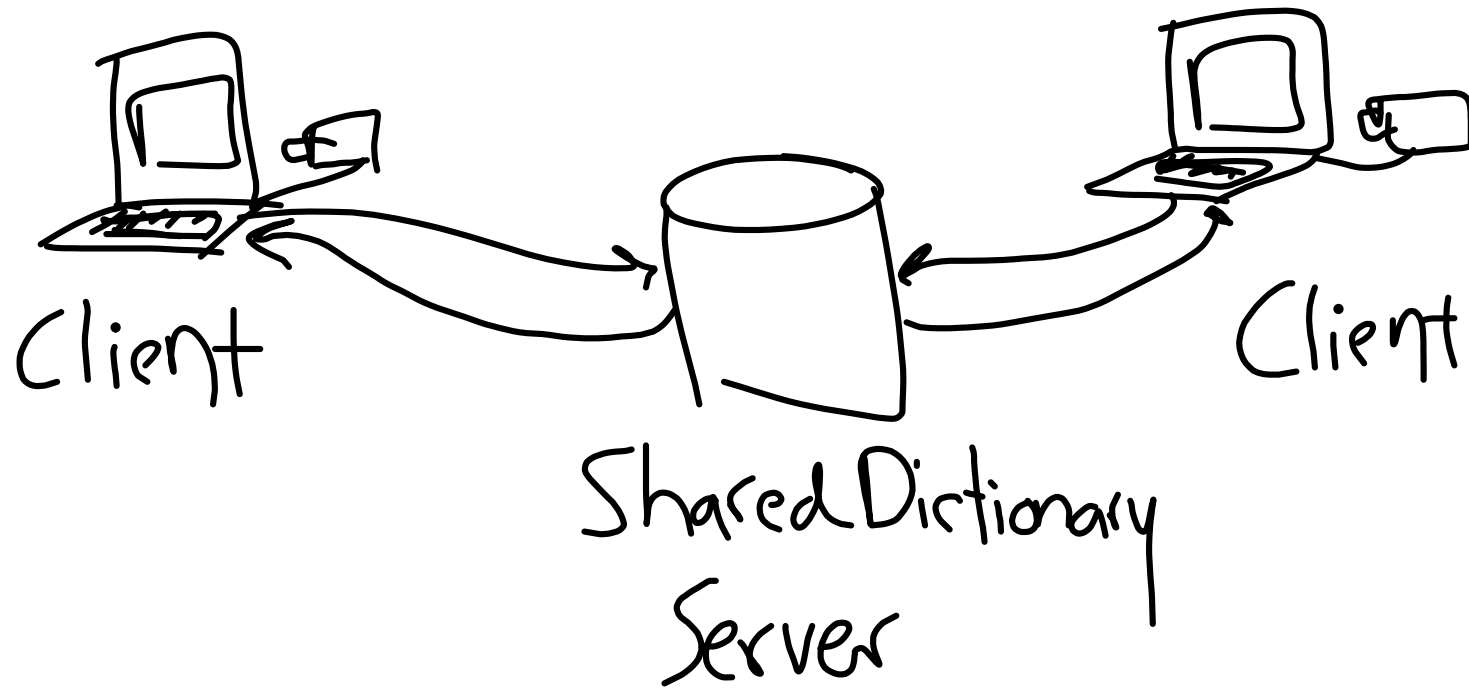
Get GroupLab Networking

[http://grouplab.cpsc.ucalgary.ca/
software/networking](http://grouplab.cpsc.ucalgary.ca/software/networking)

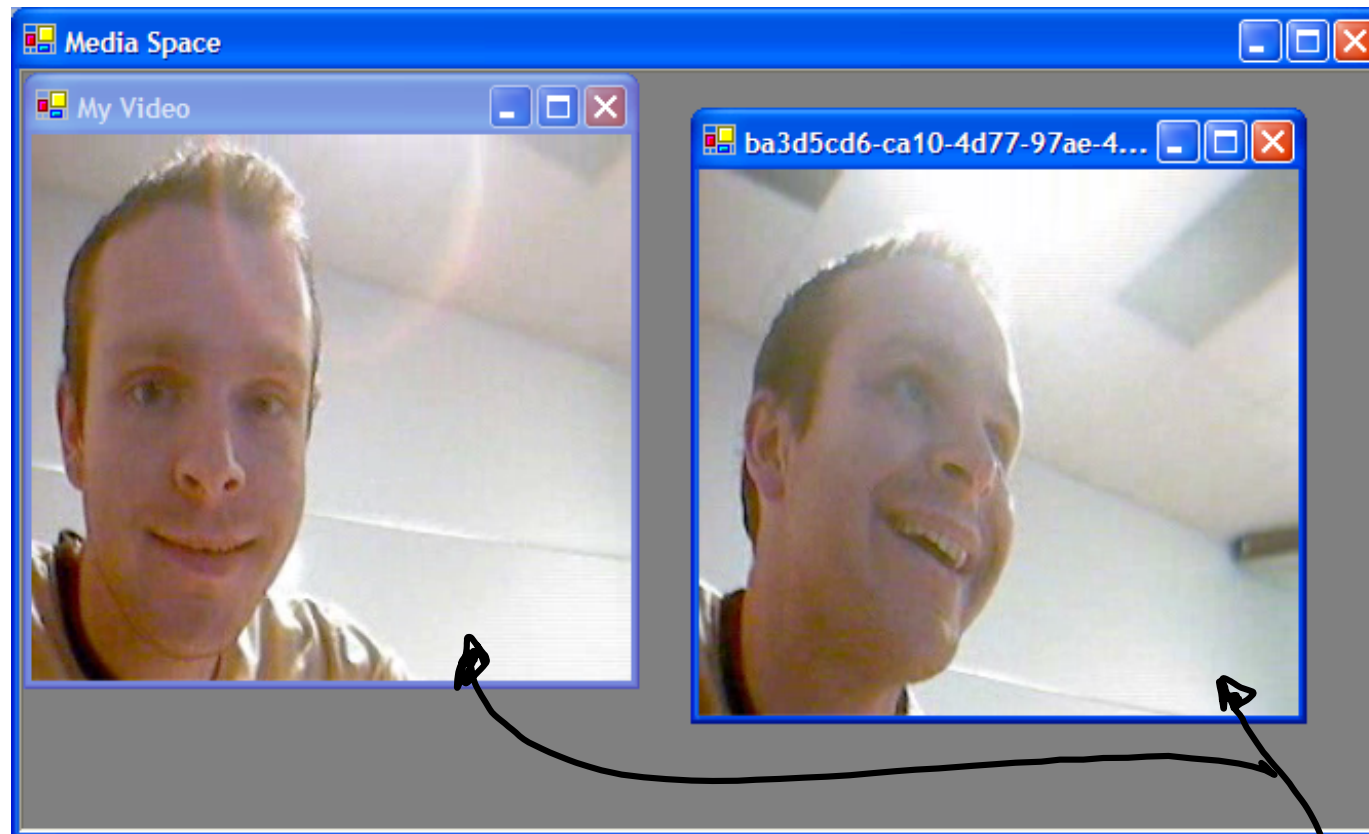
Demo 4:

Media Space

Media Space Architecture



Media Space UI



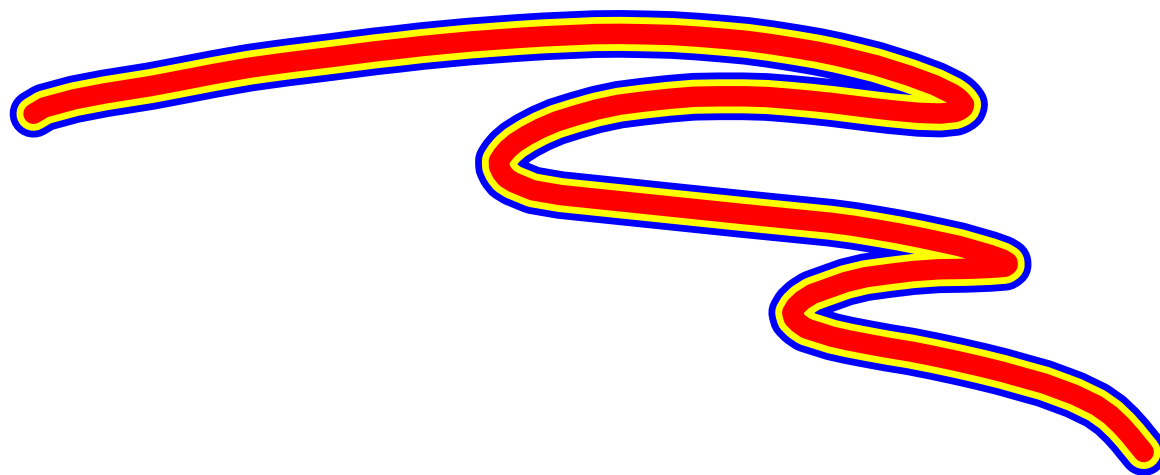
Main Window

VideoWindow

Dictionary Structure

/id → .transient := id
→ /info := Map
 "id" = id
 "name" = "Michael Boyle"
→ /video : Buffer
 MPEG-4 320x240 @ 1fps

Digression



Collabrary Photo Object

Important Methods

Resize

p.Resize(int width,
int height, bool smooth)

pretty straightforward

Copy : returns a new
Photo that is a copy
of a portion of the
original

Object p.Copy(int x, int y,
int width, int height,
PhotoCopyStyle style)

Photo p2 = p.Copy(0, 0, 0, 0,
PhotoCopyStyle.Pixels) as
Photo ;

(Zeros for width, height mean
everything; in COM there
is no overloading, but there
are default param values)

Cast the result as Photo
because actual return type
is just object

C# as operator

- Casting operator
- only works with reference types (classes) not value types (structs)

```
string s = "hi";  
Point p = new Point(2, 4);
```

```
object o = s;  
string s2 = o as string;  
string s3 = (string) o;
```

```
o = p;
```

```
Point p2 = o as Point;  
Point p3 = (Point) o;
```

What's the difference?

```
string s2 = o as string;
```

- FAST
- returns null if o not string

```
string s3 = (string) s;
```

- SLOW
- throws exception if o is not string

Photo.Paste(
Photo p,
int x, int y,
float alpha);

p1:

(200,100)



p2:



p1.Paste(p2, 200, 100, 0.5f);

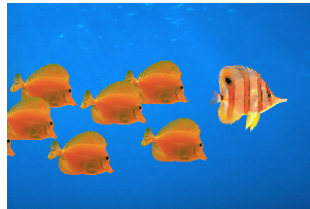
p1:

(200,100)



```
Photo.PasteEx(  
  Photo p, int x, int y,  
  Photo mask);
```

p1:



p2:



mask:



```
p1.PasteEx(p2, 0, 0, mask);
```

p1:



Note
alpha
blending

Note only
blue
channel pasted

photo.Distort (

float clarity,

PhotoDistortStyle style)

clarity: 0.0 ~> 1.0

↑
extreme
distortion

←
no
distortion

Photo.Load (string filename)

Photo.Store (string filename)

fairly straightforward

supported file types:

bmp

jpeg

png

24-bit colour

no per-pixel alpha

photo.Grab(object src):

src:

System.Reflection.Missing.Value

- Grab whole screen (primary display only)
- automatically resizes to fit

Control or Control.IWin32Window.Handle

↳ could also be Form

- Screen grab given control, form or window

string url

- launches IE in the background to capture bitmap of web page
- if size initially 0x0, automatically resizes to about 800 px wide, then how ever tall needed to fit whole document
- if size not 0x0 px initially, layout doc in window of same size and grab just that part
- ActiveX not allowed
 - no SWF, PDF, or WMV, MOV, etc.
- Can take a while to load IE, browse/download page → use same photo obj
- DHTML sometimes inconsistent

Photo.LockMemory(
 out IntPtr p,
 out int len):
 Photo.UnlockMemory(...):

Format: standard windows 24bpp DIB

3 bytes per pixel B, G, R

left-to-right, bottom to top

↳ first byte is B of bottom-left pixel (0, H-1)

• 4th byte is B of (1, H-1)

• last byte is R of (w-1, 0)

• Rows aligned on 4 byte boundaries. 0, 1, 2 or 3 pad bytes may be inserted between rows
 e.g.: 21 pixel wide image

row = 21 × 3 = 66 bytes data

each row has 2 additional pad bytes for alignment

byte 0: 63 64 65 66, 67
 row H-1 [B, G, R, ..., B, G, R, 0, 0.
 col 0 col 20 pad

byte 68:
 row H-2 [B, G, R, ..., B, G, R, 0, 0,
 col 0