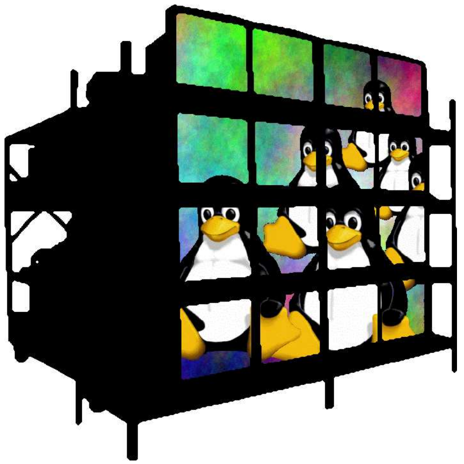


VWLib

Video Wall Library

Versions from 19991123

Computing systems supported:

- Generic uniprocessor unix
- AFAPI generic unix SMPs
- AFAPI IA32 Linux clusters

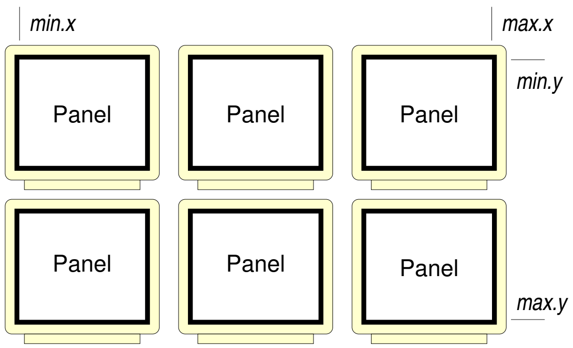
Display libraries supported:

- VGA library
- Vesa Frame Buffer
- GGI library (pending)
- X11 MIT-SHM library

`http://www.cs.uky.edu/~aggregate/`

Prof. Hank Dietz & “The Aggregate”
Department of Electrical Engineering
University of Kentucky
Lexington, KY 40506-0046
`hankd@engr.uky.edu`

Terminology & Conventions



Wall dimensions and panel positions in inches...

Panel: An individual PE’s 2D display or an image buffer within a single PE; routines starting with `vp` operate with respect to panels

Wall: A set of individual PE panel displays treated as a single logical 2D entity; routines starting with `vw` operate with respect to walls

Space: The set of all Walls; each PE displays at most one panel within one wall

Depth: Image pixel depth in bits; can be 8, 16, 24, or 32, all treated as true-color; 32 bit has 8-bit alpha channel (pending)

Routines marked with • must be called by *all* PEs

Main Interface Routines

`void vpclear(vwbuf *p)`
Clear (to black) panel `*p`

`void vwclear(vwwall *w)`
Clear (to black) wall `*w`

`void vwdefSpace(vwwall *w)`
• Install wall `w` in space; if `w` is 0, clear space

`void vwdefWalls(char *f)`
Define walls in config file `f`

`void vwexit()`
• Does AFAPI `p_exit()` and then exits VWLib

`vwwall *vwgetWall(char *n)`
Get pointer to wall named `n`

`void vwinit(void)`
Initialize video wall; this does the AFAPI `p_init()`, attempts to read the configuration file named by environment variable `VWSPACE`, and sets the `"default"` wall

Display Device Routines

`void vppaint(vwbuf *p)`
Paint panel `*p` onto this PE display

`void vwbegin(void)`
Initialize this PE’s display device

`void vwend(void)`
Release this PE’s display device

`void vwpaint(vwwall *w)`
Paint this PE’s portion of wall `*w`

Image File I/O Routines

Current restrictions: only 24-bit raw PPM files (P6 prefix) are handled directly; other formats are handled by using the ImageMagik `convert` utility to make a temporary PPM file which is then mapped.

`vwbuf vpmapFile(char *f)`
Open and read-only memory map the contents of the file named `f`, return is a panel image; files that are literally mapped will immediately reflect file updates made by other programs that change only the file data (not the header)

`void vpunmapFile(char *f)`
Unmaps the file `f`

`int vpwriteFile(char *f, vwbuf i)`
Writes the panel image `i` to the file named `f`; return is 0 for no error

`vwbuf vwmapFile(char *f)`
• Like `vpmapFile()`, except that file is sent to wall PEs that do not currently have a copy

`void vwunmapFile(char *f)`
• Unmaps the file `f` and destroys PE copies

Resampling, Magnifying, & Scaling

Currently, only 24-bit source images are supported; alpha channel info is ignored or handled by fractional dithering. Center positions are scaled 0:1. Magnifications are relative to 1.0 being the largest aspect-preserving size that would fit the image entirely within the wall.

`float vpaspect(vwbuf *p)`
Returns aspect ratio of panel image by pixel count; result is x/y

`void vpres(vwbuf *dp, vwbox d, vwbuf *sp, vwfbox s, int f)`
Resample (scale and crop with smoothing and color remapping) area `s` of panel image `*sp`, placing the result in area `d` of panel `*dp`, with a fractional opacity of `f`/255

float vwaspect(vwwall *w, vwfbox a)
Returns aspect ratio of area **a** of wall ***w** using inch dimensions (independent of pixel counts); result is x/y

void vwmag(vwwall *dw, vwbox d, vwfxy dc, vwbuf *sp, vwfxy sc, float m, int f)
Resample (scale and crop with smoothing and color remapping) panel image ***sp** magnifying by a factor of **m** so that image position **sc** is centered over position **dc** in area **d** of wall ***dw**, with a fractional opacity of **f**/255

void vwres(vwwall *dw, vwfbox d, vwbuf *sp, vwfbox s, int f)
Resample (scale and crop with smoothing and color remapping) area **s** of panel image ***sp**, placing the result in area **d** of wall ***dw**, with a fractional opacity of **f**/255

void vwscale(vwwall *dw, vwbox d, vwfxy dc, vwbuf *sp, vwfxy sc, vwfxy sm, int f)
Resample (scale and crop with smoothing and color remapping) panel image ***sp** magnifying by a factor of **sm** so that image position **sc** is centered over position **dc** in area **d** of wall ***dw**, with a fractional opacity of **f**/255

Timer Routines

Times are measured in float seconds since the reference time; typical accuracy is better than 10μs.

int vpwait(vwtime d)
Wait until deadline **d**; return is **VWONTIME** or **VWLATE** if it is later than **d**

vwtime vwnow(void)
Float seconds since reference time

void vwreftime(void)
• Establish a cluster-wide reference time

int vwwait(vwtime d)
• Synchronize all PEs, waiting until deadline **d**; return is **VWONTIME** or **VWLATE** if *any* PE is late

Mouse Routines

void vpbeginMouse(void)
Initialize the local PE mouse (pointing device)

void vpendMouse(void)
Give up control of the local PE mouse

vwinfoMouse vpreadMouse(vwtime d)
Read the local PE mouse, waiting until deadline **d** if there is no new mouse data

void vwbeginMouse(void)
• Initialize the **CPROC** PE mouse

void vwendMouse(void)
• Give up control of the **CPROC** PE mouse

vwinfoMouse vwreadMouse(vwtime d)
• Read the **CPROC** PE mouse, waiting until deadline **d** if there is no new mouse data

Special Data

NPROC, **IPROC**, & **CPROC**
Respectively, the number of PEs, number of this PE, and number of the console PE

vwbuf vwpanel;
This PE's panel buffer

vwwall *vwspace[NPROC];
Which wall each PE belongs to

Data Types & Structures

```
typedef unsigned char pixel8;
typedef unsigned short pixel16;
typedef unsigned char pixel24;
typedef unsigned int pixel32;
typedef float vwtime;

typedef struct{int x, y;} vwxy;
typedef struct{vwxy min, max;} vwbox;
typedef struct{float x, y;} vwfxy;
typedef struct{vwfxy min, max;} vwfbox;

typedef struct {
 vwxy dim;    /* pixel dimensions */
 vwbox dirty; /* dirty area */
 int depth;   /* pixel depth */
 int bytes;   /* image data bytes */
 union {
  pixel8 *data8;   pixel16 *data16;
  pixel24 *data24; pixel32 *data32;
  void *data;
 } data;      /* pixel values */
} vwbuf;      /* type of a panel */

typedef struct {
 vwfbox all;  /* inch positions */
 vwfbox one[NPROC];
 char *name;  /* name of wall */
} vwwall;     /* type of a wall */

typedef struct {
 vwfxy pos;   /* x,y position, 0:1 */
 int left, center, right;
} vwinfoMouse;
```

Sample Program

The following program implements interactive pan and zoom over a mapped image file. The image file name is given as a command line argument; the console mouse position determines the center of the magnified image, from left to right, the mouse buttons zoom out, end the program, and zoom in.

```
#include "vwsup.h"

int
main(register int argc,
register char **argv)
{
 vwfbox destarea = { {0.0, 0.0},
                     {1.0, 1.0} };
 vwinfoMouse mouse;
 vwbuf image;
 register double mag = 1.0;

 vwinit();
 image = vwmapFile(argv[1]);
 vwbegin();
 vwbeginMouse();

 for (;;) {
  vwfxy t;
  vwclear(0);
  mouse = vwreadMouse(0);
  t.x = 1.0 - mouse.pos.x;
  t.y = 1.0 - mouse.pos.y;
  if (mouse.left) mag *= 0.95;
  if (mouse.center) {
   vwexit(); exit(0);
  }
  if (mouse.right) mag *= 1.05;
  vwmag(0, destarea, mouse.pos,
        &image, t, mag, 255);
  vwpaint(0);
 }
}
```

Compile & Run

Compilation requires VWLib and the appropriate graphics library for each PE. If there is more than one PE, use the VWLib distributed with AFAPI.

Some VWLib versions use environment variables to control the running system. **VWWIDE** and **VWTALL** override settings for the PE panel. Setting **VWPROJECTION** to **rear** corrects in software for rear-screen projection. Setting **VWSUBPIXEL** to **rgb** or **bgr** specifies spatial order for subpixel rendering.