NAME

XDGA - Client library for the XFree86-DGA extension.

SYNOPSIS

#include <X11/extensions/xf86dga.h>

Bool **XDGAQueryExtension**(
Display ***dpy*,
int **eventBase*,
int **errorBase*)

Bool **XDGAQueryVersion**(
Display ***dpy*,
int **majorVersion*,
int **minorVersion*)

XDGAMode ****XDGAQueryModes**(
Display ***dpy*,
int *screen*,
int **num*)

XDGADevice ****XDGASetMode**(
Display ***dpy*,
int *screen*,
int *mode*)

Bool **XDGAOpenFramebuffer**(
Display ***dpy*,
int *screen*)

void **XDGACloseFramebuffer**(
Display ***dpy*,
int *screen*)

void **XDGASetViewport**(
Display ***dpy*,
int *screen*,
int *x*,
int *y*,
int *flags*)

void **XDGAInstallColormap**(
Display **dpy*,
int *screen*,
Colormap *cmap*)

Colormap **XDGACreateColormap**(
Display **dpy*,
int *screen*,
XDGADevice **device*,
int *alloc*)

void **XDGASelectInput**(
Display **dpy*,
int *screen*,
long *event_mask*)

void **XDGAFillRectangle**(
Display **dpy*,
int *screen*,
int *x*,
int *y*,
unsigned int *width*,
unsigned int *height*,
unsigned long *color*)

void **XDGACopyArea**(
Display **dpy*,
int *screen*,
int *srcx*,
int *srcy*,
unsigned int *width*,
unsigned int *height*,
int *dstx*,
int *dsty*)

void **XDGACopyTransparentArea**(
Display **dpy*,
int *screen*,
int *srcx*,
int *srcy*,

unsigned int *width*,
unsigned int *height*,
int *dstx*,
int *dsty*,
unsigned long *key*)

int **XDGAGetViewportStatus**(
Display **dpy*,
int *screen*)

void **XDGASync**(
Display **dpy*,
int *screen*)

Bool **XDGASetClientVersion**(
Display **dpy*)

void **XDGAChangePixmapMode**(
Display **dpy*,
int *screen*,
int **x*,
int **y*,
int *mode*)

void **XDGAKeyEventToXKeyEvent**(
XDGAKeyEvent **dk*,
XKeyEvent **xk*)

DESCRIPTION

The **XFree86-DGA** extension is an X server extension for allowing client programs direct access to the video frame buffer. This is a brief description of the programming interface for version 2.0 of the **XFree86-DGA** extension.

XFree86-DGA is not intended as a direct rendering API, but rather, as a mechanism to "get the X Server out of the way" so that some other direct rendering API can have full access to the hardware. With this in mind, DGA does provide clients some direct access to the hardware without requiring a separate rendering API, but this access is limited to direct linear framebuffer access.

Most of the reasons for the **XFree86-DGA** extension’s existence are now better served in other ways.

Further development of this extension is not expected, and it may be deprecated in a future release. The features that continue to be useful will either be provided through other existing mechanisms, or through an extension that address those needs more specifically.

XFree86-DGA is initialized by passing a number corresponding to a valid *XDGAMode* to **XDGASetMode**(). Clients can get a list of valid modes from **XDGAQueryModes**(). Each *XDGAMode* corresponds to a different framebuffer layout.

XDGAQueryModes() returns a pointer to an array of *XDGAMode*s which are valid for the given screen. *num* is the number of elements in the array. The returned array can be freed with XFree(3). The *XDGAMode* structure is as follows:

```
typedef struct {
  int num;
  char *name;
  float verticalRefresh;
  int flags;
  int imageWidth;
  int imageHeight;
  int pixmapWidth;
  int pixmapHeight;
  int bytesPerScanline;
  int byteOrder;
  int depth;
  int bitsPerPixel;
  unsigned long redMask;
  unsigned long greenMask;
  unsigned long blueMask;
  short visualClass;
  int viewportWidth;
  int viewportHeight;
  int xViewportStep;
  int yViewportStep;
  int maxViewportX;
  int maxViewportY;
  int viewportFlags;
  int reserved1;
  int reserved2;
} XDGAMode;
```

num A unique identifying number (*num* > 0) for the mode. This is the number referenced when initializing the mode.

name The name of the corresponding modeline as given in the xorg.conf file.

verticalRefresh
The vertical refresh rate for the modeline (in Hz).

flags Any of the following may be OR'd together:

XDGAConcurrentAccess
Indicates that concurrent client/server access to the framebuffer is possible. If this flag is not set it is very important to call **XDGASync**() before directly accessing the framebuffer if a call to **XDGAFillRectangle**(), **XDGACopyArea**() or **XDGACopyTransparentArea**() or any Xlib rendering function has been made prior to such accesses.

XDGASolidFillRect
Indicates that **XDGAFillRectangle**() is supported.

XDGABlitRect
Indicates that **XDGACopyArea**() is supported.

XDGABlitTransRect
Indicates that **XDGACopyTransparentArea**() is supported.

XDGAPixmap
Indicates that a Pixmap will be returned when the mode is initialized. This means that rendering with Xlib is possible for this mode.

XDGAInterlaced

XDGADoublescan
Indicates that the mode is an interlaced or doublescan mode.

imageWidth

imageHeight
The width and height of the framebuffer area accessible by the client. This rectangle is always justified to the upper left-hand corner.

pixmapWidth

pixmapHeight

The width and height of the framebuffer area accessible by Xlib. This rectangle is always justified to the upper left-hand corner. These fields are only valid if the **XDGAPixmap** flag is set in the *flags* field.

bytesPerScanline

The pitch of the framebuffer in bytes.

byteOrder

MSBFirst or **LSBFirst**.

depth The number of bits in each pixel which contain usable data.

bitsPerPixel

The number of bits taken up by each pixel.

redMask

greenMask

blueMask

The RGB masks. These do not apply to color-indexed modes.

visualClass

TrueColor, **PseudoColor**, **DirectColor**, etc.

viewportWidth

viewportHeight

The dimensions of the portion of the framebuffer which will be displayed on the screen.

xViewportStep

yViewportStep

The granularity of the x,y viewport positioning possible with the **XDGASetViewport**() function.

maxViewportX

maxViewportY
The maximum x and y positions possible with the **XDGASetViewport**() function.

viewportFlags
Any of the following may be OR'd together

XDGAFlipRetrace
Indicates that the hardware can switch viewports during the vertical retrace.

XDGAFlipImmediate
Indicates that the hardware can switch viewports immediately without waiting for the vertical retrace.

XDGASetMode() initialises the *XDGAMode* corresponding to *num*. To exit DGA mode and return to normal server operation, call **XDGASetMode**() with *num* set to zero. **XDGASetMode**() returns a pointer to an *XDGADevice* if successful. The XDGADevice can be freed with XFree(3). The *XDGADevice* structure is as follows:

```
typedef struct {
  XDGAMode mode;
  unsigned char *data;
  Pixmap pixmap;
} XDGADevice;
```

mode The *XDGAMode* structure, identical to the information returned by **XDGAQueryModes**().

data If direct framebuffer access is desired and possible, this field will contain a pointer to the mapped framebuffer memory. Generally, this field will be zero unless a call to **XDGAOpenFramebuffer**() is made prior to initialization of the mode.

pixmap If the mode supports Xlib rendering as indicated by **XDGAPixmap** in the *flags* field, this will contain a Pixmap handle suitable for passing as the drawable argument to Xlib functions. This field will be zero if Xlib rendering is not supported.

XDGAQueryExtension() checks for the presence of the extension and returns the event and error bases.

XDGAQueryVersion() returns the **XFree86-DGA** major and minor version numbers.

XDGAOpenFramebuffer() maps the framebuffer memory. The client needs sufficient privileges to be able to do this. **XDGAOpenFramebuffer**() should be called prior to initializing a DGA mode if direct

framebuffer access is desired for that mode. **XDGAOpenFramebuffer**() does not need to be called if direct framebuffer access is not required. If the framebuffer is opened,

XDGACloseFramebuffer() should be called prior to client exit to unmap the memory.

XDGAChangePixmapMode() can be used to change between two pixmap sizes in cases where a Pixmap is available for Xlib rendering. The following values for the *mode* parameter are available:

XDGAPixmapModeLarge
The pixmap size is defined by the *pixmapWidth* and *pixmapHeight* fields in the *XDGAMode* structure. The *x* and *y* values are ignored in this case.

XDGAPixmapModeSmall
The pixmap size is defined by the *viewportWidth* and *viewportHeight* fields in the *XDGAMode* structure. In this mode, the *x* and *y* values specify where in the framebuffer this pixmap rectangle is located. It may be placed anywhere within the Xlib renderable region described by the *pixmapWidth* and *pixmapHeight* fields in the *XDGAMode*. The *x* and *y* values returned are the resultant location of the pixmap and may be different from the requested x,y location due to platform specific alignment constraints. All Xlib rendering is clipped to this pixmap rectangle.

XDGASetViewport() sets the upper left-hand corner of the rectangle of framebuffer that is to be displayed on the screen. Not all locations may be supported by the hardware and requested locations will be adjusted according to the *xViewportStep* and *yViewportStep* fields in the *XDGAMode*.

flags can be **XDGAFlipRetrace** or **XDGAFlipImmediate** to adjust the viewport location at the next vertical retrace or immediately. Values other than the supported values advertised in the mode's *viewportFlags* field will result in hardware-specific default behavior. **XDGAFlipImmediate** will block until the flip is completed. **XDGAFlipRetrace** will generally NOT block so it is necessary to monitor the viewport status with **XDGAGetViewportStatus**(). **XDGAFlipImmediate** requests during pending **XDGAFlipRetrace** requests will be ignored.

XDGAGetViewportStatus() keeps track of the **XDGASetViewport**() requests still pending. The return value of the function will have consecutive bits set (LSB justified), each bit representing a pending viewport change. For example:

```
while(XDGAGetViewportStatus(dpy, screen));
```

waits for all pending viewport changes to finish.

```
while(0x2 & XDGAGetViewportStatus(dpy, screen));
```

waits until all but the last viewport changes have completed.

XDGACreateColormap() is similar to the Xlib function XCreateColormap(3) except that it takes an *XDGADevice* as an argument instead of a Window and Visual. Though XCreateColormap(3) may create usable colormaps in some cases, **XDGACreateColormap**() is the preferred method for creating colormaps in DGA since there may not be an advertised visual compatible with the DGA device.

XDGAInstallColormap() must be used to install colormaps in DGA mode. XInstallColormap(3) will not work.

XDGASelectInput() enables DGA's own event mechanism. This function is similar to XSelectInput(3), and all Xlib Key, Button and Motion masks are supported. The following DGA events are defined:

```
typedef struct {
  int type;            /* ButtonPress or ButtonRelease + the DGA event base*/
  unsigned long serial; /* # or last request processed by the server */
  Display *display;     /* Display the event was read from */
  int screen;          /* The screen number the event came from */
  Time time;           /* milliseconds */
  unsigned int state;   /* key or button mask */
  unsigned int button;  /* detail */
} XDGAButtonEvent;

typedef struct {
  int type;            /* KeyPress or KeyRelease + the DGA event base*/
  unsigned long serial; /* # or last request processed by the server */
  Display *display;     /* Display the event was read from */
  int screen;          /* The screen number the event came from */
  Time time;           /* milliseconds */
  unsigned int state;   /* key or button mask */
  unsigned int keycode; /* detail */
} XDGAKeyEvent;

typedef struct {
  int type;            /* MotionNotify + the DGA event base*/
  unsigned long serial; /* # or last request processed by the server */
  Display *display;     /* Display the event was read from */
```

```
  int screen;          /* The screen number the event came from */
  Time time;           /* milliseconds */
  unsigned int state;  /* key or button mask */
  int dx;              /* relative pointer motion */
  int dy;              /* relative pointer motion */
} XDGAMotionEvent;
```

XDGAKeyEventToXKeyEvent() is a helper function to translate *XDGAKeyEvent*s into *XKeyEvent*s suitable for use with XLookupKeysym(3).

XDGAFillRectangle(), **XDGACopyArea**(), and **XDGACopyTransparentArea**() are included with some reservation since DGA is not intended as a rendering API. These are merely convenience routines and are optionally supported. The associated flags will be set in the *XDGAMode*'s *flags* field if these functions are supported. These functions will be no-ops otherwise. they do not provide direct access to the hardware, but are simply context-less operations performed by the server.

XDGASync() blocks until all server rendering to the framebuffer completes. If Xlib or the 3 rendering functions above are used, **XDGASync**() must be called before the client directly accesses the framebuffer as the server rendering is asynchronous with the client and may have not completed. This is especially important if the **XDGAConcurrentAccess** flag is not set in the *XDGAMode*'s *flags* field since concurrent access by the server and client may result in a system lockup.

SEE ALSO

Xorg(1), xorg.conf(5)

AUTHORS

XFree86-DGA version 2 was written by Mark Vojkovich. Version 1 was written by Jon Tombs, Harm Hanemaayer, Mark Vojkovich.