NAME

Xrandr - X Resize, Rotate and Reflection extension.

SYNTAX

```
#include <X11/extensions/Xrandr.h>

Bool XRRQueryExtension (Display *dpy,
        int *event_base_return, int *error_base_return);

Status XRRQueryVersion (Display *dpy,
        int *major_version_return,
        int *minor_version_return);

XRRScreenConfiguration *XRRGetScreenInfo (Display *dpy,
        Drawable draw);

void XRRFreeScreenConfigInfo (
        XRRScreenConfiguration *config);

Status XRRSetScreenConfig (Display *dpy,
        XRRScreenConfiguration *config,
        Drawable draw,
        int size_index,
        Rotation rotation,
        Time timestamp);

Status XRRSetScreenConfigAndRate (Display *dpy,
                                  XRRScreenConfiguration *config,
                                  Drawable draw,
                                  int size_index,
                                  Rotation rotation,
                                  short rate,
                                  Time timestamp);

Rotation XRRConfigRotations(
        XRRScreenConfiguration *config,
        Rotation *current_rotation);

Time XRRConfigTimes (
        XRRScreenConfiguration *config,
```

```
        Time *config_timestamp);

XRRScreenSize *XRRConfigSizes(
        XRRScreenConfiguration *config,
        int *nsizes);

short *XRRConfigRates (
        XRRScreenConfiguration *config,
        int size_index,
        int *nrates);

SizeID XRRConfigCurrentConfiguration (
        XRRScreenConfiguration *config,
        Rotation *rotation);

short XRRConfigCurrentRate (
        XRRScreenConfiguration *config);

int XRRRootToScreen(
        Display *dpy,
        Window root);

void XRRSelectInput(Display *dpy, Window window, int mask);

/*
 * intended to take RRScreenChangeNotify,  or
 * ConfigureNotify (on the root window)
 * returns 1 if it is an event type it understands, 0 if not
 */
int XRRUpdateConfiguration(XEvent *event);

/*
 * the following are always safe to call, even if RandR is
 * not implemented on a screen
 */
Rotation XRRRotations(
        Display *dpy, int screen,
        Rotation *current_rotation);

XRRScreenSize *XRRSizes(Display *dpy,
```

int *screen*, int *nsizes);

short *XRRRates (Display *dpy, int *screen*,
int *size_index*, int *nrates);

Time XRRTimes (Display *dpy, int *screen*, Time *config_timestamp);

ARGUMENTS

display Specifies the connection to the X server.

screen Specifies which screen.

draw Specifies the screen.

rotation Specifies the possible rotations or reflections of the screen.

current_rotation
Specifies the current rotations and reflection of the screen.

timestamp Specifies the server timestamp.

config_timestamp
Specifies the timestamp when the screen was last (re)configured.

config Specifies the screen configuration being used.

sizes Specifies the array of supported sizes.

rate Specifies the refresh rate in Hz.

DATATYPES

Rotations/Reflections

Can be any of:
#define RR_Rotate_0 1
#define RR_Rotate_90 2
#define RR_Rotate_180 4
#define RR_Rotate_270 8

```
/* new in 1.0 protocol, to allow reflection of screen */
/* reflection is applied after rotation */

#define RR_Reflect_X        16
#define RR_Reflect_Y        32


typedef struct {
   int     width, height;
   int     mwidth, mheight;
} XRRScreenSize;

typedef struct {
   int type;                                /* event base */
   unsigned long serial;  /* # of last request processed by server */
   Bool send_event;             /* true if this came from a SendEvent request */
   Display *display;            /* Display the event was read from */
   Window window;                       /* window which selected for this event */
   Window root;                         /* Root window for changed screen */
   Time timestamp;                      /* when the screen change occurred */
   Time config_timestamp;       /* when the last configuration change */
   SizeID size_index;
   SubpixelOrder subpixel_order;
   Rotation rotation;
   int width;
   int height;
   int mwidth;
   int mheight;
} XRRScreenChangeNotifyEvent;
```

The **XRRScreenSize** structure contains a possible root size in pixels and in millimeters.

A **XRRScreenChangeNotifyEvent** is sent to a client that has requested notification whenever the screen configuration is changed. A client can perform this request by calling **XRRSelectInput**, passing the display, the root window, and the **RRScreenChangeNotifyMask** mask.

XRRScreenConfiguration is an opaque data type containing the configuration information for a screen.

Timestamps

Time stamps are included and must be used to ensure the client is playing with a full deck: the screen may change properties on the fly and this ensures its knowledge of the configuration is up to date. This is to help issues when screens may become hot-pluggable in the future.

DESCRIPTION

Xrandr is a simple library designed to interface the X Resize and Rotate Extension. This allows clients to change the size and rotation of the root window of a screen, along with the ability to reflect the screen about either axis (if supported by the implementation). Rotation and reflection may be implemented by software and may result in slower performance if rotation and reflection are implemented in this fashion (as are all implementations as of October 2002).

The Xrandr library does some minimal caching to avoid roundtrips to provide clients frequently used information. See "The X Resize and Rotate Extension" for a detailed description; also note that depth switching, as described in the document is not implemented, and may (or may not) ever be implemented, as display memory is growing rapidly, and toolkits are already beginning to support migration, mitigating the need for depth switching. If it is implemented in the future, we expect to do so via an upward compatible extension to the current library/protocol; functionality described here should continue to work.

Rotation and reflection and how they interact can be confusing. In Randr, the coordinate system is rotated in a counter-clockwise direction relative to the normal orientation. Reflection is along the window system coordinate system, not the physical screen X and Y axis, so that rotation and reflection do not interact. The other way to consider reflection is to treat it as specified in the "normal" orientation, before rotation.

The **XRRScreenChangeNotify** event is sent to clients that ask to be informed whenever the root window configuration changes. Configuration changes may include resolution, physical size, subpixel order (see XRender(3)), and rotation. Note that changes to any or all of these could occur due to external events (user control in the X server, a different monitor/flat panel display being hot-plugged) and is not only the result of a protocol/library request to the X server.

Additionally, to eliminate a potential race condition, this event may be generated immediately upon selecting for notification if the screen has changed since the client of Xrandr connected to the X server, to enable reliable screen resolution changing when a user may log in and change the configuration while one or many clients are starting up.

Xlib notification

Clients must call back into Xlib using **XRRUpdateConfiguration** when screen configuration change

notify events are generated (or root window configuration changes occur, to update Xlib's view of the resolution, size, rotation, reflection or subpixel order. Generally, toolkits will perform this operation on behalf of applications; we did not want to change display structure data behind the back of toolkits, as in multithreaded clients, various race conditions might occur. Toolkits should provide clients some mechanism for notification of screen change, of course.

FUNCTIONS

There are two classes of interfaces: those which can be safely called even if RandR is not implemented on a screen (to make common idioms not dependent on the server having support), and those which will return errors if the extension is not present.

XRRRotations returns both the possible set of rotations/reflections supported (as a bitmask) as the value of the function, along with the current rotation/reflection of the screen.

XRRSizes returns the size and a pointer to the current sizes supported by the specified screen. The first size specified is the default size of the server. If RandR is not supported, it returns 0 for the number of sizes.

XRRRates returns a pointer to the rates supported by the specified size. If RandR is not supported, it returns 0 for the number of rates.

XRRTimes returns the time last reported by the server along with the timestamp the last configuration changed. If the configuration has changed since the client last updated its view of the server time, requests to change the configuration will fail until the client has an up to date timestamp.

XRRRootToScreen returns the screen number given a root window (for example, from an **XRRScreenChangeNotifyEvent**).

The rest of the functions will fail if applied to screens not implementing the RandR extension. **XRRSetScreenConfig** sets the screen size and rotation and reflection to the desired values on the screen specified by *draw*, or returns a **BadValue** error. *size_index* specifies which size configuration is to be used, *rotation* specifies which rotation or reflection is to be used (or a **BadValue** error is returned). The *timestamp* is used by the server to make sure the client has up to date configuration information. Status is returned to indicate success or failure; a client must refresh its configuration information if it fails and try the call again (by calling **XRRGetScreenInfo**).

XRRSetScreenConfigAndRate is like **XRRSetScreenConfig** but also sets the refresh rate. If specified rate is not supported a **BadValue** error is returned.

XRRConfigRotations, XRRConfigSizes, XRRConfigCurrentConfiguration, XRRConfigTimes, XRRConfigRates, and **XRRConfigCurrentRate** are used to get specific configuration information out of a screen configuration.

XRRGetScreenInfo returns a screen configuration for later use; the information is private to the library. Call **XRRFreeScreenConfigInfo** to free this information when you are finished with it. It forces a round trip to the server.

Other functions include: **XRRQueryExtension** which returns the event and error base codes, **XRRQueryVersion** , which returns the current version of the extension (this information is cached by the library).

RESTRICTIONS

Xrandr will remain upward compatible after the current 1.0 release.

AUTHOR

Jim Gettys, and Keith Packard, HP.