

**XFillRectangle, XFillRectangles, XFillPolygon, XFillArc, XFillArcs** – fill rectangles, polygons, or arcs

**XFillRectangle**(*display, d, gc, x, y, width, height*)

**Display** \**display*;  
**Drawable** *d*;  
**GC** *gc*;  
**int** *x, y*;  
**unsigned int** *width, height*;

**XFillRectangles**(*display, d, gc, rectangles, nrectangles*)

**Display** \**display*;  
**Drawable** *d*;  
**GC** *gc*;  
**XRectangle** \**rectangles*;  
**int** *nrectangles*;

**XFillPolygon**(*display, d, gc, points, npoints, shape, mode*)

**Display** \**display*;  
**Drawable** *d*;  
**GC** *gc*;  
**XPoint** \**points*;  
**int** *npoints*;  
**int** *shape*;  
**int** *mode*;

**XFillArc**(*display, d, gc, x, y, width, height, angle1, angle2*)

**Display** \**display*;  
**Drawable** *d*;  
**GC** *gc*;  
**int** *x, y*;  
**unsigned int** *width, height*;  
**int** *angle1, angle2*;

**XFillArcs**(*display, d, gc, arcs, narcs*)

**Display** \**display*;  
**Drawable** *d*;  
**GC** *gc*;  
**XArc** \**arcs*;  
**int** *narcs*;

<i>angle1</i>	Specifies the start of the arc relative to the three-o'clock position from the center, in units of degrees * 64.
<i>angle2</i>	Specifies the path and extent of the arc relative to the start of the arc, in units of degrees * 64.
<i>arcs</i>	Specifies an array of arcs.
<i>d</i>	Specifies the drawable.
<i>display</i>	Specifies the connection to the X server.
<i>gc</i>	Specifies the GC.
<i>mode</i>	Specifies the coordinate mode. You can pass <b>CoordModeOrigin</b> or <b>CoordModePrevious</b> .
<i>narcs</i>	Specifies the number of arcs in the array.
<i>npoints</i>	Specifies the number of points in the array.
<i>nrectangles</i>	Specifies the number of rectangles in the array.

<i>points</i>	Specifies an array of points.
<i>rectangles</i>	Specifies an array of rectangles.
<i>shape</i>	Specifies a shape that helps the server to improve performance. You can pass <b>Complex</b> , <b>Convex</b> , or <b>Nonconvex</b> .
<i>width</i>	
<i>height</i>	Specify the width and height, which are the dimensions of the rectangle to be filled or the major and minor axes of the arc.
<i>x</i>	
<i>y</i>	Specify the x and y coordinates, which are relative to the origin of the drawable and specify the upper-left corner of the rectangle.

The **XFillRectangle** and **XFillRectangles** functions fill the specified rectangle or rectangles as if a four-point **FillPolygon** protocol request were specified for each rectangle:

[x,y] [x+width,y] [x+width,y+height] [x,y+height]

Each function uses the x and y coordinates, width and height dimensions, and GC you specify.

**XFillRectangles** fills the rectangles in the order listed in the array. For any given rectangle, **XFillRectangle** and **XFillRectangles** do not draw a pixel more than once. If rectangles intersect, the intersecting pixels are drawn multiple times.

Both functions use these GC components: function, plane-mask, fill-style, subwindow-mode, clip-x-origin, clip-y-origin, and clip-mask. They also use these GC mode-dependent components: foreground, background, tile, stipple, tile-stipple-x-origin, and tile-stipple-y-origin.

**XFillRectangle** and **XFillRectangles** can generate **BadDrawable**, **BadGC**, and **BadMatch** errors.

**XFillPolygon** fills the region closed by the specified path. The path is closed automatically if the last point in the list does not coincide with the first point. **XFillPolygon** does not draw a pixel of the region more than once. **CoordModeOrigin** treats all coordinates as relative to the origin, and **CoordModePrevious** treats all coordinates after the first as relative to the previous point.

Depending on the specified shape, the following occurs:

- If shape is **Complex**, the path may self-intersect. Note that contiguous coincident points in the path are not treated as self-intersection.
- If shape is **Convex**, for every pair of points inside the polygon, the line segment connecting them does not intersect the path. If known by the client, specifying **Convex** can improve performance. If you specify **Convex** for a path that is not convex, the graphics results are undefined.
- If shape is **Nonconvex**, the path does not self-intersect, but the shape is not wholly convex. If known by the client, specifying **Nonconvex** instead of **Complex** may improve performance. If you specify **Nonconvex** for a self-intersecting path, the graphics results are undefined.

The fill-rule of the GC controls the filling behavior of self-intersecting polygons.

This function uses these GC components: function, plane-mask, fill-style, fill-rule, subwindow-mode, clip-x-origin, clip-y-origin, and clip-mask. It also uses these GC mode-dependent components: foreground, background, tile, stipple, tile-stipple-x-origin, and tile-stipple-y-origin.

**XFillPolygon** can generate **BadDrawable**, **BadGC**, **BadMatch**, and **BadValue** errors.

For each arc, **XFillArc** or **XFillArcs** fills the region closed by the infinitely thin path described by the specified arc and, depending on the arc-mode specified in the GC, one or two line segments. For **ArcChord**, the single line segment joining the endpoints of the arc is used. For **ArcPieSlice**, the two line segments joining the endpoints of the arc with the center point are used. **XFillArcs** fills the arcs in the order listed in the array. For any given arc, **XFillArc** and **XFillArcs** do not draw a pixel more than once. If regions intersect, the intersecting pixels are drawn multiple times.

Both functions use these GC components: function, plane-mask, fill-style, arc-mode, subwindow-mode, clip-x-origin, clip-y-origin, and clip-mask. They also use these GC mode-dependent components:

foreground, background, tile, stipple, tile-stipple-x-origin, and tile-stipple-y-origin.

**XFillArc** and **XFillArcs** can generate **BadDrawable**, **BadGC**, and **BadMatch** errors.

**BadDrawable** A value for a Drawable argument does not name a defined Window or Pixmap. **BadGC** A value for a GContext argument does not name a defined GContext. **BadMatch** An **InputOnly** window is used as a Drawable. **BadMatch** Some argument or pair of arguments has the correct type and range but fails to match in some other way required by the request. **BadValue** Some numeric value falls outside the range of values accepted by the request. Unless a specific range is specified for an argument, the full range defined by the argument's type is accepted. Any argument defined as a set of alternatives can generate this error.

**XDrawArc(3X11), XDrawPoint(3X11), XDrawRectangle(3X11)**

*Xlib – C Language X Interface*