Tema IV
Contextos gráficos y regiones (R-1.1)

Programación en Entornos Interactivos.

16 de febrero de 2012

Dpto. Lenguajes y Sistemas Informáticos
Universidad de Alicante

Resumen

Introducción. Creación de un contexto gráfico. Estructura XGCValues. Manipulación de un contexto gráfico. Modos de transferencia. Regiones.

Introducción.

- Cualquier 'elemento' que pueda ser obtenido como resultado de realizar una operación gráfica consta de una serie de atributos que pueden ser 'modificados' a voluntad por el programador, por ejemplo los *colores de fondo* y *primer plano*, *tipo de letra* empleado, *grosor de una línea*, etc...
- Estos atributos se guardan en una estructura que se denomina **Contexto Gráfico**, abreviado **CG**.
- Los **CG** pueden ser modificados tanto desde Xlib como desde cualquier biblioteca.
- Qt los llama '`QPainter`', Gtkmm puede usar '`Gdk::GC`' o '`Gdk::Cairo::Context`'.

Creación de un contexto gráfico. (I)

- Los **CG** se crean con la función: **XCreateGC(dsp, drw, mask, &vals)**. Esta función devuelve un ID para el nuevo **CG** recién creado.
- Al crear un contexto gráfico lo asociamos con un '*drawable*' específico, pero puede ser utilizado con cualquier otro de la misma profundidad.
- El parámetro **vals** anterior es una estructura del tipo **XGCValues**, cuyos campos podremos ver más adelante.
- El parámetro **mask** es una máscara que dice que campos de la estructura anterior contienen información válida. Si vale 0, el **CG** será creado con valores por defecto distintos para cada campo.

Creación de un contexto gráfico. (II)

```
1   /* Ejemplo de creacion de un contexto grafico */
    XGCValues      gcv;
3   GC             gc;
    unsigned long mascara;
5
    gcv.foreground = 1;  /* Color de primer plano */
7   gcv.background = 2;  /* Color de fondo */
    /* Estilo de linea */
9   gcv.line_style = LineOnOffDash;

11  /* Solo modificamos:
      1) Color de primer plano.
13    2) Color de fondo.
      3) Estilo de las lineas dibujadas. */
15
    mascara = (GCForeground |
17             GCBackground | GCLineStyle );
    gc      = XCreateGC(dsp, win, mascara, &gcv);
```

Estructura XGCValues.

```
typedef struct {
    int function;                    int ts_x_origin;
    unsigned long plane_mask;        int ts_y_origin;
    unsigned long foreground;        Font font;
    unsigned long background;        int subwindow_mode;
    int line_width;                  Bool
        graphics_exposures;
    int line_style;                  int clip_x_origin;
    int cap_style;                   int clip_y_origin;
    int join_style;                  Pixmap clip_mask;
    int fill_style;                  int dash_offset;
    int fill_rule;                   char dashes;
    int arc_mode;                } XGCValues;
    Pixmap tile;
    Pixmap stipple;
```

Manipulación de un contexto gráfico. (I)

- De forma general, podemos modificar los distintos atributos de un **CG** con la función: **XChangeGC(dsp, gc, mask, &vals)**, donde `mask` y `vals` tienen los mismos significados que en **XCreateGC**.
- Además, `Xlib` proporciona diversas funciones para modificar atributos concretos de un **CG** de manera individual. A continuación vamos a estudiar algunas de ellas.

Manipulación de un contexto gráfico. (II)

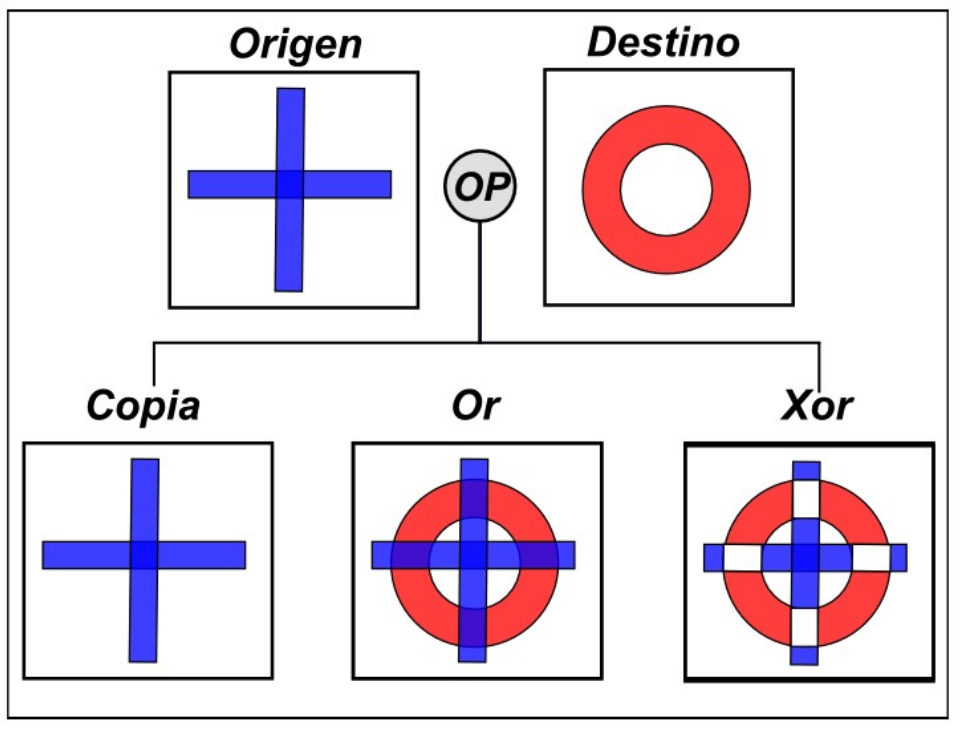
1. **XSetPlaneMask(dsp, gc, mask)**, se utiliza para elegir qué planos de un *drawable* son afectados por una operación gráfica. El parámetro `mask` tiene un 1 en cada bit que representa un plano que queremos que sea afectado. El atributo que es modificado es `GCPlaneMask`, y el valor por defecto que tiene es `AllPlanes`.
2. **XSetFunction(dsp, gc, function)**, se utiliza para cambiar el *modo de transferencia* o lo que en `Xlib` se denomina *display function*. Esta función dice con qué operación lógica se combinará cada pixel de una nueva imagen con el pixel correspondiente del *drawable* destino. El atributo que afecta es `GCFunction`. Los distintos modos de transferencia son estos:

Modos de Transferencia. (I)

Mask	Modo de Transferencia
`GXclear`	0
GXand	src AND dst
GXandReverse	src AND NOT dst
GXcopy (**valor por defecto**)	src
GXandInverted	(NOT src) AND dst
GXnoop	dst
GXxor	src XOR dst
GXor	src OR dst
GXnor	NOT (src OR dst)
GXequiv	(NOT src) XOR dst
GXinvert	NOT dst
GXorReverse	src OR (NOT dst)
GXcopyInverted	NOT src
GXorInverted	(NOT src) OR dst
GXnand	NOT (src AND dst)
GXset	1

Modos de Transferencia. (II)

Efectos de los modos COPIA/OR/XOR



Manipulación de un contexto gráfico. (I)

1. **XSetForeground(dsp, gc, pixel)** y `XSetBackground(dsp, gc, pixel)`, se emplean para cambiar los colers de primer y segundo plano respectivamente. Modifican los atributos `GCForeground` y `GCBackground`, cuyos valores por defecto son 0 y 1.
2. **XSetLineAttributes(dsp, gc, w, style, capStyle, joinStyle)**, se emplea para modificar diversos atributos de una línea cuando es dibujada (estilo -*GCLineStyle*-, grosor -*GCLineWidth*-, finales de líneas anchas -*GCCapStyle*- y líneas conectadas -*GCJoinStyle*-).
3. **XtSetFillStyle(dsp, gc, style)**, `XtSetTile(dsp, gc, tile)`, `XtSetStipple(dsp, gc, stipple)`, `XtSetFillRule(dsp, gc, rule)`.

Manipulación de un contexto gráfico. (II)

1. **XSetFont(dsp, gc, fontID)**, modifica el atributo GCFont. El valor de `fontID` se obtiene con `XLoadFont(dsp, fontName)`.
2. **XSetClipMask(dsp, gc, bitmap)**, se emplea para especificar un bitmap como máscara de recorte para cualquier operación de dibujo. El atributo modificado es `GCClipMask`, cuyo valor por defecto es None. Funciones adicionales: `XSetClipOrigin(dsp, gc, x, y)` y `XSetClipRectangles(dsp, gc, xoff, yoff, rect, nRect, ordering)`.

Regiones. (I)

- Una región es un área no rectangular de la pantalla. El servidor **X** puede representar y manipular -*de manera matemática*- este tipo de 'zonas' de la pantalla.
- El tipo **Region** está declarado de manera opaca para el programador, y solo puede utilizarse con las funciones que proporciona **Xlib** para ello.
- Las funciones que trabajan con regiones no hacen peticiones al servidor **X**, sino que realizan todos los cálculos de manera local en el cliente.
- Para poder usar estas funciones debemos incluir: <**X11/Xutil.h**>.

Regiones. (II)

Algunas de las funciones que trabajan con `regiones`:

- **XCreateRegion()** y **XDestroyRegion(region)** crean y destruyen una región. **XPolygonRegion(pointsArray, nPoints, fillRule)** crea una región a partir del polígono definido por el vector de puntos.
- **XEqualRegion(r1, r2)**, **XEmptyRegion(r)**, **XIntersectRegion(r1, r2, rResult)**, **XPointRegion(region, x,y)**,**XRectInRegion(region, x, y, w, h)**, trabajan con regiones a nivel matemático.
- **XClipBox(region, &rect)**, devuelve el rectángulo más pequeño que contiene a la región.

Regiones. (III)

- **XSetRegion(dsp, gc, region)**, se utiliza para hacer que una región sirva como máscara de recorte.
- Además tenemos disponibles funciones que calculan la **unión** y la **diferencia** de dos regiones, **mueven** una región a otro lugar, etc. . .