X-I

Xlib : les fenêtres

- Création d'une fenêtre
- Attributs d'une fenêtre
- Attributs courants
- Autres attributs
- Création d'une fenêtre: méthode générale
- Fenêtres et gestionnaire de fenêtres

X-I

Création d'une fenêtre

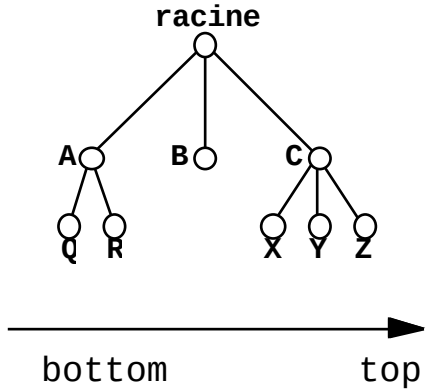
- L'arbre des fenêtres
- Caractéristiques d'une fenêtre
- Géométrie
- Couleurs de bordure et de fond
- Création et affichage

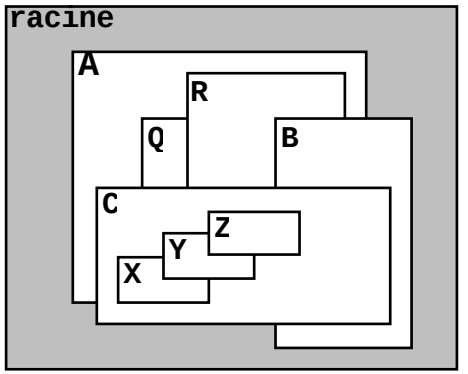
X-I

L'arbre des fenêtres

- **Une *fenêtre* est une zone rectangulaire sur un écran.**
- **Les fenêtres sont organisées en arbre:**
  - **la *fenêtre racine* occupe tout l'écran. Elle est toujours accessible, sur l'écran par défaut, par**

    `DefaultRootWindow(dpy)`
  - **les autres fenêtres ont une *mère*. Les *sœurs* sont organisées en listes correspondant à l'*ordre d'empilement* (stack-order) sur l'écran**
  - **l'affichage correspond à un *parcours préfixe en profondeur* avec *modèle du peintre*.**
  - **seule est affichée la partie d'une fenêtre *contenue* dans sa mère.**





X-I

Caractéristiques d'une fenêtre

Les caractéristiques d'une fenêtre concernent :

- **configuration**
  - **géométrie : position, largeur, hauteur, épaisseur du bord,**
  - **rang : ordre d'empilement (parmi ses sœurs);**
- **aspect et contenu**
  - **couleur ou motif de la bordure et du fond,**
  - **curseur,**
  - **comportement du contenu lors de changements de configuration (mémoire d'écran);**
- **réceptivité aux évènements**
  - **évènements reçus,**
  - **transmission éventuelle à la mère.**

Ces attributs sont modifiables.

X-I

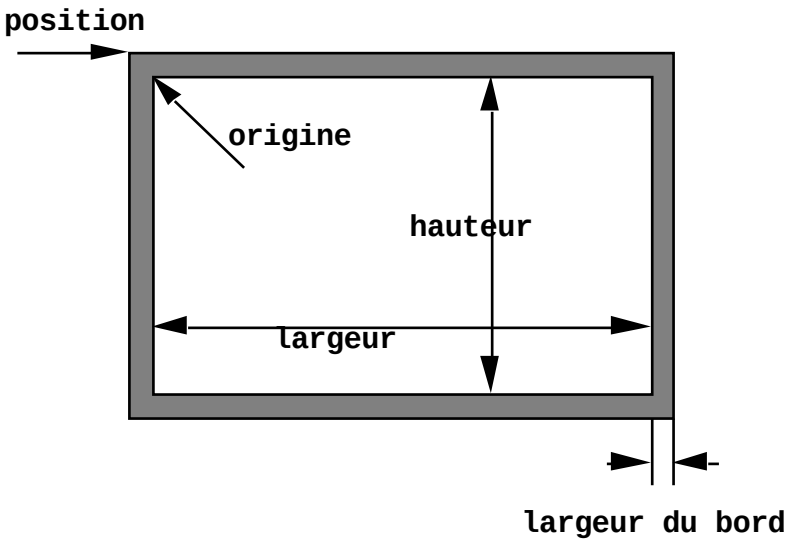
Autres attributs, plus particuliers :

- **propriétés, utilisées**
  - **pour la communication avec le gestionnaire de fenêtres (comportement de la fenêtre principale)**
  - **pour l'échange de données avec d'autres clients.**
- **caractéristiques non modifiables : (*héritées par défaut de la fenêtre mère*)**
  - **classe** `InputOnly` **ou** `InputOutPut`**,**
  - **classe visuelle (capacité de gestion et type de palette de couleurs),**
  - **profondeur (nombre de plans).**

X-I

Géométrie d'une fenêtre

- **Chaque fenêtre est placée par rapport à sa mère (coordonnées du coin nord-ouest, bordure comprise);**
- **Chaque fenêtre a son propre repère, origine au coin nord-ouest, bordure exclue;**
- **L'axe des x est horizontal, l'axe des y vertical descendant.**



X-I

Couleurs de bordure et du fond

- **Chaque fenêtre a une *couleur* de bordure et une *couleur* du fond;**
- **Les couleurs sont en fait repérées par des indices dans une *table de couleurs.***
- **Chaque indice est une *valeur de pixel* (“pixel value”) représentée par un entier long et qui est éminemment dépendant des capacités de l’écran.**
- **Deux valeurs de pixels sont toujours présentes:**

```
BlackPixel(dpy,DefaultScreen(dpy))
WhitePixel(dpy,DefaultScreen(dpy))
```

- **Regardez les valeurs de votre écran par `xdpyinfo`.**

X-I

Création d'une fenêtre (syntaxe)

```
Window XCreateSimpleWindow(
  /* le display            (Display *)    */ dpy,
  /* fenetre mere          (Window)       */ mere,
  /* position de l'origine (int)          */ x, y,
  /* largeur, hauteur      (unsigned int) */ largeur, hauteur,
  /* largeur bord          (unsigned int) */ largeur_bord,
  /* couleurs bord et fond (unsigned long)*/ c_bord, c_fond);
```

- **Exemple :**

```
fen=XCreateSimpleWindow( dpy, DefaultRootWindow(dpy),
      /* geometrie     */   10,50,500,300,
      /* largeur bord  */   6,
      BlackPixel(dpy,DefaultScreen(dpy)),
      WhitePixel(dpy,DefaultScreen(dpy)));
```

X-I

Création d'une fenêtre (suite)

- **Le type `Window` est entier long sans signe.**
- **Une variable de ce type contient un entier par lequel cleint et serveur désignent un nœud dans l'arbre des fenêtres.**
- **L'arbre est géré sur le serveur.**
- **D'autres types sont de même nature. Ils désignent les "ressources serveur", (à ne pas confondre avec les ressources des widgets !).**
- **Ces ressources sont stockées sur le serveur, et "nommés" au moyen de tels entiers.**

```
typedef unsigned long XID;

typedef XID Window;
typedef XID Drawable;
typedef XID Font;
typedef XID Pixmap;
typedef XID Cursor;
typedef XID Colormap;
typedef XID GContext;
typedef XID KeySym;
```

X-I

Affichage d'une fenêtre

(ou : une fenêtre créée n'est pas une fenêtre affichée)

- **Une fenêtre créée est insérée dans la liste des filles de sa mère, en général comme fille cadette ("on the top").**
- **Mais pour être visible, doit être affichée (*mapped*) par**

```
XMapWindow(dpy,fen)
```

- **Même "mappée", pour être vue :**
  - **toutes ses fenêtres ancêtres doivent être mappées;**
  - **elle ne doit pas être couverte par une fenêtre sœur ou cousine;**
  - **le tampon de requêtes doit avoir été vidé de cette requête d'affichage.**
- **Dans le cas d'une fenêtre principale, l'interaction avec le gestionnaire de fenêtres peut retarder l'affichage et modifier certains paramètres : largeur du bord,....**
- **Une fenêtre est retirée par**

```
XUnmapWindow(dpy,fen)
```

X-I

Exemple

- **Création et affichage d'une fenêtre principale et de trois sous-fenêtres:**

```
Window fen, fille[3], quitter;

void Installer(void)
{
  int i;

  fen = XCreateSimpleWindow( dpy, DefaultRootWindow(dpy),
    0,0,300,300,6, Noir, Blanc);
  for (i=0; i<3; i++)
    fille[i]= XCreateSimpleWindow( dpy, fen,
      50*i+10,10,40,40,4, Noir, Blanc);
  quitter = XCreateSimpleWindow( dpy, fen,
    250,250,40,40,0, Noir, Noir);
  XMapWindow(dpy,fen);
  XMapSubwindows(dpy,fen);
}
```

X-I

Gestion de l'arbre des fenêtres

Les filles d'une même fenêtre sont organisées en file susceptible d'être modifiée par des opérations. Une fenêtre plus haute dans la file recouvre à l'écran une fenêtre qui est plus basse.

- **Permutation circulaire des fenêtres des sœurs :**

```
XCirculateSubWindows(
/* le display */       Display   *dpy,
/* la mère    */       Window     fen,
/* direction  */       int        direction);
```

- **Mise en avant :**

```
XRaiseWindow(dpy, fen);
```

- **Mise en avant et affichage :**

```
XMapRaised(dpy, fen);
```

X-I

suite...

- **Changer de mère :**

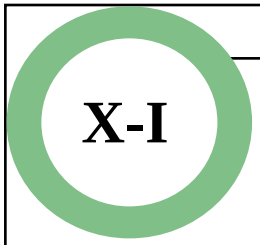
```
XReparentWindow(dpy,
  /* la fenêtre */                    fen,
  /* la nouvelle mère*/               maratre,
  /* coord. dans nouvelle mère*/      x, y);
```

- **Plus fréquent : trouver la situation d'une fenêtre `fen` dans la hierarchie :**

  - **quelle est la fenêtre racine ?**

  - **la fenêtre mère ? l'ensemble de ses soeurs ? leur nombre ?**

```
XQueryTree(Display *dpy,
  /* fenêtre */                  Window     fen,
  /* fenêtre racine */           Window *   racine,
  /* fenêtre mère   */           Window *   mere,
  /* ses filles     */           Window **  filles,
  /* nombre de filles*/          int*       nbfilles);
```



Attributs d'une fenêtre

- **Attributs**
- **Structure pour les attributs modifiables**
- **Modifier les attributs**
- **Attributs consultables**
- **Consulter les attributs**

X-I

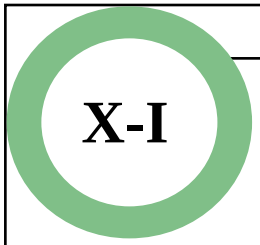
Attributs

- **Les attributs sont initialisés à la création de la fenêtre.**
- **Les caractéristiques d'une fenêtre sont *modifiables* (après la création) ou *consultables*, ou les deux.**
- **Les attributs d'une fenêtre sont accessibles via deux structures :**
  - `XSetWindowAttributes` **contient les attributs *modifiables*,**
  - `XWindowAttributes` **contient les attributs *consultables*.**
- **Plus de la moitié environ des attributs est accessible dans les deux structures.**
- **Les attributs sont modifiables**
  - **par une fonction générale** `XChangeWindowAttributes`
  - **par des fonctions spécifiques.**
- **Attributs non modifiables :**
  - **Type de fenêtre** `InputOnly`**, ou** `InputOutput`**. Une fenêtre sert seulement à recueillir des évènements.**
  - **Classe visuelle (capacité de couleurs)**
  - **Profondeur (nombre de plans de couleurs)**
- **Attributs non consultables :**
  - **Les pixmaps du bord et du fond**
  - **Le curseur.**

X-I

Remarques générales

- **Les fenêtres de classe `InputOnly` ont une épaisseur de bordure et une profondeur de plans égales à `0`.**
- **Les fenêtres créées par `XCreateSimpleWindow` héritent nécessairement des caractéristiques de leur mère. En revanche, avec la fonction complète `XCreateWindow`, on peut créer des fenêtres de classe quelconque, avec la restriction que les filles d'une fenêtre `InputOnly` sont nécessairement aussi `InputOnly`. On peut donc redéfinir les attributs concernant l'aspect des fenêtres**
  - **arrière-plan**
  - **bordure**
  - **profondeur**
  - **palette de couleurs**
  - **classe visuelle**
  - **curseur**
- **Chaque attribut a une valeur par défaut, éventuellement vide. Pour hériter des attributs de la mère on dispose de constantes du genre `CopyFromParent` ou `ParentRelative`. La valeur de la mère n'est pas nécessairement la valeur par defaut.**



Structure (Xlib.h) pour les attributs modifiables

```
typedef struct {
    Pixmap background_pixmap;   /* background or None or ParentRelative */
    unsigned long background_pixel;     /* background pixel */
    Pixmap border_pixmap;       /* border of the window */
    unsigned long border_pixel; /* border pixel value */
    int bit_gravity;            /* one of bit gravity values */
    int win_gravity;            /* one of the window gravity values */
    int backing_store;          /* NotUseful, WhenMapped, Always */
    unsigned long backing_planes;/* planes to be preseved if possible */
    unsigned long backing_pixel;/* value to use in restoring planes */
    Bool save_under;            /* should bits under be saved? (popups) */
    long event_mask;            /* set of events that should be saved */
    long do_not_propagate_mask; /* set of events that should not propagate*/
    Bool override_redirect;     /* boolean value for override-redirect */
    Colormap colormap;          /* color map to be associated with window */
    Cursor cursor;              /* cursor to be displayed (or None) */
} XSetWindowAttributes;
```

X-I

Modifier des attributs

Deux méthodes :

- **Utiliser des *fonctions utilitaires* (“convenience functions”) pour fixer ou récupérer un attribut donné :**

```
XSetWindowBorderWidth(dpy,fen,4)
```

- **Utiliser la méthode des *masques* :**
  - **Un *masque* (type `Mask` ou `unsigned long`) groupe les champs à changer;**
  - **Une *structure* contient les valeurs pour les champs à changer spécifiés dans le masque.**
- **La deuxième méthode est *générale*, mais aussi plus lourde d’écriture.**

X-I

Exemples

Soient à changer les pixels de bord et de fond d'une fenêtre.

- **Les utilitaires :**

```
SetWindowBackground(dpy, fen, pixel);
SetWindowBorder(dpy, fen, pixel);
```

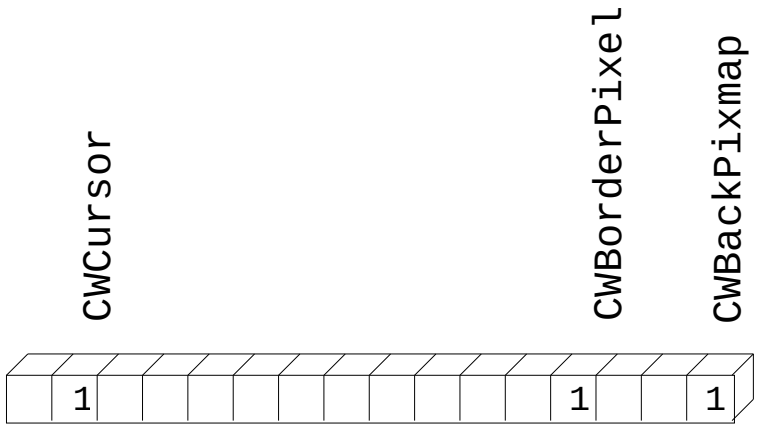
- **La méthode des masques :**

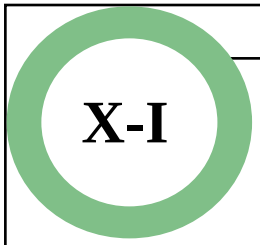
```
XSetWindowAttributes attributs;
Mask masque;

masque = CWBackPixel | CWBorderPixel;
attributs.background_pixel = Blanc;
attributs.border_pixel = Blanc;

XChangeWindowAttributes(dpy,fen,masque, &attributs);
```

X-I

Les masques des attributs modifiables

```
#define CWBackPixmap        (1L<<0)
#define CWBackPixel         (1L<<1)
#define CWBorderPixmap      (1L<<2)
#define CWBorderPixel       (1L<<3)
#define CWBitGravity        (1L<<4)
#define CWWinGravity        (1L<<5)
#define CWBackingStore      (1L<<6)
#define CWBackingPlanes     (1L<<7)
#define CWBackingPixel      (1L<<8)
#define CWOverrideRedirect  (1L<<9)
#define CWSaveUnder         (1L<<10)
#define CWEventMask         (1L<<11)
#define CWDontPropagate     (1L<<12)
#define CWColormap          (1L<<13)
#define CWCursor            (1L<<14)
```





Attributs consultables

```
typedef struct {
    int x, y;                       /* location of window */
    int width, height;              /* width and height of window */
    int border_width;               /* border width of window */
    int depth;                      /* depth of window */
    Visual *visual;                 /* the associated visual structure */
    Window root;                    /* root of screen containing window */
#if defined(__cplusplus) || defined(c_plusplus)
    int c_class;                    /* C++ InputOutput, InputOnly*/
#else
    int class;                      /* InputOutput, InputOnly*/
#endif
    int bit_gravity;                /* one of bit gravity values */
    int win_gravity;                /* one of the window gravity values */
    int backing_store;              /* NotUseful, WhenMapped, Always */
    unsigned long backing_planes;   /* planes to be preserved if possible */
    unsigned long backing_pixel;    /* value to be used when restoring planes */
    Bool save_under;                /* boolean, should bits under be saved? */
    Colormap colormap;              /* color map to be associated with window */
    Bool map_installed;             /* boolean, is color map currently installed*/
    int map_state;                  /* IsUnmapped, IsUnviewable, IsViewable */
    long all_event_masks;           /* set of events all people have interest in*/
    long your_event_mask;           /* my event mask */
    long do_not_propagate_mask;     /* set of events that should not propagate */
    Bool override_redirect;         /* boolean value for override-redirect */
    Screen *screen;                 /* back pointer to correct screen */
} XWindowAttributes;
```

X-I

Consulter des attributs

Pour *consulter des attributs*, à nouveau deux méthodes:

- **Les fonctions utilitaires : par exemple**

```
XGetGeometry(dpy,fen, &racine, &x,&y, &l, &h, &b, &p);
```

avec

```
Window racine;          racine de l'écran
int x,  y;              origine relativement à la mère
unsigned int l, h;      largeur et hauteur
unsigned int b, p;      largeur du bord et profondeur
```

- **La fonction globale :**

```
XGetWindowAttributes(dpy, fen, &attributs);
```

avec

```
» XWindowAttributes attributs;
```

X-I

Modification d'attributs (utilitaires)

- `XSetWindowBackgroundPixmap(dpy, fen, pixmap);`
- `XSetWindowBackground(dpy, fen, couleur);`
- `XSetWindowBorderPixmap(dpy, fen, pixmap);`
- `XSetWindowBorder(dpy, fen, couleur);`
- `XSetWindowBorderWidth(dpy, fen, epaisseur);`
- `XSelectInput(dpy, fen, masque_evenements);`
- `XDefineCursor(dpy, fen, curseur);`

- `XMoveWindow(dpy, fen, pos_x, pos_y);`
- `XResizeWindow(dpy, fen, largeur, hauteur);`
- `...`

X-I

Modification de la géométrie et du rang

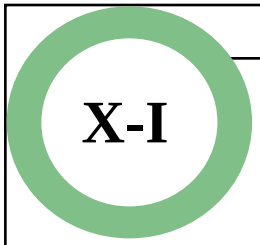
- **Pour modifier la géométrie, et la place d'une fenêtre parmi ses sœurs, il y a une fonction intermédiaire :**

```
XConfigureWindow(
/* le display */          Display *         dpy,
/* la fenêtre */          Window            fen,
/* le masque  */          unsigned int      masque,
/* les changments */      XWindowChanges *  xwc);
```

```
typedef struct {
    int x;
    int y;
    int width;
    int height;
    int border_width;
    Window sibling;
    int stack_mode;
} XWindowChanges;
```

masques associés

CWX	(1<<0)
CWY	(1<<1)
CWWidth	(1<<2)
CWHeight	(1<<3)
CWBorderWidth	(1<<4)
CWSibling	(1<<5)
CWStackMode	(1<<6)

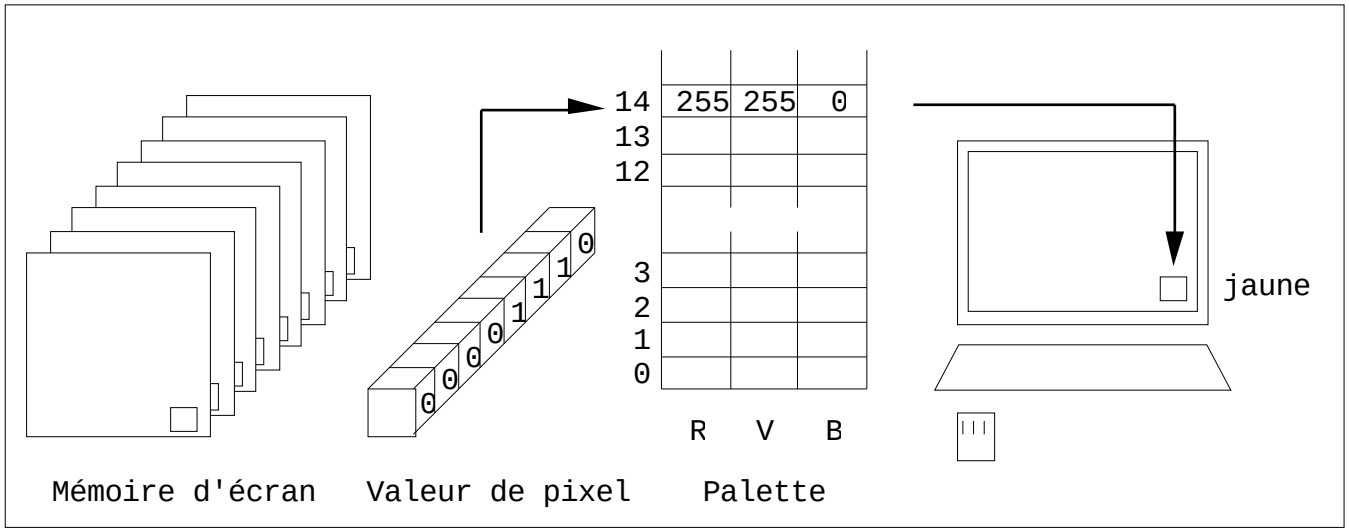


Attributs courants

- **Pixels et couleurs**
- **Pixmaps et drawables**
- **Curseurs**
- **Evènements**

X-I

Pixels et couleurs



- **Chaque valeur de pixel est un indice dans une *palette de couleur*.**
- **Chaque entrée d'une palette est une *cellule de couleur* ("color cell").**
- **Tout écran possède au moins une palette.**
- **Le nombre de bits par pixel est le *nombre de plans* (entre 1 et 24).**
- **Les macros `BlackPixel` et `WhitePixel` retournent les valeurs de pixel pour le noir et le blanc de l'écran.**

X-I

Pixmaps et drawables

- Un *pixmap* est un bloc de mémoire du serveur hors écran. Les fenêtres et pixmaps sont appelés *drawables* (planches).
  - Un pixmap est un tableau rectangulaire de valeurs de pixels.
  - Un pixmap a une profondeur.
  - Un pixmap n'a pas de position (relativement à une fenêtre ou un autre pixmap), n'a pas de palette de couleur, n'est visible que s'il est copié dans une fenêtre.
  - Toutes les fonctions de dessin s'appliquent aussi bien aux fenêtres qu'aux pixmaps.
  - Un *bitmap* est un pixmap de profondeur 1 (il n'existe pas de type bitmap).
- Les pixmaps de petite taille servent à paver le fond d'une fenêtre.
- Les bitmaps peuvent servir à imprimer des motifs superposés ou aux tracés.
- Création des pixmaps et bitmaps à partir du programme `bitmap`.

X-I

Curseurs

- **Chaque fenêtre a son propre curseur (par défaut celui de maman).**
- **Les 77 curseurs prédéfinis ont des noms symboliques (`int`) que l'on inclut par**

```
#include <X11/cursorfont.h>
```

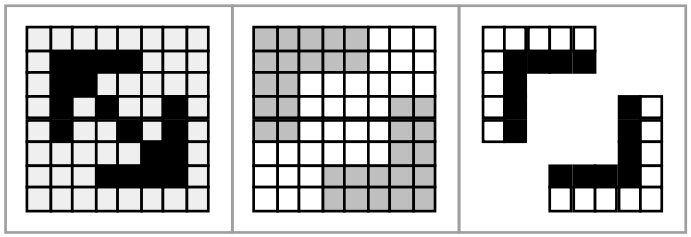
- **Les noms commencent par `XC_`, par exemple : `XC_crosshair`, `XC_pirate`.**
- **Les curseurs forment une police qui est chargée lors de la sélection d'un curseur. Regarder la police `cursor` par `xfd`.**
- **Exemple :**

```
#include <X11/cursorfont.h>
Cursor croix;

croix = XCreateFontCursor(dpy,XC_crosshair);
XDefineCursor(dpy,fen,croix);
```

- **Pour revenir au curseur de maman, faire `XUndefineCursor(dpy,fen)`.**

X-I

Curseurs (suite)



- **Les curseurs ont**
  - **une *forme* (c'est un bitmap),**
  - **un *masque* (aussi un bitmap),**
  - **un *hotspot* (point de référence, deux entiers)**
  - **et deux *couleurs* (valeurs de pixel).**
- **Le masque détermine la partie de la forme qui est affichée.**
- **Le point de référence détermine la position sensible du curseur.**
- **Pour un curseur standard, le masque est légèrement plus grand que la forme.**
- **Les couleurs par défaut sont le noir pour la couleur d'encre, et blanc pour la couleur de fond.**

X-I

Evènements

- **Chaque fenêtre peut être “sensibilisée” à une famille d’évènements, par `XSelectInput`.**
- **C’est le *client* qui choisit les évènements *pour* une fenêtre.**
- **Le *serveur* envoie les évènements aux *clients*, en indiquant la fenêtre concernée.**
- **Pour *sélectionner* les évènements à recevoir, on forme un masque, union de masques d’évènements.**
- **Il n’y a pas bijection entre évènements et masques.**
- **Plusieurs évènements peuvent se partager le même champ de la structure.**

```
typedef union _XEvent {
  int type;
   XAnyEvent xany;
   XKeyEvent xkey;
   XButtonEvent xbutton;
   XMotionEvent xmotion;
   XExposeEvent xexpose;
   XCreateWindowEvent xcreatewindow;
   XDestroyWindowEvent xdestroywindow;
   XUnmapEvent xunmap;
   XMapEvent xmap;
   XMapRequestEvent xmaprequest;
   XReparentEvent xreparent;
   ...
} XEvent;
```

X-I

Noms et masques d'évènements

- **Quelques évènements**

```
KeyPress
KeyRelease
ButtonPress
ButtonRelease
MotionNotify
EnterNotify
LeaveNotify
Expose
MapNotify
MapRequest
ReparentNotify
```

- **Quelques masques**

```
KeyPressMask
KeyReleaseMask
ButtonPressMask
ButtonReleaseMask
EnterWindowMask
LeaveWindowMask
PointerMotionMask
PointerMotionHintMask
Button1MotionMask
Button5MotionMask
ButtonMotionMask
KeymapStateMask
ExposureMask
ResizeRedirectMask
SubstructureNotifyMask
SubstructureRedirectMask
```

X-I

Evènements (suite)

- **Pour recevoir les clics de souris:**

```
XSelectInput(dpy,fen,
  ButtonPressMask);
```

- **Pour savoir si on entre ou sort d'une fenêtre:**

```
XSelectInput(dpy,fen,
  EnterWindowMask|
  LeaveWindowMask);
```

- **Pour recevoir l'évènement** `expose` **:**

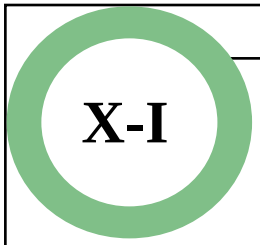
```
XSelectInput(dpy,fen,
  ExposureMask);
```

```
XEvent evmt;

XSelectInput(dpy,fille,
   ButtonPressMask|
   EnterWindowMask|
   LeaveWindowMask);
for (;;) {
   XNextEvent(dpy, &evmt);
   switch(evmt.type) {
      case ButtonPress :
         exit(0);
      case EnterNotify:
         XSetWindowBorderWidth(dpy,fille,4);
         break;
      case LeaveNotify:
         XSetWindowBorderWidth(dpy,fille,2)
         break;
   }
}
```

X-I

Propagation d'évènements

- **Si un évènement se produit dans une fenêtre F, mais que le type d'évènement n'est pas sélectionné pour F:**
  - **le serveur teste si la mère de F a sélectionné ce type d'évènements, puis sa grand-mère, etc.**
  - **si l'une des fenêtres a sélectionné l'évènement, l'évènement est envoyé;**
  - **sinon, l'évènement n'est pas envoyé.**
- **Pour inhiber la propagation des évènements d'un certain type, on utilise le masque `do_not_propagate`.**
- **Par défaut, les évènements sont propagés.**



Autres attributs

- **“Bit gravity” et “win gravity”**
- **“Backing store” et “save under”**
- **“Override redirect”**

X-I

“Bit gravity” et “win gravity”

- **Comportements par défaut:**
  - **Lorsqu’une fenêtre est déplacée, son contenu est conservé.**
  - **Lorsqu’une fenêtre est retaillée, son contenu est effacé.**
  - **Lorsqu’une partie d’une fenêtre est couverte par une autre fenêtre, le contenu de la partie couverte est perdu: après découverte, il n’est pas dessiné automatiquement, mais un évènement `expose` est engendré.**

- `bit_gravity` **et** `win_gravity` **spécifient le comportement du contenu (**`bit`**) respectivement d’une fenêtre (**`win`**) lors d’un changement de taille.**

- `bit_gravity` **pour le contenu. Les possibilités sont:**

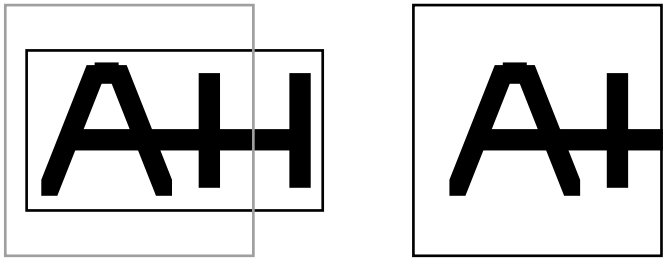
`ForgetGravity`	**(défaut) le contenu est oublié**
`StaticGravity`	**la place du contenu ne change pas par rapport à la racine.**

et:

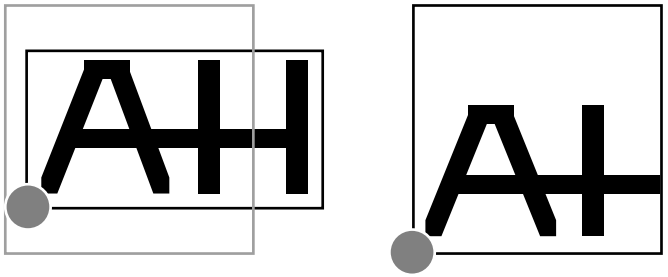
`NorthWestGravity`	`NorthGravity`	`NorthEastGravity`
`WestGravity`	`CenterGravity`	`EastGravity`
`SouthWestGravity`	`SouthGravity`	`SouthEastGravity`

Bit gravity : exemples

- StaticGravity



- SouthWestGravity



X-I

Win gravity

- `win_gravity` **pour une fenêtre. Les possibilités sont:**

  `UnmapGravity` **la fenêtre est enlevée.**
  `StaticGravity` **ne change pas par rapport à la mère.**
  et:

`NorthWestGravity`	`NorthGravity`	`NorthEastGravity`
`WestGravity`	`CenterGravity`	`EastGravity`
`SouthWestGravity`	`SouthGravity`	`SouthEastGravity`

- **Par défaut, c'est** `NorthWestGravity`**.**
- **Exemple: une fenêtre** `quitter` **doit rester dans le coin inférieur droit:**

```
XSetWindowAttributes attributs;

attributs.win_gravity = SouthEastGravity;

XChangeWindowAttributes(dpy,quitter,CWWinGravity,
  &attributs);
```

X-I

“Backing store” et “save under”

Backing store :

- **Restitution automatique d’une partie de fenêtre *redécouverte* après avoir été couverte par une autre.**
- **N’est pas disponible sur tous les serveurs (mais le serveur dit au client si c’est possible au moment de la connexion) .**
- **Si la taille d’une fenêtre change, le contenu est quand-même perdu si le “bit-gravity” est** `ForgetGravity`.
- **Trois possibilités**

`NotUseful`	**défaut**
`WhenMapped`	**lorsque affichée**
`Always`	**dans tous les cas (même icônifiée).**

- **Précisions possibles : masque de plans à sauvegarder, et valeur de pixel à utiliser dans les plans non spécifiés.**

Save under :

- **Symétrique du précédent : si** `True` **(**`False` **est le défaut), la partie obscurcie est sauvegardée et restituée après disparition de la fenêtre.**
- **Sert dans les fenêtres passagères , comme les menus.**

X-I

“Override redirect”

- **Sert à se passer du gestionnaire de fenêtres, dont on parlera plus loin (override = outrepasser).**
- **Indique un comportement *asocial* : une telle fenêtre est ignorée par le gestionnaire de fenêtres, donc**
  - **pas d’icônification,**
  - **pas de titre,**
  - **impossible de couvrir,...**
- **Utile pour les fenêtres affichées passagèrement, comme les menus déroulants et dynamiques (“popup”).**

X-I

Création d'une fenêtre: méthode générale

- **La fonction de création non simplifiée d'une fenêtre est rarement utilisée. Elle sert lorsque les attributs non modifiables:**
  - **Classe** `InputOnly` **ou** `InputOutput`
  - **Classe visuelle**
  - **Nombre de plans visuels**

  diffèrent de ceux de sa mère, et donc ne peuvent être hérités (avec `XCreateSimpleWindow`**, ces attributs sont hérités de la mère).**
- **Une fenêtre** `InputOnly` **n'a aucun attribut graphique. Elle sert uniquement comme cadre pour recevoir des évènements. Rarement utilisée.**
- **La *classe visuelle* indique les capacités d'affichage du display que l'on veut utiliser. Par exemple, faire uniquement du noir et blanc dans un display avec couleurs.**
- **Le nombre de plans visuels indique sur combien de bits les couleurs sont à coder. Par exemple, sur 3 bits (8 couleurs) dans un écran qui en permet 256.**
- **Attention, toutes les combinaisons ne sont pas toujours possibles (cela aussi, le serveur le dit au client au moment de la connexion; faire** `xdpyinfo`**); en général, sont possibles : 1 plan et les nombres divisant le nombre maximal de plan (p. ex.** `1, 2, 4, 8`**).**

X-I

Syntaxe

```
Window XCreateWindow(
  /* le display             (Display *)    */          dpy,
  /* fenetre mère           (Window)       */          mere,
  /* position de l'origine  (int)          */          x, y,
  /* largeur, hauteur       (unsigned int) */          largeur, hauteur,
  /* largeur bord           (unsigned int) */          largeur_bord,
  /* profondeur             (int)          */          prof,
  /* input ou inputoutput   (unsigned int) */          classe,
  /* structure "Visual"     (Visual *)     */          visuel,
  /* masque d'attributs     (unsigned long) */         masque,
  /* attributs              (XSetWindowAttributes *) */attributs);
```

- **Les seuls cas où cette fonction est indispensable sont**
  - **pour une fenêtre de classe** `InputOnly`**.**
  - **les fenêtres dont le** `background` **est indéfini.**

X-I

Fenêtres et gestionnaire de fenêtres

- **Rôle du gestionnaire**
- **Redirection**
- **Propriétés**
- **“Size Hints”**
- **Fenêtres passagères**

X-I

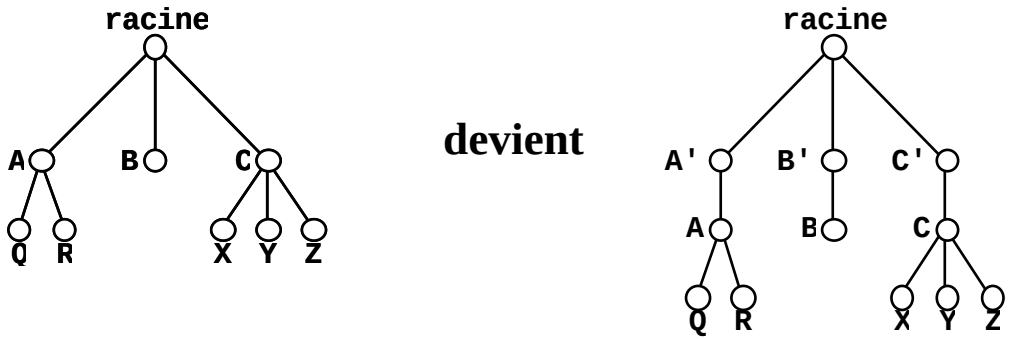
Rôle du gestionnaire de fenêtres

- **Le gestionnaire de fenêtres est un client particulier, mais basé sur les mécanismes X comme tous les autres clients.**
- **Son rôle est de gérer les fenêtres principales d'une application : il**
  - ***décore*** **les fenêtres principales par une *bordure*, une *barre de titre*, divers *boutons*, éventuellement un *menu*;**
  - ***déplace*** **et *redimensionne* les fenêtres;**
  - ***modifie*** **l'empilement des fenêtres;**
  - ***icônifie*** **les fenêtres.**
- **Pour ce faire, le gestionnaire sélectionne un évènement sur la fenêtre racine de l'écran.**
- **Certaines requêtes adressées au serveur, au lieu d'être exécutées, sont alors*redirigées*** **(sous forme d'évènements particuliers) vers le gestionnaire de fenêtres.**

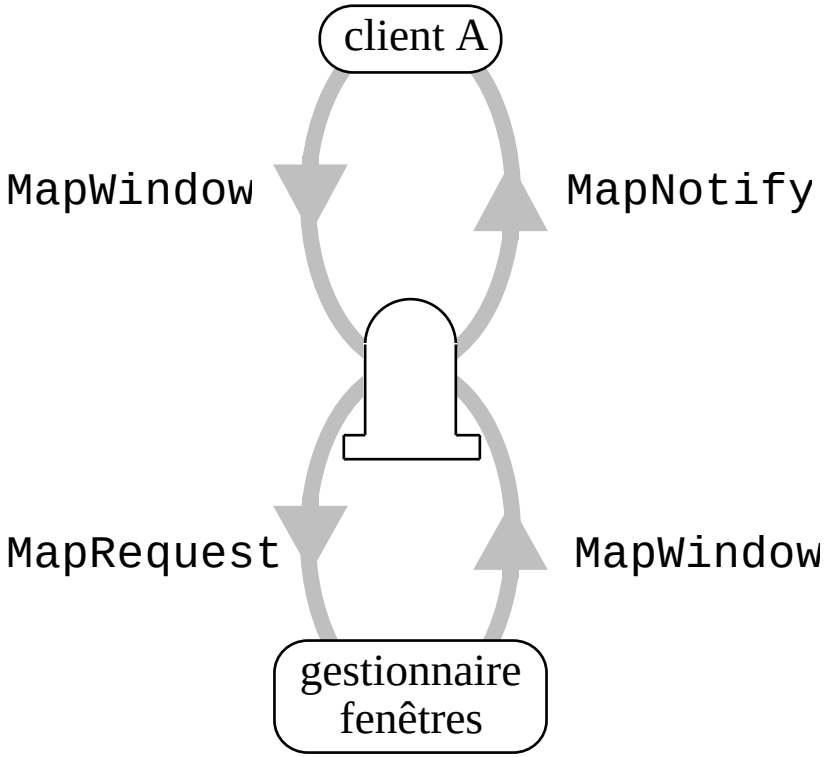
X-I

Redirection

- **Les requêtes redirigées (et donc prises en charge par le gestionnaire) sont**
  - `MapWindow` **affichage d'une fenêtre**
  - `ConfigureWindow` **redimensionnement de fenêtre**
  - `CirculateWindow` **changement dans l'empilement**
- **Pour un** `MapWindow` ***sur une fenêtre principale*** **F, le gestionnaire**
  - **crée une (ou plusdieurs) fenêtre de décoration F',**
  - **change la parenté : F' devient fille de la racine et F devient fille deF'.**
- **En présence d'un gestionnaire de fenêtres :**



X-I



- **Important : toutes les opérations réalisées par le gestionnaire de fenêtres le sont au moyen de fonctions `Xlib`, donc le gestionnaire est un client comme un autre.**
- **Le gestionnaire sélectionne un évènement par:**

```
Mask m;
...
m  |= SubstructureRedirectMask;
XSelect(dpy, racine, m);
```

- **Le *serveur*, recevant une requête d'affichage (`MapWindow`) du *client* A, regarde**
  - **si un *client* B a sélectionné `SubstructureRedirectMask` sur la mère;**
  - **si *client* A ≠ *client* B;**
  - **dans l'affirmative, envoie un évènement de type `MapRequest` au *client* B.**
- **Le *client* B en fait ce qu'il veut. Un gestionnaire crée une (ou plusieurs) fenêtre intermédiaire, et change la parenté par la fonction `Xlib`**

```
XReparentWindow(dpy, F, F',x,y).
```

X-I

Propriétés

- **La communication des clients avec le gestionnaire de fenêtres se fait par les *propriétés*.**
- **Une *propriété* est un couple (nom, valeur) associé à une fenêtre; elle peut être spécifiée par tout client, modfiée par tout client et consultée par tout client; les données sont stockées dans le serveur.**
- **Les *noms* des propriétés sont des *atomes*; ils ont des noms symboliques, dont les plus usuels sont prédéfinis: exemples**

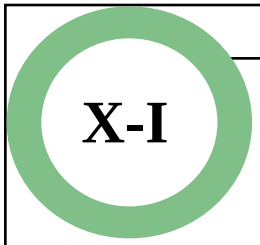
```
XA_CUT_BUFFER0          XA_CUT_BUFFER7
XA_RESOURCE_MANAGER     XA_WM_STATE
XA_WM_NAME              XA_WM_ICON_NAME
XA_WM_NORMAL_HINTS      XA_WM_TRANSIENT_FOR
```

- **Chaque propriété a un *type* (qui est lui-même un atome...). Par exemple**

```
XA_STRING, XA_WM_HINTS, XA_PIXMAP,...
```

- **Exemples: tous les atomes ci-dessus ont le type** `XA_STRING` **sauf**

`XA_WM_NORMAL_HINTS` **de type** `XA_WM_HINTS`
`XA_WM_STATE` **de type** `XA_WM_STATE.`



- **Quelques propriétés:**

XA_CUT_BUFFER0	XA_CUT_BUFFER7
XA_RESOURCE_MANAGER	XA_WM_STATE
XA_WM_NAME	XA_WM_ICON_NAME
XA_WM_NORMAL_HINTS	XA_WM_TRANSIENT_FOR

- **Les propriétés** XA_CUT_BUFFER **de la fenêtre racine servent au "couper-coller"** ***ancienne manière*** **entre clients comme** xterm**. Un autre mécanisme, basé sur les sélections, est plus moderne.**
- **La propriété** XA_RESOURCE_MANAGER **de la fenêtre racine contient les ressources transférées lors du lancement de la session (par** xrdb**).**
- **La propriété** XA_WM_STATE **est définie par le gestionnaire de fenêtres pour un gestionnaire de sessions. Sert aussi à distinguer les fenêtres principales de leur décoration. Etats possibles:**

NormalState	**fenêtre principale est visible**
IconicState	**la fenêtre pricipale est icônifiée et visible**
WithdrawnState	**ni l'une ni l'autre n'est visible.**

- **Les autres propriétés servent à donner des "indications" (hints) au gestionnaire de fenêtres.**

X-I

```
% xprop
WM_STATE(WM_STATE):
                window state: Normal
                icon window: 0x0
WM_NORMAL_HINTS(WM_SIZE_HINTS):
                user specified location: 100, 100
                program specified minimum aspect ratio: 1/1
                program specified maximum aspect ratio: 1/1
WM_ICON_NAME(STRING) = "shs2"
WM_NAME(STRING) = "SIZEHINTS2"
```

- **Donner un nom au client :**

```
XStoreName(dpy, fen, “SIZEHINTS2”)
```

- **Donner un nom à l’application icônifiée:**

```
XSetIconName(dpy, fen, “shs2”)
```

- **Donner ses souhaits quant à la géométrie :**

```
XSizeHints xsh;
...
XSetWMNormalHints(dpy, fen, &xsh);
```

X-I

Les “SizeHints”

```
typedef struct {
  long flags;                   /* marks which fields in this structure are defined */
  int x, y;                     /* obsolete for new window mgrs, but clients */
  int width, height;            /* should set so old wm's don't mess up */
  int min_width, min_height;
  int max_width, max_height;
  int width_inc, height_inc;
  struct {
    int x;                      /* numerator */
    int y;                      /* denominator */
  } min_aspect, max_aspect;
  int base_width, base_height;  /* added by ICCCM version 1 */
  int win_gravity;              /* added by ICCCM version 1 */
} XSizeHints;
```

- **NB: ICCCM= Inter-Client Communication Conventions Manual**

X-I

- **Exemple d'utilisation:**

```
void FixerPosition(void)
{
   XSizeHints xsh;

   xsh.flags = USPosition|PAspect;
   xsh.x = 100;
   xsh.y = 100;
   xsh.min_aspect.x=1;
   xsh.min_aspect.y=1;
   xsh.max_aspect.x=1;
   xsh.max_aspect.y=1;
   XSetWMNormalHints(dpy,fen, &xsh);
}
```

- **Arguments du masque:**

```
USPosition
USSize
PPosition
PSize
PMinSize
PMaxSize
PResizeInc
PAspect
PBaseSize
PWinGravity
```

X-I

Fenêtres passagères (“transient”)

- **Une fenêtre passagère ou éphémère, comme une fenêtre de dialogue, d’aide, ou certains menus, peut *apparaître en dehors* de la fenêtre “mère”, mais avec une décoration réduite. Or**
  - **Une telle fenêtre ne peut être fille de la fenêtre mère;**
  - **c’est donc aussi une fenêtre principale, mais “secondaire”.**
- **Pour ces fenêtres, on indique le caractère passager au gestionnaire par:**

`XSetTransientForHint`

- **Exemple :**

```
void Installer(void)
{
   fen = XCreateSimpleWindow( dpy, DefaultRootWindow(dpy),
      0,0,500,500, 6, Noir, Blanc);
   XStoreName(dpy,fen,"Transient");
   XSetIconName(dpy, fen, "transient");
   soeur = XCreateSimpleWindow( dpy, DefaultRootWindow(dpy),
      400,100,400,400, 1, Noir, Blanc);
   XStoreName(dpy,soeur,"Je m'iconifie avec ma soeur.");
   XSetTransientForHint(dpy,soeur,fen);
}
```

X-I

... et le gestionnaire

- **Le gestionnaire de fenêtres peut respecter ou non les souhaits des clients.**
- **L'utilisateur peut configurer son gestionnaire de fenêtres.**

- **twm (agit par variables)**

`défaut`	**fenêtre collée à la souris**
`RandomPlacement`	**placement aléatoire**
`UsePPosition`	**honore PPosition (si "on")**
`ClientBorderWidth`	**honore largeur bord**
`DecorateTransient`	**décore fenêtres passagères**

- **mwm (agit par ressources)**

`*clientAutoPlace`	`True`	**place fenêtres décalées**
	`False`	**honore client**
`*interactivePlacement`	`True`	**collée à la souris**
	`False`	**honore client**
`*clientDecoration`		**donne degré de décoration**
`*transientDecoration`		**idem pour passagères**

X-I

```
% xprop
WM_STATE(WM_STATE):
                window state: Normal
                icon window: 0x0
WM_NORMAL_HINTS(WM_SIZE_HINTS):
                user specified location: 100, 100
                program specified minimum aspect ratio: 1/1
                program specified maximum aspect ratio: 1/1
WM_ICON_NAME(STRING) = "shs2"
WM_NAME(STRING) = "SIZEHINTS2"
```

- **Donner un nom au client :**

  XStoreName(dpy, fen, “SIZEHINTS2”)

- **Donner un nom à l’application icônifiée:**

  XSetIconName(dpy, fen, “shs2”)

- **Donner ses souhaits quant à la géométrie :**

```
XSizeHints xsh;
...
XSetWMNormalHints(dpy, fen, &xsh);
```