

2. *Intrinsics et Toolkits : Généralité*

Xt Intrinsics

Xt Intrinsics et Toolkits Introduction

- o Les “Toolkits”
- o Widgets
- o Ressources
- o Réflexes

Jean Berstel - Institut Gaspard Monge, Université Marne-la-Vallée

1

Xt Intrinsics

“Toolkits”

Terminologie

- o X Toolkit
Nom collectif pour deux bibliothèques de procédures (Xt et Xaw). Par extension, tout ensemble de widgets basé sur Xt.
- o Xt ou Xt Intrinsics
Bibliothèque de procédures pour l'utilisation et pour la construction de widgets. Ecrite sur Xlib.
- o Xaw (*Athena Widget Set*)
Ensemble de widgets écrit au départ comme test de bon fonctionnement des Intrinsics;
— est maintenu et développé,
— est gratuit.
- o OSF/Motif
Ensemble de widgets développé par *Open Software Foundation*;
— un “look and feel” très précis;
— est payant.
- o Open Look
Ensemble de widgets créé par ATT/Sun.

Jean Berstel - Institut Gaspard Monge, Université Marne-la-Vallée

2

2. *Intrinsics et Toolkits : Généralité*

Xi Intrinsics

Concepts

- o *Widget* un composant d'interface configurable.
- o *Ressources* : les attributs d'une widget; déterminent les paramètres de son aspect visuel.
- o *Evènement sémantiques*. Déterminent la sensibilité du composant. Abstraction des évènements Xlib.
- o *Fonctions réflexes (callbacks)* Ecrites par le programmeur d'applications, et qui spécifient le comportement des widgets lorsqu'un évènement sémantique a lieu.
- o *Gestion géométrique* : placement des filles dans un conteneur.

Pour les Intrinsics tout est *modulable, paramétrisable, redéfinissable*.

Pour Motif ou un autre ensemble de widgets, le *look and feel* déterminent un certain nombre de paramètres.

3

Jean Berstel - Institut Gaspard Monge, Université Marne-la-Vallée

Xi Intrinsics

Widgets

- o Les *Intrinsics* définissent un nouvel objet, la *widget*.
 - l C'est un composant d'interface.
 - l Dans un programme, les widgets sont organisées en arbre.
- o Chaque widget comporte
 - l une *fenêtre* (au sens Xlib);
 - l des *ressources* (aspects géométriques);
 - l des *listes de réflexes* (pour le traitement des évènements).
- o Une *widget* est une *instance* d'une *classe* de widgets.
- o Les *ressources* d'une widget (par exemple `borderwidth`, `foreground`, `label`) déterminent *l'aspect visuel*.
 - l Les ressources disponibles sont spécifiées par la classe;
 - l toute instance a des valeurs de ressources par défaut;
 - l les valeurs sont configurables de multiples façons.
- o Les *réflexes* ("callbacks") sont des fonctions *programmeur* que l'on attache à une widget.
 - l Une fonction réflexe spécifie le comportement du programme en réaction à des évènement sémantiques (comme cliquer dans un bouton).
 - l Chaque classe définit ses propres catégories d'évènements sémantiques.

4

Jean Berstel - Institut Gaspard Monge, Université Marne-la-Vallée

2. *Intrinsics et Toolkits : Généralité*

Xt Intrinsics

Gadgets

Un *gadget* est une widget sans fenêtre propre.

- o Les gadgets améliorent en principe les performances. Ils sont très répandus dans Motif.
- o Particularités :
 - | un gadget n'a pas d'enfant;
 - | l'aspect graphique d'un gadget est hérité de sa mère : tout ce qui est dessiné dans un gadget l'est dans la fenêtre de sa mère.
- o Précisons :
 - | L'*arbre des widgets ou gadgets* d'une application est créé et maintenu dans le *client*
 - | L'*arbre des fenêtres* correspondant est créé et maintenu dans le serveur.
 - | Un gadget est une feuille de l'arbre des widgets du client, sans contre-partie dans l'arbre du serveur.
 - | Un gadget réduit donc les échanges avec le serveur.

Jean Berstel - Institut Gaspard Monge, Université Marne-la-Vallée

5

Xt Intrinsics

Xt Intrinsics et Xlib

- o Xt versus Xlib
 - | Les *Intrinsics* et les boîtes à widgets complètent *Xlib*, mais ne le remplacent pas.
 - | *Xt* offre des outils de haut niveau, alors que *Xlib* fournit toutes les fonctionnalités de base :
 - *Xlib* est dirigé par les événements;
 - *Xt* introduit des "événements symboliques" de plus haut niveau.
 - | *Xt* se concentre sur l'interface utilisateur, à un niveau d'abstraction supérieur, alors que *Xlib* est nécessaire pour le graphisme, et pour les réglages fins.
- o Le double aspect des *Intrinsics*
 - | Fournit les outils pour *utiliser* les widgets : création, modification, configuration, etc.
 - | Fournit les outils pour *construire* de nouvelles classes de widgets.
 - | Même si *Xt* est écrit en C, le mécanisme d'héritage est utilisé, et facilite l'écriture (sans la rendre simple !).

Jean Berstel - Institut Gaspard Monge, Université Marne-la-Vallée

6

2. Intrinsic et Toolkits : Généralité

Xt Intrinsic

Classes de widgets

Trois catégories de classes de widgets

- o Widgets "simples" ou *primitives* (boutons, étiquettes, textes) qui interagissent directement avec l'utilisateur.
 - ┆ accomplissent une fonction simple (bouton !);
 - ┆ leur comportement visuel doit refléter leur état.
- o Les classes de widgets "*conteneurs*" (barre de menu, panneau...) qui groupent d'autres widgets, organisées selon des contraintes de *placement*. Elles sont responsables:
 - ┆ de la disposition géométrique de leurs filles;
 - ┆ de les afficher ("mapper") ou au contraire de les cacher.

Pour ce faire, toute classe conteneur a un gestionnaire de géométrie (qui diffère d'une classe à l'autre).

- o Les classes de "*widgets shell*" (application, dialogue, alerte, aide) qui sont en interaction directe avec le gestionnaire de fenêtre.
 - ┆ Les widgets de ces classes interviennent comme interface; elles ont une fille unique.

Jean Berstel - Institut Gaspard Monge, Université Marne-la-Vallée

Xt Intrinsic

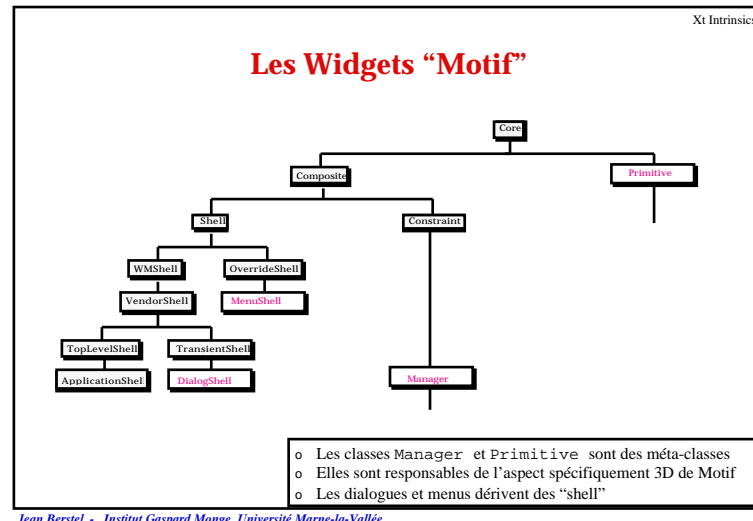
Les classes de widgets Xt Intrinsic

```
graph TD; Core[Core] --> Composite[Composite]; Composite --> Shell[Shell]; Composite --> Constraint[Constraint]; Shell --> WMShell[WMShell]; Shell --> OverrideShell[OverrideShell]; WMShell --> VendorShell[VendorShell]; VendorShell --> TopLevelShell[TopLevelShell]; VendorShell --> TransientShell[TransientShell]; TopLevelShell --> ApplicationShell[ApplicationShell];
```

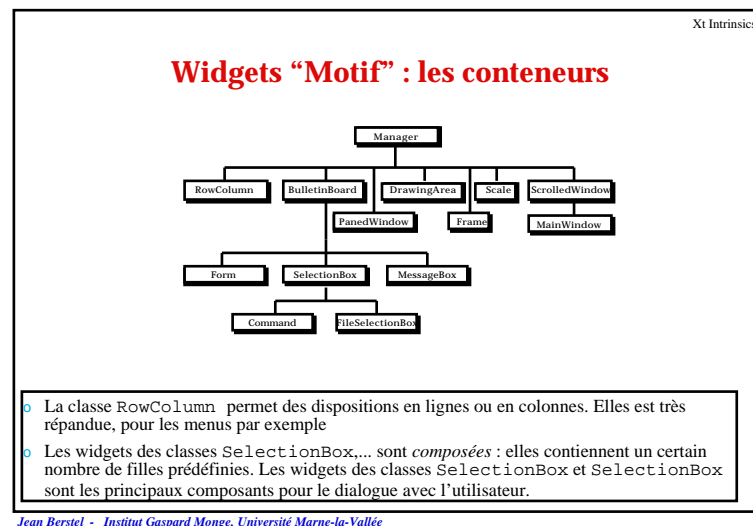
- o Xt définit des *méta-classes* de widgets qui ne sont pas censées être instanciées.
- o Les classes directement utilisées sont ApplicationShell, TransientShell et OverrideShell.
- o Malgré leurs noms, les widgets "shell" ne sont pas des *shell* au sens de Unix.
- o Chaque ensemble de widgets "accroche" ses classes à cet arbre.

Jean Berstel - Institut Gaspard Monge, Université Marne-la-Vallée

2. Intrinsic et Toolkits : Généralité



Jean Berstel - Institut Gaspard Monge, Université Marne-la-Vallée



Jean Berstel - Institut Gaspard Monge, Université Marne-la-Vallée

2. Intrinsic et Toolkits : Généralité

Xi Intrinsic

Widgets "Motif" : les primitifs

```
graph TD
    Primitive --> Text
    Primitive --> List
    Primitive --> ScrollBar
    Primitive --> Label
    Primitive --> ArrowButton
    Primitive --> Separator
    Text --> TextField
    Label --> DrawnButton
    Label --> PushButton
    Label --> CascadeButton
    PushButton --> ToggleButton
```

- o TextField est une version allégée de Text, pour des textes qui tiennent sur une ligne.
- o Les boutons ont des variantes:
 - ! PushButton est un bouton ordinaire.
 - ! ToggleButton est un bouton faisant partie d'un groupe.
 - ! CascadeButton sert comme point d'ancrage d'un menu.
 - ! DrawnButton est comme PushButton, mais a des possibilités graphiques supplémentaires.
 - ! ArrowButton est comme PushButton, mais affiche une flèche...

11

Jean Berstel - Institut Gaspard Monge, Université Marne-la-Vallée

Xi Intrinsic

... et les gadgets

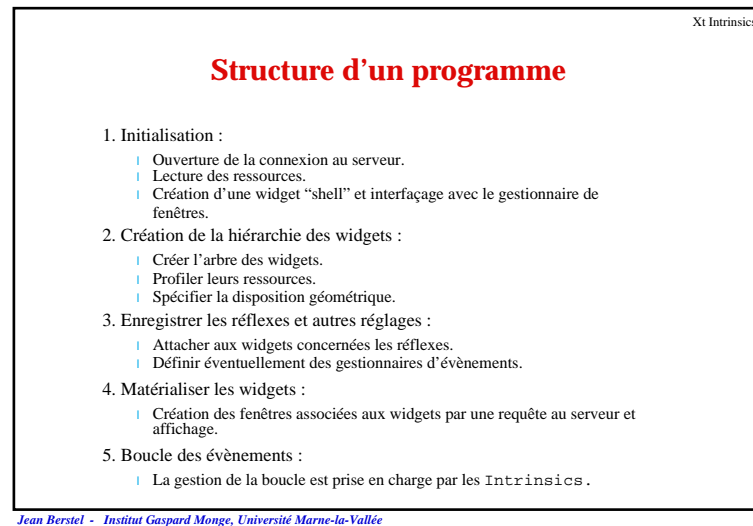
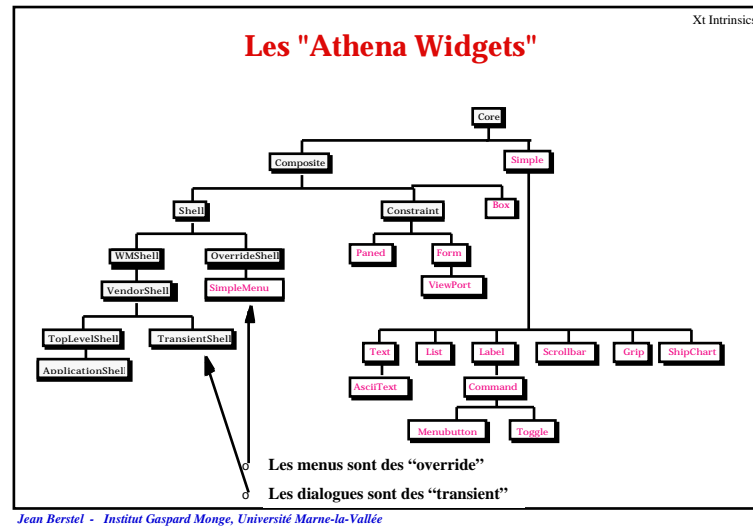
```
graph TD
    Gadget --> ArrowButton
    Gadget --> Separator
    Gadget --> Label
    Gadget --> CascadeButton
    Gadget --> PushButton
    Gadget --> ToggleButton
    Gadget --| Core
    Core --| unnamed
    unnamed --| RectObj
    RectObj --| Object
```

- o Les classes de gadgets ne peuvent hériter des classes de widgets.
- o Les classes de gadgets sont accrochées au-dessus de la classe Core.
- o La classe Core a une fenêtre, donc des ressources du type couleur, profondeur, etc.
- o La classe unnamed est "réservée à une utilisation ultérieure".
- o La classe RectObj est limitée à des rectangles, donc connaît largeur, hauteur, épaisseur du bord..
- o La classe Object n'a pas de ressources.

12

Jean Berstel - Institut Gaspard Monge, Université Marne-la-Vallée

2. Intrinsic et Toolkits : Généralité



2. Intrinsic et Toolkits : Généralité

Xt Intrinsic

Exemple : le programme

```
#include <Xm/PushButton.h>
#include <Xm/RowColumn.h>
#include <Xm/Label.h>

main(int argc, char **argv)
{
    Widget racine, boite, etiquette, bouton, fin;
    XtAppContext app;

    racine = XtVaAppInitialize(&app, "Bonjour", NULL, 0, &argc, argv, NULL, NULL);
    boite = XtVaCreateManagedWidget("boite", xmRowColumnWidgetClass, racine,
    NULL);
    etiquette = XtVaCreateManagedWidget("titre", xmLabelWidgetClass, boite, NULL);
    bouton = XtVaCreateManagedWidget("bouton", xmPushButtonWidgetClass, boite,
    NULL);
    fin = XtVaCreateManagedWidget("quit", xmPushButtonWidgetClass, boite,
    NULL);

    XtAddCallback(fin, XmNactivateCallback, salut, NULL);

    XtRealizeWidget(racine);
    XtAppMainLoop(app);
}
```

Initialisation
Création
Réflexe
Réaliser
Boucle

Jean Berstel - Institut Gaspard Monge, Université Marne-la-Vallée

15

Xt Intrinsic

Exemple : le fichier de ressources

- o Le fichier de ressources n'est pas indispensable
- o Il contient les ressources personnalisables par l'utilisateur final
- o Il est très commode pendant tout la phase de développement car
 - | c'est un fichier texte
 - | les ressources sont faciles à spécifier
 - | le changement d'une ressource ne demande pas de recompilation
- o Son nom est donné en deuxième argument dans XtVaAppInitialize

```
!
! Bonjour : fichier de ressources pour bonjour
!
*titre.labelString :Bonjour, monde !
*titre.borderWidth :0
*bouton.labelString :Cliquer ici ne sert a rien
*quit.labelString :Cliquer ici
```

Jean Berstel - Institut Gaspard Monge, Université Marne-la-Vallée

16

2. Intrinsic et Toolkits : Généralité

Xt Intrinsic

Exemple : la fonction réflexe

- o Une *fonction réflexe* définit un comportement d'une widget
- o Elle a un prototype prédéfini:
 - l w est la Widget où l'évènement s'est produit
 - l clientData sont les données passées à l'enregistrement
 - l callData sont les données fournies par Motif, contenant le détail de l'évènement.

```
void salut(Widget w, XtPointer clientData, XtPointer callData)
{
    exit(0);
}
```

17

Jean Berstel - Institut Gaspard Monge, Université Marne-la-Vallée

Xt Intrinsic

Un exemple (Athena)

```
main(int argc, char **argv)
{
    Widget racine, boite, etiquette, bouton, fin;
    XtAppContext app;

    racine = XtVaAppInitialize(&app, "Bonjour", ← m Initialisation
                             NULL, 0, &argc, argv, NULL, NULL);
    boite = XtVaCreateManagedWidget("boite", ← m Création
                                    boxWidgetClass, racine, ← m Ressources
                                    NULL);
    etiquette = XtVaCreateManagedWidget("titre", ← m Ressources
                                        labelWidgetClass, boite,
                                        XtNlabel, "Bonjour, monde !",
                                        XtNborderWidth, 0, NULL);
    bouton = XtVaCreateManagedWidget("bouton", ← m Ressources
                                     commandWidgetClass,
                                     boite, XtNlabel, "Cliquer ici ne sert a rien", NULL);
    fin = XtVaCreateManagedWidget("quit", commandWidgetClass, ← m Ressources
                                   boite, XtNlabel, "Cliquer ici", NULL);
    XtAddCallback(fin, XtNcallback, salut, NULL); ← m Réflexe

    XtRealizeWidget(racine); ← m Réaliser
    XtAppMainLoop(app); ← m Boucle
}
```

18

Jean Berstel - Institut Gaspard Monge, Université Marne-la-Vallée

2. Intrinsic et Toolkits : Généralité

Xt Intrinsic

Initialisation

```
racine = XtVaAppInitialize(  
    &app,          /* Contexte d'application */  
    "Bonjour",    /* Nom de la classe de l'application */  
    NULL, 0,      /* Liste d'options */  
    &argc, argv,  /* Ligne de commande */  
    NULL,         /* Ressources par défaut*/  
    NULL);       /* Liste de ressources, terminée par NULL */
```

- o `XtVaAppInitialize` est une fonction *à nombre variable* d'arguments.
- o **Contexte d'application** : pointeur vers une structure opaque dans laquelle Xt gère les données associées à l'application.
 - | Ne sert que pour être passée à `XtAppMainLoop`.
 - | Sert à gérer l'espace d'adressage des processus.
- o **Classe** c'est une chaîne de caractères qui sert :
 - | dans la spécifications des ressources, comme les classes de widgets;
 - | comme *nom* pour le fichier des ressources de l'application.
- o **Convention d'écriture**. Le nom de la classe est :
 - | le nom de l'application, avec première lettre en majuscule : `bonjour` -> `Bonjour`
 - | si l'initiale est x, les deux premières en majuscule : `xlatex` -> `XLatex`

Jean Berstel - Institut Gaspard Monge, Université Marne-la-Vallée

19

Xt Intrinsic

`XtVaAppInitialize` (suite)

- o **Liste d'options** et leur nombre : chaque application peut définir une liste d'options de lignes de commande qui lui est propre. Elle est transmise ici.
- o `&argc, argv` : la ligne de commande est analysée.
 - | Chaque application Xt peut contenir les options standard Xt et des options de la liste précédente.
 - | Chaque option reconnue est enlevée : après analyse réussie, `argv` est réduit au nom de l'application.
- o Ressources par défaut ("fallback resources") : chaîne de caractères contenant des ressources.
- o Liste de ressources (nom, valeur) pour la widget racine. Terminée par `NULL`.
- o `XtVaAppInitialize` retourne une widget de la classe `ApplicationShell`.

Jean Berstel - Institut Gaspard Monge, Université Marne-la-Vallée

20

2. Intrinsic et Toolkits : Généralité

Xt Intrinsic

Variations

```
racine = XtVaAppInitialize(&app, "Bonjour", NULL, 0, &argc, argv, NULL, NULL);
racine = XtAppInitialize(&app, "Bonjour", NULL, 0, &argc, argv, NULL, NULL, 0);
racine = XtInitialize(argv[0], "Bonjour", NULL, 0, &argc, argv);
```

```
Widget XtVaAppInitialize(
    XtAppContext * app_contexte_retour,
    String        nom_de_classe,
    XrmOptionDescList options,
    Cardinal      num_options,
    Cardinal *    argc_in_out,
    String *      argv_in_out,
    String *      ressources_secours,
    ..., NULL);
```

```
Widget XtAppInitialize(
    XtAppContext * app_contexte_retour,
    String        nom_de_classe,
    XrmOptionDescList options,
    Cardinal      num_options,
    Cardinal *    argc_in_out,
    String *      argv_in_out,
    String *      ressources_secours,
    ArgList      args,
    Cardinal      num_args);
```

```
Widget XtInitialize(
    String        nom_application,
    String        nom_de_classe,
    XrmOptionDescList options,
    Cardinal      num_options,
    Cardinal *    argc_in_out,
    String *      argv_in_out);
```

21

Jean Berstel - Institut Gaspard Monge, Université Marne-la-Vallée

Xt Intrinsic

Ressources

Une *ressource* est un attribut configurable d'une widget.

- o On la spécifie par un couple (nom, valeur).
 - | *nom* désigne un attribut de la widget;
 - | *valeur* est la valeur de cette attribut pour cette instance.
- o Chaque classe définit ses ressources, et hérite des ressources de ses superclasses.
- o Le *nom* d'une ressource est une **chaîne de caractères**.
- o Chaque ressource a un **nom symbolique** :
 - | il permet de contourner des fautes de frappe.
 - | il est obtenu en préfixant la chaîne par **XtN** (resp. **XmN** pour Motif):

Exemples

- o "background", "x", "width"
- o XtNbackground, XtNx, XtNwidth
- o XmNbackground, XmNx, XmNwidth

22

Jean Berstel - Institut Gaspard Monge, Université Marne-la-Vallée

2. *Intrinsics et Toolkits : Généralité*

Xt Intrinsics

- o Chaque ressource, en plus du nom et de la valeur, possède:
 - | une *classe* (ne pas confondre avec les classes de widgets)
 - | un *type interne* ("representation type")
 - | un *type C* correspondant
- o La *classe* permet de regrouper des ressources similaires pour les spécifier par leur nom de classe.
- o Le type interne est utilisé pour la conversion entre représentations.
- o Le type interne et le type C servent à l'allocation consistante de place.
- o Les Intrinsics fournissent, de plus, les outils pour
 - | définir de nouvelles ressources
 - | convertir des ressources d'un type vers un autre (par exemple : un nom de couleur en valeur de pixel)
 - | d'écrire des convertisseurs de ressources.

Jean Berstel - Institut Gaspard Monge, Université Marne-la-Vallée

23

Xt Intrinsics

Création de widgets

```
etiquette = XtVaCreateManagedWidget(  
    "titre", /* nom de la widget */  
    xmLabelWidgetClass, /* sa classe */  
    boite, /* mère de la widget */  
    XmNborderWidth, 0, /* Liste de ressources */  
    NULL); /* terminée par NULL */
```

- o *Nom* de la widget : chaîne de caractères utilisable dans les fichiers de ressources.
- o *Création* d'une widget :
 - | allocation de la place (dans le client);
 - | initialisation des ressources;
 - | insertion dans la liste des filles de sa mère.
- o *Gestion* ("manage") :
 - | calcul de la forme géométrique de la widget
 - | peut induire une modification de la géométrie de la mère, etc.
 - | affichage si mère affiché.
- o Tout ce processus se fait chez le client, sans contact avec le serveur.

Jean Berstel - Institut Gaspard Monge, Université Marne-la-Vallée

24

2. *Intrinsics et Toolkits : Généralité*

Xt Intrinsics

Variations

<code>XtVaCreateWidget</code>	création sans gestion
<code>XtManageChild</code>	gestion séparée
<code>XtUnmanageChild</code>	ôter des filles gérées
<code>XtManageChildren</code>	gestion d'une liste de filles

25

Jean Berstel - Institut Gaspard Monge, Université Marne-la-Vallée

Xt Intrinsics

“Réaliser”

`XtRealizeWidget(racine);`

- *Réaliser* : création de la fenêtre associée à la widget, dans le serveur.
 - ┆ Création récursive pour chacune des descendantes, s'il y en a
- Si la ressource `XtMappedWhenManaged` d'une widget est `True` (défaut), la fenêtre est aussi “mappée”, et donc visible selon les règles `Xlib`.

26

Jean Berstel - Institut Gaspard Monge, Université Marne-la-Vallée

2. Intrinsic et Toolkits : Généralité

Xt Intrinsic

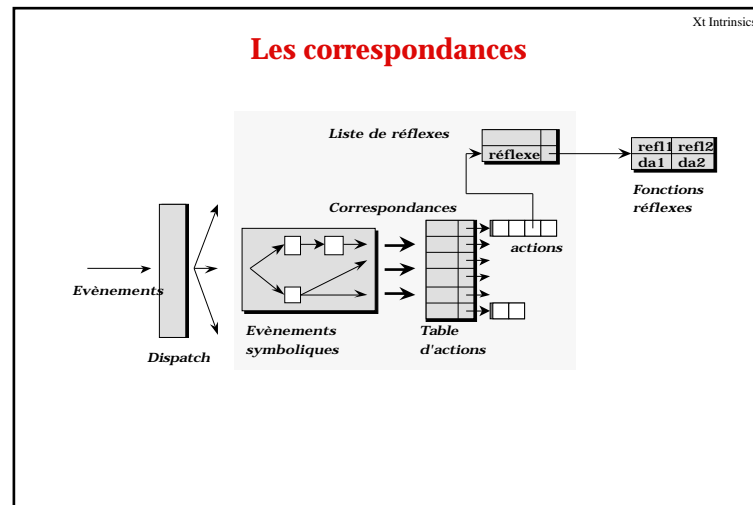
La boucle principale

```
XtAppMainLoop (app) ;
```

- La gestion des évènements est entièrement pris en charge par Xt.
- La fonction `XtAppMainLoop` :
 - | lit le prochain évènement : `XtAppNextEvent`;
 - | l'envoi à la procédure appropriée: `XtDispatchEvent`.
- Cette fonction utilise le champ `window` de l'évènement pour chercher une widget qui possède cette fenêtre.
 - | Les clients qui ont sélectionné les bons évènements sur les (fenêtres des) widgets les reçoivent, et les traitent;
 - | Cette gestion est "automatique" pour les widgets qui ont des *tables de correspondances*.
 - | Elle est à un niveau d'abstraction supérieur à la gestion des évènements bruts.

Jean Berstel - Institut Gaspard Monge, Université Marne-la-Vallée

27



Jean Berstel - Institut Gaspard Monge, Université Marne-la-Vallée

28

2. Intrinsic et Toolkits : Généralité

Xt Intrinsic

Les listes de fonctions réflexes

```
void salut(Widget w, XtPointer clientData, XtPointer callData)
{
    exit(0);
}
```

- o Chaque classe de widgets possède des **listes de fonction réflexes**. Une telle liste
 - | est identifiée par un **nom**, p. ex: `XmNactivateCallback` (Motif).
 - | contient une liste, initialement vide, mais modifiable à volonté, de fonctions (*fonction réflexes* ou "callbacks").
 - | les fonctions enregistrées (par `Xt.AddCallback`) sont **exécutées** lorsque l'évènement sémantique correspondant a eu lieu.
- o **Syntaxe** des fonctions réflexes:

```
void nom (Widget w, XtPointer clientData, XtPointer callData)
```

 - | La widget `w` est celle où l'évènement a eu lieu;
 - | La `clientData` est une donnée enregistrée avec la fonction réflexe;
 - | Les `callData` décrivent (en Motif) les *circonstances* de l'évènement.

Jean Berstel - Institut Gaspard Monge, Université Marne-la-Vallée

29

Xt Intrinsic

Les ressources

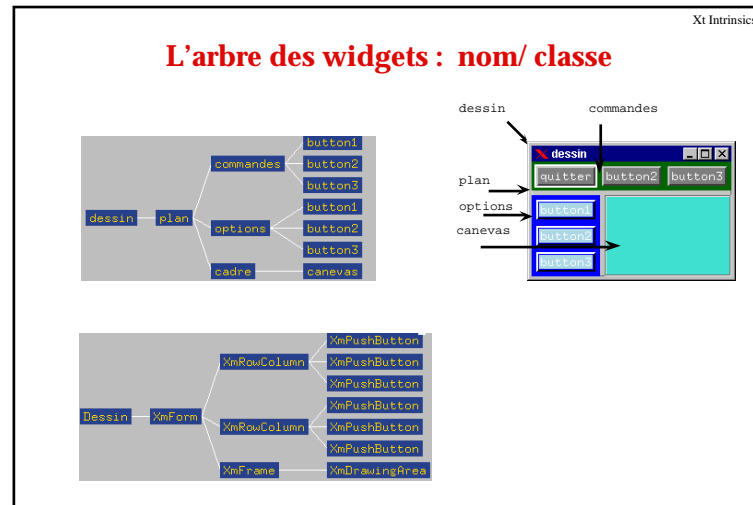
Les ressources d'une application peuvent être spécifiées à trois niveaux:

- o Les **fichiers** de ressources, dont l'ordre d'inclusion est fixe, et la localisation est régie par des variables d'environnement.
- o La **ligne de commande** qui accepte des options standard et des options définies dans l'application.
- o Le **programme** lui-même, soit à la création des widgets, soit par modification ultérieure (codage "en dur").
- o Dans les deux premiers cas:
 - | les ressources et leurs valeurs sont données sous forme de *chaînes de caractères*
 - | Xt les traduit en valeurs internes à l'aide de *convertisseurs de types* prédéfinis.

Jean Berstel - Institut Gaspard Monge, Université Marne-la-Vallée

30

2. Intrinsics et Toolkits : Généralité



Jean Berstel - Institut Gaspard Monge, Université Marne-la-Vallée

Xt Intrinsics

Programme (extrait)

```

void main ( int argc, char* argv[] ) {
    Widget shell, panneau;
    XtAppContext app;
    shell = XtAppInitialize (&app, "Dessin", NULL, 0, &argc, argv, NULL, NULL, 0 );
    panneau = XtVaCreateManagedWidget ("plan", xmFormWidgetClass, shell, NULL);
    Installer(panneau);
    XtRealizeWidget ( shell );
    XtAppMainLoop ( app );
}

void Installer(Widget mere){
    Widget canevas, cadre, commandes, options, b;

    commandes = XtVaCreateManagedWidget ("commandes", xmRowColumnWidgetClass, mere, NULL );
    options = XtVaCreateManagedWidget ("options", xmRowColumnWidgetClass, mere, NULL );
    cadre = XtVaCreateManagedWidget ("cadre", xmFrameWidgetClass, mere, NULL );
    canevas = XtVaCreateManagedWidget ("canevas", xmDrawingAreaWidgetClass, cadre, NULL );
    b = XtVaCreateManagedWidget ( "button1", xmPushButtonWidgetClass, commandes, NULL);
    XtAddCallback(b, XmNactivateCallback, quitter, NULL);
    XtVaCreateManagedWidget ( "button2", xmPushButtonWidgetClass, commandes, NULL);
    XtVaCreateManagedWidget ( "button3", xmPushButtonWidgetClass, commandes, NULL);
    XtVaCreateManagedWidget ( "button1", xmPushButtonWidgetClass, options, NULL);
    XtVaCreateManagedWidget ( "button2", xmPushButtonWidgetClass, options, NULL);
    XtVaCreateManagedWidget ( "button3", xmPushButtonWidgetClass, options, NULL);
}
    
```

32

Jean Berstel - Institut Gaspard Monge, Université Marne-la-Vallée

2. Intrinsic et Toolkits : Généralité

Xi Intrinsic

```
*fontList: 8x13
*foreground: white

*commandes.orientation : XmHORIZONTAL
*commandes.topAttachment : XmATTACH_FORM
*commandes.rightAttachment : XmATTACH_FORM
*commandes.leftAttachment : XmATTACH_FORM
*commandes.bottomAttachment : XmATTACH_NONE
*commandes*background : DarkGreen
*commandes*XmPushButton.background: grey50
Dessin.plan.commandes.bouton1.labelString: quitter

*options.topAttachment : XmATTACH_WIDGET
*options.topWidget : commandes
*options.rightAttachment : XmATTACH_NONE
*options.leftAttachment : XmATTACH_FORM
*options.bottomAttachment : XmATTACH_FORM
*options*background : Blue
*options*XmPushButton.background: LightBlue
*options.topOffset : 5

*cadre.topAttachment : XmATTACH_WIDGET
*cadre.topWidget : commandes
*cadre.rightAttachment : XmATTACH_FORM
*cadre.leftAttachment : XmATTACH_WIDGET
*cadre.leftWidget : options
*cadre.bottomAttachment : XmATTACH_FORM
*cadre.topOffset : 5
*cadre.leftOffset : 5
*XmDrawingArea.background : Turquoise
```

Fichier
de
ressources

Jean Berstel - Institut Gaspard Monge, Université Marne-la-Vallée

33

Xi Intrinsic

Chemin d'accès

- o Le **chemin d'accès** est la suite des noms des widgets ou de leur classe, menant de la racine à une feuille
- o La feuille est le nom de la ressource (ou de sa classe).
- o La spécification d'une ressource dans un fichier se fait par

chemin d'accès : valeur
- o **Chemins d'accès par noms de classe**
Dessin.XmForm.XmRowColumn.XmPushButton.Foreground
représente les six boutons
- o **Chemins d'accès par noms de widget**
dessin.plan.commandes.bouton1.foreground
dessin.pplan.XmRowColumn.bouton1.foreground
- o **Jokers:**
*bouton1.foreground
Dessin.plan.?.bouton1.foreground

Jean Berstel - Institut Gaspard Monge, Université Marne-la-Vallée

34

2. Intrinsic et Toolkits : Généralité

Xi Intrinsic

Règles

Règles appliquées, dans l'ordre, à chaque niveau des chemins:

Règle 1. Un *nom* l'emporte sur une *classe* qui l'emporte sur un '?' qui l'emporte sur une étoile :

nom > classe > ? > *

Règle 2. Un champ précédé d'un point l'emporte sur le même champ précédé d'une étoile :

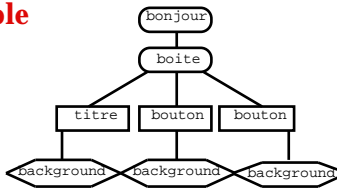
.bouton > *bouton

Jean Berstel - Institut Gaspard Monge, Université Marne-la-Vallée

37

Xi Intrinsic

Exemple



- o Fichier ressources :
- o Ventilation par champs :

	bonjour	boite	bouton
*background	: rouge	*	*
*bouton.background	: vert	*	*
*XmPushButton.background	: bleu	*	*
*XmRowColumn*background	: jaune	*	XmRowColumn
*XmRowColumn.XmPushButton.background	: rose	*	XmRowColumn XmPushButton

- o Analyse :le fond est rose
 - le deuxième champ sélectionne les deux dernières ressources
 - le troisième champ sélectionne la dernière.

Jean Berstel - Institut Gaspard Monge, Université Marne-la-Vallée

38

2. Intrinsic et Toolkits : Généralité

Xi Intrinsic

Exemple (xmh)

- Fichier ressources :

(A)	xmh*Panel*activeForeground
(B)	*inclure.Foreground
(C)	xmh.toc*Command*activeForeground
(D)	xmh.toc*?.Foreground
(E)	xmh.toc*Command.activeForeground
- Ventilation par champs :

nom	xmh	toc	cop	inclure	activeForeground
classe	Xmh	Panel	Box	Command	Foreground
(A)	xmh	*	Panel	*	activeForeground
(B)	*			inclure	. Foreground
(C)	xmh	.	toc	*	Command * activeForeground
(D)	xmh	.	toc	*	? . Foreground
(E)	xmh	.	toc	*	Command . activeForeground
- Analyse :
 - niveau 1 élimine B
 - niveau 2 élimine A
 - niveau 3 n'élimine rien
 - niveau 4 élimine D
 - niveau 4-5 élimine C par règle 2

Jean Berstel - Institut Gaspard Monge, Université Marne-la-Vallée

39

Xi Intrinsic

Options sur la ligne de commande

<pre>-background *background -bordercolor *borderColor -borderwidth .borderWidth -display .display -foreground *foreground -font *font -geometry .geometry -iconic .iconic -name .name -reverse .reversVideo -selectionTimeout .selectionTimeout -synchronous .synchronous +synchronous .synchronous -title .title -xrm .title -xnllanguage xnllanguage</pre>	<p>le display</p> <p>au départ</p> <p>nom de l'instance</p> <p>inversion fore et back</p> <p>attente max. transfert sélection</p> <p>pour déboguer</p> <p>titre de la fenêtre</p> <p>utile</p> <p>change LANG</p>	<ul style="list-style-type: none"> o Equivalents : -background -bg -bordercolor -bd -foreground -fg -font -fn -reverse -rv +rv
---	---	---

- o **Equivalents :**
- test -background blue
- test -xrm "*background : blue"

Jean Berstel - Institut Gaspard Monge, Université Marne-la-Vallée

40

2. Intrinsic et Toolkits : Généralité

Xt Intrinsic

Ressources “en dur”, que choisir ?

- o Spécifier les ressources **dans un fichier** est
 - | *simple*, car la conversion de types est automatique,
 - | *commode* durant la mise au point, car pas de recompilation,
 - | *agréable* pour l'utilisateur final qui peut personnaliser l'application.
- o Spécifier les ressources dans le programme (“en dur”) est
 - | *nécessaire* à la création de certaines widgets (menus, textes par exemple)
 - | *indispensable* lorsque l'on veut imposer un comportement.
- o Il est recommandé de mettre le *maximum* de ressources dans le fichier, donc un *minimum* en dur.

Jean Berstel - Institut Gaspard Monge, Université Marne-la-Vallée

41

Xt Intrinsic

Structures pour les ressources en dur

- o Dans un programme, on donne un couple (nom, valeur), de type `Arg`.
- o Le type `XtArgVal` est “implementation specific”. Contient un `long` et tout pointeur, en particulier `XtPointer`.
- o Dans les fonctions à *nombre variable d'arguments*, énumérer les composantes des couples, terminer par `NULL`.
- o Dans les fonctions à *nombre constant d'arguments* (comme certaines fonctions Motif), on passe un *tableau d'Args* et leur *nombre*.

```
typedef struct {
    String name;
    XtArgVal value;
} Arg, *ArgList;

typedef void* XtPointer;

Arg a[];
int n;
...
n = 0;
XtSetArg(a[n], XtNx, 10);n++;
XtSetArg(a[n], XtNy, 10);n++;
XtSetArg(a[n], XtNlabel, "Bonjour, monde!");n++;
```

Jean Berstel - Institut Gaspard Monge, Université Marne-la-Vallée

42

2. Intrinsic et Toolkits : Généralité

Xt Intrinsic

Gérer les ressources

- o **Modifier** la valeur de ressources d'une widget :

```
XtVaSetValues(widget, nom, valeur, ..., NULL)
XtSetValues(widget, args, nargs);
```
- o **Récupérer** les valeurs de ressources d'une widget :

```
XtVaGetValues(widget, nom, &valeur, ..., NULL)
```
- o Exemples :

```
XtVaSetValues(texte,
  XmNrows,      10,
  XmNcolumns,   80,
  XmNeditMode,  XmMULTI_LINE_EDIT,
  XmNautoShowCursorPosition, TRUE, NULL);

XtVaGetValues(texte,
  XmNrows,      &lignes,
  XmNcolumns,   &colonnes, NULL);
```

43

Jean Berstel - Institut Gaspard Monge, Université Marne-la-Vallée

Xt Intrinsic

Conversion de ressources

- o Exemple : comment spécifier une couleur par son nom (lightgray) dans un programme ?
 - | XmNbackground demande une valeur de pixel (Pixel).
 - | Il faut *convertir* la chaîne de caractères lightgray en un Pixel.

```
XtVaSetValues(w, XmNbackground, NumCoul("red"), NULL);
```
- o Voici une procédure NumCoul de conversion :

```
Pixel NumCoul(String nom)
{
  Display *dpy = XtDisplay(racine);
  Colormap cmap = DefaultColormapOfScreen(XtScreen(racine));
  XColor col, x;
  if (!XAllocNamedColor(dpy, cmap, nom, &col, &x)) {
    char tamp[32];
    sprintf(tamp, "Peux pas allouer le %s", nom);
    XtWarning(tamp);
    return(0);
  }
  return(col.pixel);
}
```

44

Jean Berstel - Institut Gaspard Monge, Université Marne-la-Vallée

2. Intrinsic et Toolkits : Généralité

Xt Intrinsic

Convertisseurs Intrinsic

- o Les Intrinsic (et Motif) fournissent des *convertisseurs* de types.
- o Ces convertisseurs sont utilisés implicitement dans les fichiers de ressources.
- o La conversion dans un programme est sollicitée par la spécification
XtVaTypedArg
- o XtVaTypedArg prend 4 arguments, qui suivent dans la liste des spécifications :
 - XmNforeground » le *nom* de la ressource à affecter;
 - XmRString » le *type interne* ("representation type") de la valeur (en général XmRString);
 - "Red" » la *valeur* à affecter, dans le type C correspondant au type interne (en général char *);
 - 4 » la *taille* de la valeur (y compris le caractère nul final).

```
echelle = XtVaCreateManagedWidget("rouge", xmScaleWidgetClass, boite,  
    XtVaTypedArg, XmNtitleString, XmRString, "rouge", 6,  
    XtVaTypedArg, XmNforeground, XmRString, "Red", 4,  
    NULL);
```