Boîtes à Outils

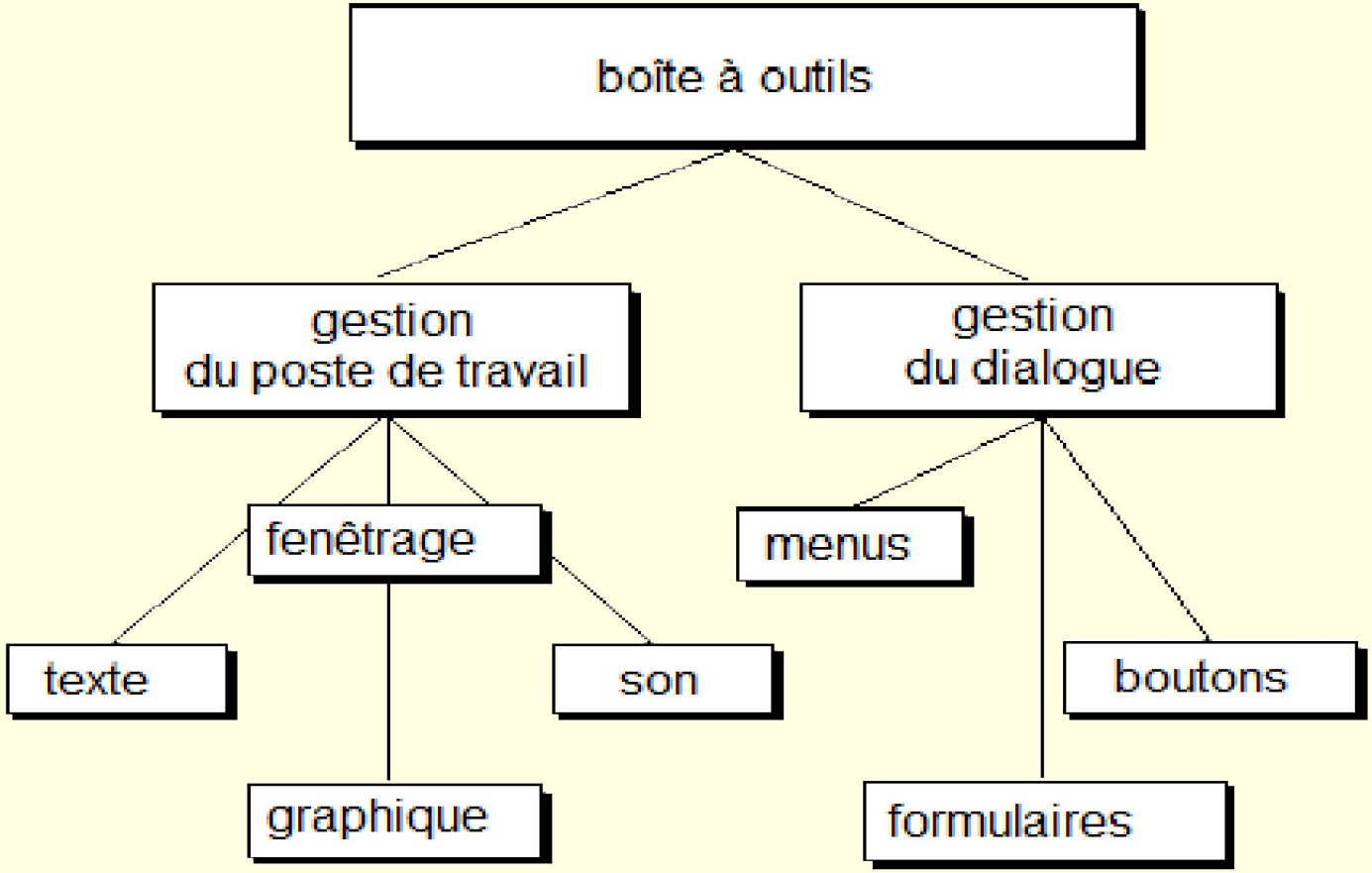
1. Définition et Caractéristiques

Une **boîte à outils** est une **bibliothèque** de procédures adaptées à l'écriture **d'interfaces** homme-machine.

Organisée généralement en 2 catégories:

- **gestion du poste de travail**
- **gestion du dialogue.**

1. Définition et Caractéristiques



Fonctions usuelles d`une Boite à Outils

1. Définition et Caractéristiques

Gestion du Poste de Travail

Permet d`assurer les Tâches :

- Graphiques de base (surface d'affichage),
- Graphiques construites (icône, fenêtre)
- Textuelles (police de caractères)
- Evénements (provenant des unités logiques ou des programmes clients)

1. Définition et Caractéristiques

Gestion du Dialogue

S'appuie sur les fonctions du poste de travail.

Contient deux sortes d'objets de présentation :

- Objets **élémentaires**: boutons, zones de saisie…
- Objets **composés**: comme les barres de défilement et les tableaux de bord

1. Définition et Caractéristiques

Environnement XWindow: les 2 classes, *gestion du poste de travail* et *gestion du dialogue*, sont interfacées dans deux bibliothèques distinctes (Xlib et Xt):

→***Xlib*** *regroupe les primitives d'accès aux services de base (poste de travail)*

→***Xt*** *(ou* ***Intrinsics)*** *regroupe les services de dialogue* , et définit les classes de base à partir desquelles on peut élaborer de nouvelles classes de widgets

Hiérarchie sous XWindow

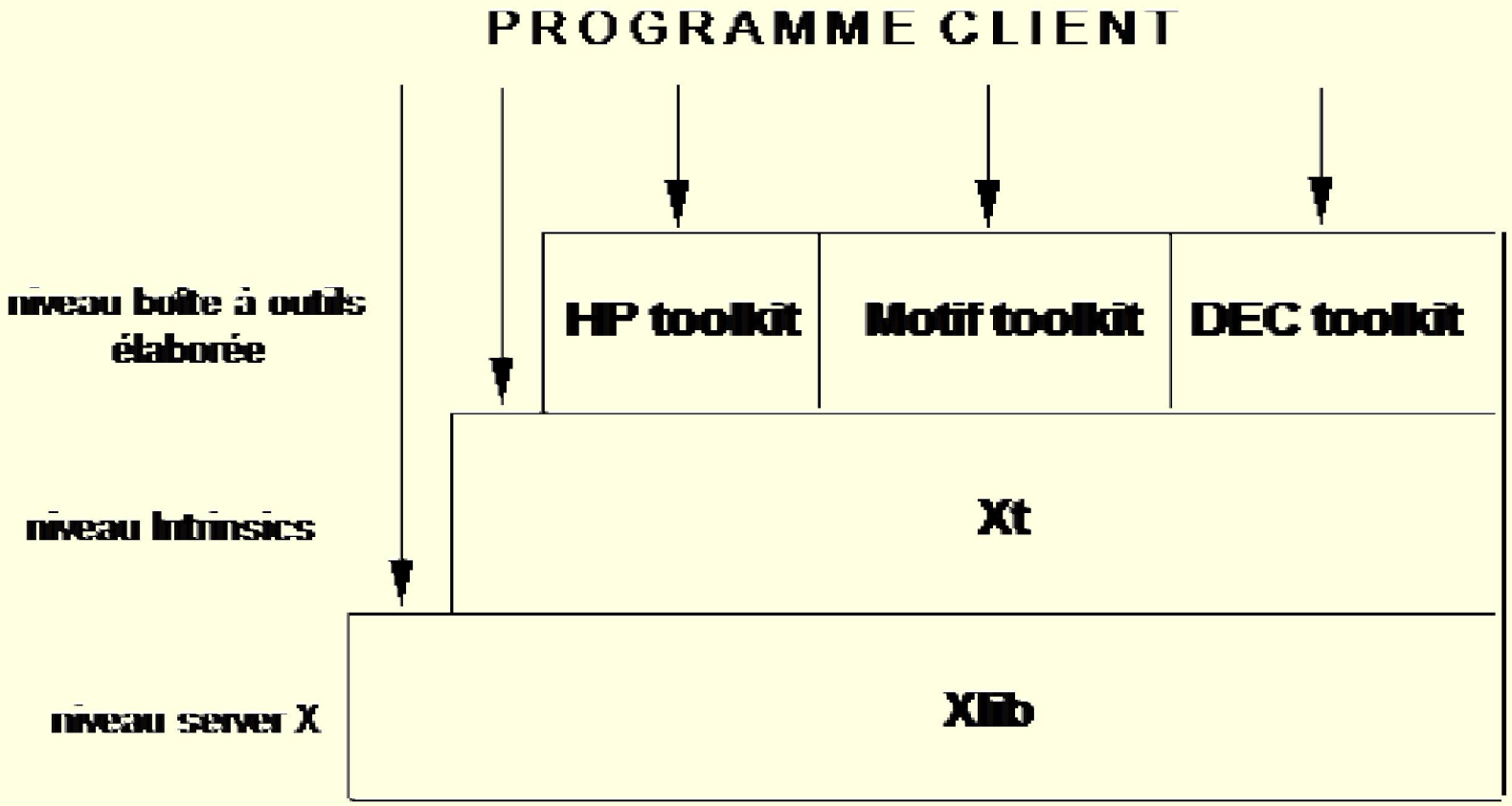


Fig. Les niveaux de Service de l'environnement X

2. Stratégies de Contrôle

2.1 Protocole Embarqué

Tout objet de l'interface (bouton…) véhicule un mécanisme de traitement des événements via des fonctions **associées** à chaque événement (clic bouton gauche,droite..) appelées des ***callback procedures (fonctions de rappel).***

Stratégie appliquée dans l'environnement **Xwindow** et **Windows.** La **boucle de gestion des événements** est **implicite**

2.1 Protocole Embarqué exemple

```
main  { .....

   /*  Etablir une liaison avec le serveur X,
     * Créer la fenêtre toplevel.    */


      toplevel = XtInitialize ("MyProg", ...);
   /*  Créer le bouton quitb, de pere toplevel,
    * de classe XwpushButtonWidgetClass.
    */
   quitb = XtCreateManagedWidget ("quit",
         XwpushButtonWidgetClass,  toplevel, ...);
```

2.1 Protocole Embarqué exemple

- /* Associer la procedure **quit_callback** à l'evenement
  * XtNrelease: relachement du bouton de la souris */
  XtAddCallback (**quitb, XtNrelease, quit_callback**, ..);

  /* **Afficher la fenêtre toplevel et sa descendance** */
  XtRealizeWidget (toplevel);

  /* * **Boucle d'événement: acquérir les événements et * les distribuer aux widgets concernés** */
  XtMainLoop ();
  } /* fin du main */

2.1 Protocole Embarqué exemple

/* Definition de la procedure callback suite à
l'événement **XtNrelease** et termine le programme*/

```
void   quit_callback  (Widget w, ....)
        /* w: widget concerne */
{
   printf  ("Begin exiting ... please standby ... \n");
   fflush (stdout);
   exit (0);
}
```

Protocole embarqué et callback procedure dans l'environnement X

2. Stratégies de Contrôle

2.2 Protocole non Embarqué (moins utilisé)

La boucle d'événement doit explicitement figurer dans le programme client qui doit identifier chaque classe d'événement et appeler la méthode de l'objet qui répond à cette classe

Les objets de la ToolBox de Macintosh fonctionnent selon le protocole non embarqué

2.2 Protocole non Embarqué

```
/* boucle infinie d'acquisition des evenements
     */
    while (go) {
     if (GetNextEvent (everyEvent, myevent) )
       switch (myevent->what) {
         case keyDown : ..... break;

         case mouseDown :
     /*  determiner la fenetre concernee par l'evenement.
      * puis la localisation dans cette fenetre      */
       wheremouse = FindWindow (myevent->where,
                                &whichwindow);

     …
 } /* end while (go) */
```

3. Boîte à Outils QT

Exemple de programme **Hello World**. Il contient juste le minimum pour créer une application Qt.

3. Boîte à Outils QT

```
#include <QApplication>
#include <QPushButton>
int main (int argc, char * argv[])
{
   QApplication app(argc, argv);
   QPushButton  hello ("Hello world!");
   hello.show();
   return app.exec();
}
```

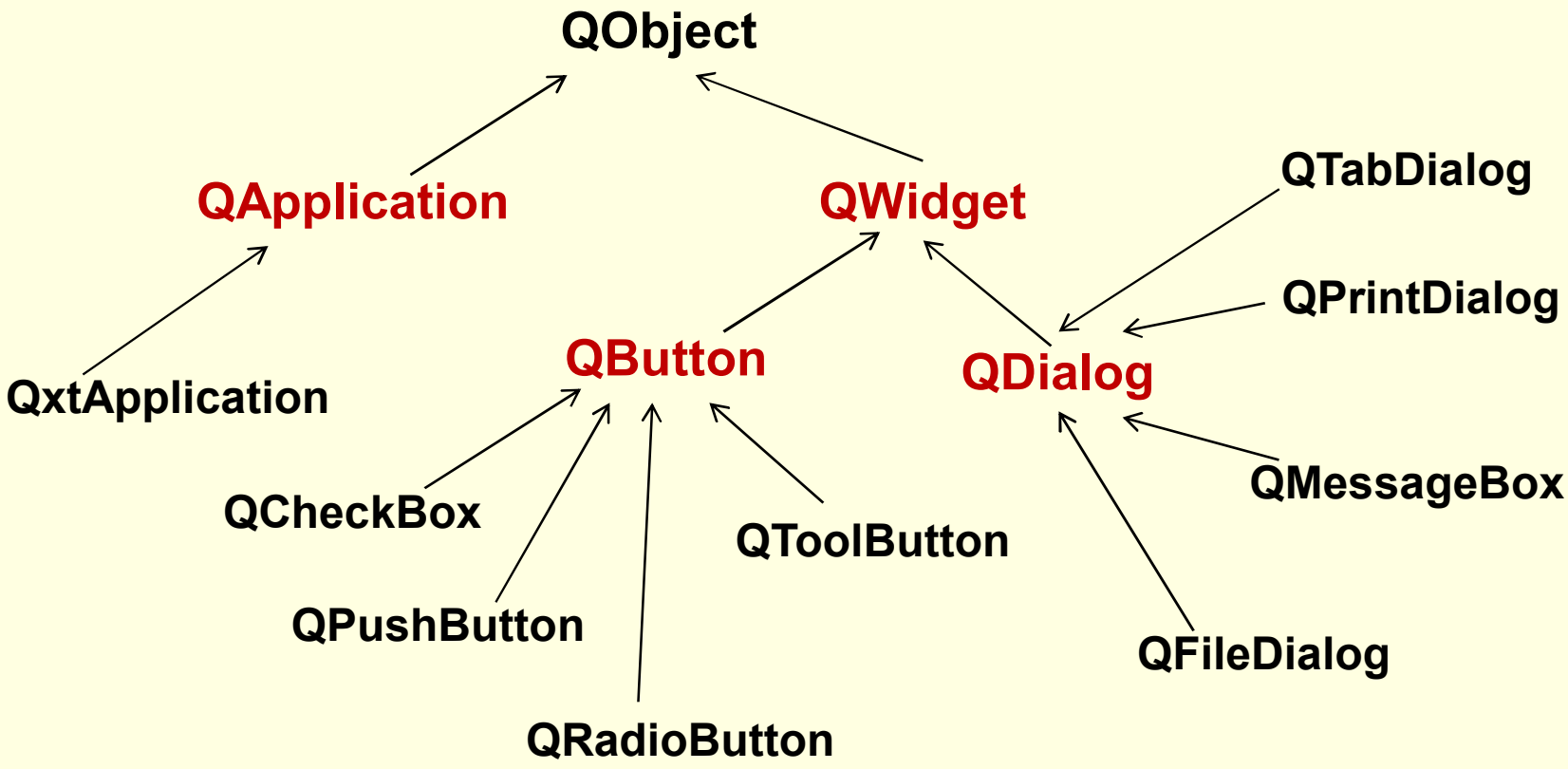
3. Boîte à Outils QT

app: instance de **QApplication** du programme

Qapplication:

- Objet qui se charge de la gestion des événements d'une application
- Classe centrale de Qt qui reçoit des évènements et les transmet à un objet graphique particulier **(widget)**

3. QT (organisation hiérarchique des classes)



3. Boîte à Outils QT

Boite de Dialogue : correspond à une fenêtre, généralement composée de plusieurs composantes atomiques.

Deux modes d'interaction: suspend l'application jusqu'à la fermeture de la boîte de dialogue (**B. Dialogue modale**), ou fonctionne de façon parallèle à l'application (**B. Dialogue non modale**)

Exemple: boîte de message, boîte à onglets

3. Boîte à Outils QT

Disposition: à l'intérieur d'une fenêtre ou B. de dialogue:

A) Disposition Absolue: simple et rapide de conception des composantes à l'intérieur d'une boite de dialogue.

La méthode **SetGeometry()** permet de positionner et dimensionner tous les widgets

```
QWidget *monwidget = new Qwidget ();
 monwidget->setGeometry (200, 300, 120, 90);
Qlabel * montexte= new Qlabel ("Hello", monwidget);
 montexte->setGeometry (10, 10, 80, 30);
```

3. Boîte à Outils QT

B) Disposition Relative: **définit la position** d'une composante **par rapport**:

→ à la position d'une autre composante (**gauche, droite, en haut, en bas**)

→ au bord de **la fenêtre** qui la contient sous forme de contraintes géométriques entre composantes.

Qt utilise des objets **gestionnaires de géométrie**:

QHBoxLayout et ***QVBoxLayout*** pour maintenir une relation horizontale ou verticale entre composantes.

QGridLayout → relation matricielle entre composantes

Compression et étirement : fournit des informations sur la taille (hauteur, largeur) minimale, maximale, idéale de chaque composante.

3. Boîte à Outils QT

Exemple 1 :

```
MonDialogue :: MonDialogue ()
{ QPushButton *b1 = new QPushButton ("bouton 1",
    this);
  b1->setMinimumSize (b1->sizeHint ());
QPushButton *b2 =new QPushButton ("bouton2",this) ;
    b2->setMinimumSize (b2->sizeHint ());
QHBoxLayout *gest = new QHBoxLayout (this);
   gest->addWidget (b1);
   gest->addWidget (b2);
   gest->activate();    }
```

3. Boîte à Outils QT

Exemple 2 :

```
MonDialogue :: MonDialogue ()
{ QPushButton *b1 = new QPushButton ("bouton 1",
    this);
    b1->setMinimumSize (b1->sizeHint ());
QPushButton *b2=new QPushButton("bouton2",this);
    b2->setMinimumSize (b2->sizeHint ());
QCheckBox *c1 = new QCheckBox("cocher ici", this);
    c1->setMinimumSize (c1->sizeHint ());
 c1->setChecked(true);
```

3. Boîte à Outils QT

Exemple 2 (suite) :

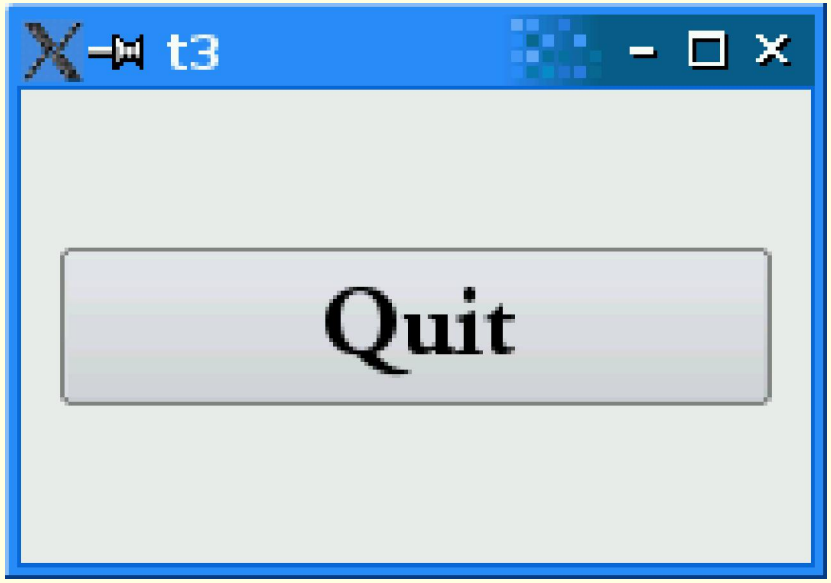
```
QVBoxLayout *gest = new QVBoxLayout (this);
QHBoxLayout *gest2 = new QHBoxLayout (this);
    gest->addWidget (b1);
    gest->addWidget (b2);

   gest2→addLayout(gest);
   gest2->addWidget(c1);
  gest2->activate();   }
```

Exercice (Application)

- Donner en QT le code qui permet de faire le dessin suivant en utilisant :
- La Disposition relative
- La Disposition absolue

3. Boîte à Outils QT

Cet exemple montre comment créer des widgets parents et enfants.



3. Boîte à Outils QT

```
#include <QApplication>
#include <QFont>
#include <QPushButton>        #include <QWidget>
int  main (int argc, char *argv[ ])
{
    QApplication app(argc, argv);
    QWidget window;
    window.resize(200, 120);

    QPushButton quit("Quit", &window);
        quit.setFont (QFont("Times", 18, QFont::Bold) ) ;
        quit.setGeometry(10, 40, 180, 40);
   QObject::connect(&quit, SIGNAL(clicked()),  &app,
    SLOT(quit()));
     window.show();              return app.exec();   }
```

3. Boîte à Outils QT

QObject::connect() → **fct statique indispensable de Qt, établit une connexion à sens unique entre deux objets Qt (qui héritent de QObject, directement ou indirectement)**

Tous les objets Qt peuvent avoir des signaux (pour envoyer des messages) et des slots (pour recevoir).

Tous les widgets sont des objets Qt car ils héritent de QWidget, qui hérite elle-même de Qobject