Référence rapide sur WindowTools

Charles De Léan

23 novembre 1995

1 WindowTools

Windowtools est un progiciel[1] destiné à faciliter la construction d'interfaces graphiques pour les programmes écrits en Simula. Windowtools utilise X, un protocole d'interaction graphique largement répandu dans le monde UNIX qui normalise l'encodage des actions de l'usager comme le "clickage" d'une souris ou l'appui sur une touche du clavier et qui fournit aussi un certain nombre de primitives graphiques pour le dessin de lignes, de rectangles et de cercles de diverses couleurs. Par contre, ce que X ne fait pas, c'est implanter (imposer ?) le comportement des éléments graphiques. Si on veut avoir des fenêtres d'édition de texte ou des menus déroulants, il faut ajouter une couche de logiciel entre X et l'usager. Dans la parlance de X, cette couche est appelée un "Toolkit" et Motif en est l'exemple le plus connu. Si la programmation graphique a été normalisée et simplifiée par l'emploi de X et de Motif, elle reste relativement complexe à cause du grand nombre de paramètres et d'objets fournis afin de répondre au plus grande nombre d'applications.

Windowtools est un "toolkit" comme Motif dont l'objectif est de faciliter le plus possible la création d'interface du style Mac/Windows. Pour ce faire, Windowtools utilise la programmation par objet. Le comportement par défaut des objets implantés est celui auquel les usagers modernes sont habitués: nous avons sacrifié la flexibilité à la simplicité. Dans la majorité des cas, les objets fournis peuvent être utilisés tels quels et il est donc possible de mettre en oeuvre une interface GUI[2] professionnelle avec peu d'efforts. Effectivement, pour faire apparaître une fenêtre affichant "Hello World" à l'écran, il suffit d'une dizaine de lignes avec WindowTools au lieu de plusieurs pages avec X (c.f 4.1).

Windowtools est l'oeuvre de B. Kirkerud qui a travaillé dessus depuis plusieurs années. En 1995, ce logiciel a été installé au DIRO et il a servi pour les TP de ift1020. Entre temps, il a subit une première révision par J. Vaucher. Durant l'été 1995, Charles De Léan (sous la direction du prof. Vaucher) a soumis Windowtools à une phase de vérification intensive, d'extension et de réécriture.

Le document quit suit vise les programmeurs qui se servent de WindowTools pour créer des interfaces graphiques. Son but est de présenter rapidement les differéntes classes (ainsi que leurs fonctions) qui peuvent leur être utiles. Un développeur éventuel pourrait avoir besoin de plus d'information, voir à lire le code source. Pour d'autres informations, vous pouvez explorer le site ftp `ftp.ifi.uio.no` dans le rṕertoire `/pub/simulatools`. Mais de façon plus simple, vous pouvez explorer le répertoire `vaucher/simulatools` sur le réseau JSP.

Windowtools contient une grande varieté de classes. La majorité d'entre elles sont des classes de fenêtres dont nous discuterons plus loin. Sinon, il y a quelques classes à part. La plus importante d'entre elles est `XDisplay`. C'est la classe d'un objet qui gère les messages (venant du clavier ou de la souris par exemple) et les ressources (fontes et curseurs par exemple). Il ne doit y avoir qu'un seule instance de cet objet dans le programme. Cette instance sera mentionné dans ce document sous le nom de "display". Pour s'assurer de l'unicité d'un tel objet, nous introduisons deux fenêtres spéciales qui seront les fenêtres principales dans le programme. Ces fenêtres créent le display, et donc, il faut en faire une et

[1]Produit logiciel

[2]Graphic User Interface

une seule (c.f MainWindow 3.1 et SimpleWindow 3.2) au tout début du programme.

Les ressources, gérées par le display, sont définies par plusieurs classes. Les fontes sont définies par `XFont` (c.f 2.6), les curseurs le sont par la classe `Cursor` (c.f 2.11.3), les bitmaps et pixmaps (très utiles pour simuler de la couleur) par `Bitmap` et `PixMap` (c.f 2.10), et les requêtes d'alarmes ainsi que les messages internes, par `alarm_request` et `own_event` (c.f 2.8).

Chaque fenêtre possède un objet très utile pour ses fonctions graphiques. C'est le contexte graphique (graphic context), défini par la classe `XGraphicalContext`. C'est un objet contenant des paramètres servant aux fonctions graphiques. Ces paramètres peuvent être modifiés par plusieurs fonctions (c.f 2.10).

Passons maintenant aux fenêtres. On peut dire qu'il existe trois types de fenêtres :

- Fenêtres principales (MainWindow et SimpleWindow)

  Il doit y avoir une et une seule de ces fenêtres crée au début du programme.

- Fenêtres indépendantes (SideWindow et ses sous-classes)

  Comme les fenêtres principales, sauf qu'on peut les créer et détruire quand on veut et à volonté.

- Sous-fenêtres (Subwindow et ses sous-classes)

  Ce sont les classes les plus nombreuses. Ces fenêtres appartiennent à une autre. Elles sont détruites en même temps que leurs parents et se repèrent comme si leur parent était l'écran.

Les classes utiles pour fabriquer les fenêtres de WindowTools ne sont pas toutes décrites ici. Certaines classes ont été volontairement omises car elles ne risquent pas d'être utiles à un programmeur. Ces classes omises sont

- HeadWindow

  Fenêtre pouvant contenir un titre (c.f 2.3.1)

- ScrollBar

  Fenêtre permettant d'indiquer où on se trouve dans une liste dont on ne voit qu'une partie. Ne peut servir qu'à `ScrollWindow`

- ScrollBox

  Des fenêtres pouvant changer la sélection courante dans une fenêtre. Ne peut servir que dans `ScrollWindow` et `MultipleChoiceWindow`

- RadioButton

  Classe de boutons dans une fenêtre de classe `RadioPanel`. Ne peut servir que dans `RadioPanel`.

- PopupListWindow(popupmenu)

  Sous-classe de `ListWindow` servant à `PopupMenu`. Elle n'est pas pensée pour servir à construire d'autres types de fenêtres.

Voici la hiérarchie des fenêtres dont nous discuterons.

Hiérarchie des classes de fenêtres

```
XWindow
  |---MainWindow
  |      |---SideWindow
  |      |      |---GraphPage
  |      |      |---PopupWindow
  |      |      |      |---PopupMenu
  |      |---SimpleWindow
  |---Subwindow
         |---Button
         |      |---FancyButton
         |      |---MenuButton
         |---DrawWindow
         |      |---GraphWindow
         |---InputWindow
         |      |---ItemWindow
         |      |      |---BoolItemWindow
         |      |      |---CharItemWindow
         |      |      |---IntItemWindow
         |      |      |---ReallItemWindow
         |      |      |---TextItemWindow
         |      |---ListWindow
         |             |---SelectionListWindow
         |---MessageWindow
         |---ParameterList
         |---ParameterMenu
         |---ParameterWindow
         |      |---BoolParameter
         |      |---CharParameter
         |      |---IntParameter
         |      |---RealParameter
         |      |---TextParameter
         |---PromptWindow
         |---RadioPanel
         |---ScrollWindow
                |---MenuScrollWindow
                       |-DirectoryMenu
```

2 XWindow

Cette classe contient toutes les fonctions communes aux fenêtres et sert de super-classe pour toute fenêtre graphique. Elle ne permet pas la création directe d'une fenêtre. Seules ses sous-classes peuvent y mener.

2.1 Géometrie

Dans cette section, on retrouve toutes les fonctions ayant rapport à la disposition des fenêtres (position et taille).

2.1.1 Accès

Ces fonctions donnent les différentes caractéristiques géométriques de la fenêtre en pixels. Toute position est relative au point supérieur gauche de sa fenêtre parente.

- integer procedure width;
- integer procedure height;

  Largeur et longeur de la partie dessinable de la fenêtre.

- integer procedure min_x;
- integer procedure min_y;
- integer procedure max_x;
- integer procedure max_y;
- integer procedure mid_x;
- integer procedure mid_y;

  Positions relatives à la fenêtre parente des coins et centre de la partie dessinable de la fenêtre.

- integer procedure BorderWidth;

  Epaisseur de la bordure à l'extérieur de la partie dessinable.

- integer procedure DepthBorder;

  Epaisseur de la bordure à l'intérieur de la partie dessinable.

- procedure FindAbsolutePos(x,y);name x,y;integer x,y

  Position absolue de la partie dessinable par rapport à la fenêtre

2.1.2 Changement

Ces fonctions modifient (ou initialisent) la taille,position et bordure de la fenêtre à partir de paramètres qui leurs sont donnés.

- ref(XWindow) procedure SetSize(new_width, new_height);
- ref(XWindow) procedure SetWidth(w); integer w;

- ref(XWindow) procedure SetHeight(h); integer h;

    Changent la taille de la fenêtre

- ref(XWindow) procedure PlaceAt(minx, miny); integer minx, miny;
- ref(XWindow) procedure PlaceWithCenterIn(x, y); integer x, y;

    Place la fenêtre à la position donnée. La position est relative à la fenêtre parente.

- ref(XWindow) procedure SetBorderWidth(new_borderwidth);
- ref(XWindow) procedure SetDepthBorder(db); integer db;

    Voir BorderWidth et DepthBorder (2.1.1)

- ref(XWindow) procedure SetColumns(ncolumns); integer ncolumns;
- ref(XWindow) procedure SetLines(lines); integer lines;

    Fixent la taille de la fenêtre en fonction de la fonte utilisée.

2.1.3 Positionnement relatif

Ces fonctions permettent de placer les fenêtres par rapport à la fenêtre parente (ou le fond de l'écran au besoin), ou par rapport à une autre fenêtre. Dans le deuxième cas, il faut que cette fenêtre soit une fenêtre soeur (sous-fenêtre de la fenêtre parente). Sinon, les résultats ne seront pas bons.

Windowtools essaiera de conserver la disposition donnée lors des changements de taille de fenêtre.

- ref(XWindow) procedure PlaceCentered;
- ref(XWindow) procedure PlaceUpLeft;
- ref(XWindow) procedure PlaceUpRight;
- ref(XWindow) procedure PlaceDownLeft;
- ref(XWindow) procedure PlaceDownRight;
- ref(XWindow) procedure PlaceAfter(wnd); ref(XWindow) wnd
- ref(XWindow) procedure PlaceBefore(wnd); ref(XWindow) wnd
- ref(XWindow) procedure PlaceBelow(wnd); ref(XWindow) wnd
- ref(XWindow) procedure PlaceAbove(wnd); ref(XWindow) wnd
- ref(XWindow) procedure PlaceRightof(wnd); ref(XWindow) wnd
- ref(XWindow) procedure PlaceLeftof(wnd); ref(XWindow) wnd
- ref(XWindow) procedure PlaceRightBelow(wnd); ref(XWindow) wnd
- ref(XWindow) procedure PlaceLeftBelow(wnd); ref(XWindow) wnd

2.1.4 Sous-fenêtres

Ces fonctions utilisent et/ou modifient les sous-fenêtres.

- integer procedure SubwindowSpacing;
- ref(XWindow) procedure SetSubwindowSpacing(w); integer w;

    Séparation entre les sous-fenêtres lors du placement automatique de celles-ci.

- ref(XWindow) procedure MakeButtonsSameWidth;

  Met tous les boutons de la fenêtre à la même largeur (soit celle de la plus large).

- ref(XWindow) procedure SetSizeToSubwindowSize;
- ref(XWindow) procedure SetHeightToSubwindowHeight;
- ref(XWindow) procedure SetWidthToSubwindowWidth;

  Changent la taille de la fenêtre pour englober exactement les sous-fenêtres. Il ne se passe rien s'il n'y a pas de sous-fenêtre.

2.2 Création de fenêtres

Dans cette partie, on vous explique deux façons de créer des fenêtres et comment les détruire.

2.2.1 Version manuelle

Il est toujours possible de créer des fenêtres à l'aide de la fonction `new`. Une fois un objet instancié, vous pouvez le manipuler comme une fenêtre. N'oubliez pas, dans plusieurs cas, les fenêtres ne s'affichent pas tout seuls et peuvent avoir des tailles ou des positions indésirables (c.f 2.5.1, 2.1.2 et 2.1.3).

2.2.2 Version automatique

Ces fonctions servent à répondre à la demande de création de sous-fenêtres. Par défaut, elles créent la fenêtre, la place et l'affichent à l'écran. Ces fenêtres seront séparées d'une distance prédéfinie accessible par `subwindowspacing` (c.f 2.1.4). Attention ! Si jamais vous détruisez une fenêtre créé par ces fonctions, ne les réutilisez plus pour cette fenêtre.

- ref(Button) procedure MakeButton(heading); text heading;
- ref(FancyButton) procedure MakeFancyButton(heading); text heading;
- ref(PromptWindow) procedure MakePromptWindow;
- ref(PromptWindow) procedure MakeFancyPromptWindow;
- ref(MessageWindow) procedure MakeMessageWindow;
- ref(IntItemWindow) procedure MakeIntItemWindow;
- ref(RealItemWindow) procedure MakeRealItemWindow;
- ref(CharItemWindow) procedure MakeCharItemWindow;
- ref(BoolItemWindow) procedure MakeBoolItemWindow;
- ref(TextItemWindow) procedure MakeTextItemWindow;
- ref(ScrollWindow) procedure MakeScrollWindow;
- ref(ListWindow) procedure MakeListWindow;
- ref(MultipleChoiceWindow) procedure MakeMultipleChoiceWindow;
- ref(RadioPanel) procedure MakeRadioPanel;
- ref(MenuWindow) procedure MakeMenu;
- ref(ParameterMenu) procedure MakeParameterMenu;
- ref(ParameterList) procedure MakeParameterList;
- ref(DrawWindow) procedure MakeDrawWindow;

2.2.3 Destruction

La création des choses prévoit souvent leur destruction. C'est particulièrement vrai en programmation. Pour détruire une fenêtre, il suffit d'appeler la méthode `Destroy`, dont la déclaration est : `procedure Destroy`.

L'appel de cette méthode devrait être la dernière chose que vous faites avec l'objet.

Vous n'êtes pas obligés de détruire une fenêtre une fois qu'elle est inutile. La fin de l'execution du programme s'en chargera. Mais, comme pour l'allocation de mémoire, vous pouvez ainsi libérer un espace qui peut devenir précieux.

2.3 Titre

Ces fonctions servent donner un titre à la fenêtre. Les titres peuvent être mis dans une sou-fenêtre ou dans la fenêtre mère.

2.3.1 HeadWindow

Ces fonctions manipulent une sous-fenêtre spéciale qui sert à contenir un titre.

- ref(XWindow) procedure SetHeading(heading); text heading;
- ref(XWindow) procedure MakeHeadWindow(heading); text heading;

  Créent au besoin une fenêtre titre et mettent le texte dans la fenêtre titre.

- integer procedure head_height;
- integer procedure head_width;

  Donnent la taille de la fenêtre titre.

- ref(XWindow) procedure SetHeadFont(font_name); text font_name;

  Change la fonte de la fenêtre titre.

2.3.2 Label

Ces fonctions donnent une façon alternative pour donner un titre. Le "label" est un titre que la fenêtre mère doit afficher elle-même.

- ref(XWindow) procedure SetLabel(a_label); text a_label

  Permet de donner un titre qui sera affiché à côté de la fenêtre.

- procedure ShowCurrentLabel

  Fonction qui sera appelée lorsque la fenêtre sera affichée. Son implantation par défaut demande à la fenêtre mère (s'il y en a), d'afficher le label à gauche de la fenêtre. Il pourrait être utile de la réecrire si vous voulez changer la position du label. Cependant, si vous changez la position du label, il est possible que vous ayez des problèmes avec le positionnement automatique qui suppose que le label est toujours a gauche.

2.4 Dessin

Les prochaines fonctions servent a dessiner dans le fond de la fenêtre. En aucune façon, elles font dessiner par-dessus une autre fenêtre. Il existe plus loin des fonctions plus avancés qui permettent de modifier le contexte graphique, donc le comportement général des ces fonctions, à la section 2.10.

2.4.1 Fonctions Générales

Ces fonctions utilisent le contexte graphique sans modification. Toutes position ici est donnée en pixels.

- ref(XWindow) procedure DrawLine(x1, y1, x2, y2); integer x1, y1, x2, y2;

  Point de départ et d'arrivée.

- ref(XWindow) procedure DrawPoint(x, y); integer x, y;

  Les coordonnées du point.

- ref(XWindow) procedure DrawPolygon(x, y, n); integer array x, y; integer n;

  Les points des différents sommets et le nombre de sommets.

- ref(XWindow) procedure DrawArc(x1, y1, x2, y2, ang1, ang2); integer x1, y1, x2, y2; real ang1, ang2;

  Les coordonnées du rectangle dans lequel serait inscrit l'éllipse si l'arc devait en être une. Les coordonnées sont le coin supérieur gauche et le point inférieur droit. Les angles sont en degrés et vont dans le sens géometrique. L'angle 0 est en position "est" si le "nord" est le haut de l'écran.

- ref(XWindow) procedure DrawCircle(center_x, center_y, radius); integer center_x, center_y, radius;

  Centre et rayon du cercle.

- ref(XWindow) procedure DrawEllipse(x1, y1, x2, y2); integer x1, y1, x2, y2;

  Comme DrawArc, sauf que les angles sont 0 et 360.

- ref(XWindow) procedure DrawRectangle(x, y, width, height);

  Point supérieur gauche et taille du rectangle.

2.4.2 Fonctions Inverse

Ces fonctions modifient momentanément le contexte graphique pour inverser les zones où ils doivent dessiner. On peut faire l'équivalent en appelant la fonction `invert`, suivis d'appels aux fonctions générales (c.f 2.10).

Note : Les fonctions `DrawInvertRectangle` et `InvertRectangle` sont identiques.

- ref(XWindow) procedure DrawInvertLine(x1, y1, x2, y2); integer x1, y1, x2,y2;
- ref(XWindow) procedure DrawInvertPoint(x, y); integer x, y;
- ref(XWindow) procedure DrawInvertRectangle(x, y, width, height);
- ref(XWindow) procedure InvertRectangle(x, y, width, height);

2.4.3 Fonctions de Remplissage

Comme pour les fonctions inverse, ces fonction modifient temporairement le contexte graphique pour remplir les zones dessinées. La couleur utilisée peut etre modifiée par `SetFill` (c.f 2.10).

- ref(XWindow) procedure FillArc(x1, y1, x2, y2, ang1, ang2); integer x1, y1, x2, y2; real ang1, ang2;
- ref(XWindow) procedure FillCircle(center_x, center_y, radius); integer center_x, center_y, radius;
- ref(XWindow) procedure FillEllipse(x1, y1, x2, y2);
- ref(XWindow) procedure FillRectangle(x, y, width, height);
- ref(XWindow) procedure FillPolygon(x, y, n); integer array x, y; integer n

2.4.4 Texte

Ces fonctions servent a afficher du texte. Elle utilisent la fonte courante pour la forme des lettres (c.f 2.6).

- ref(XWindow) procedure HighLightText(on); Boolean on;

  Indique si on veut une couleur de fond derrière le texte qu'on affiche.
- ref(XWindow) procedure DrawText(x, y, t); integer x, y; text t;

  On donne les coordonnées du point supérieur gauche du rectangle contenant le texte.

- ref(XWindow) procedure DrawTextCenterIn(x, y, t); integer x, y; text t;

  On donne le point sur lequel le texte doit être centré.

- ref(XWindow) procedure DrawCenterText(t); text t;

  Demande que le texte soit centré dans la fenêtre.

2.4.5 Autres

Ces fonctions opèrent sur des rectangles dans le fond de la fenêtre.

- ref(XWindow) procedure CopyRectangle(srcx, srcy, width, height, destx, desty);

  Copie le rectangle donné par les 4 premiers paramètres à une autre position dont le coin supérieur gauche est donné par les 2 derniers.

- ref(XWindow) procedure WhiteRectangle(x, y, width, height);

  Dessine un rectangle blanc plein.

- ref(XWindow) procedure ClearRectangle(x, y, width, height);

  Efface le contenu du rectangle.

2.5 Affichage

Ces fonctions manipulent la présentation et le comportement graphique de la fenêtre en général sans modifier le comportement du dessinage dans le fond de la fenêtre (i.e. on ne touche pas au contexte graphique encore).

2.5.1 La fenêtre dans l'écran

On a ici le comportement graphique de la fenêtre par rapport aux autres.

- ref(XWindow) procedure Raise;

  Met la fenêtre au-dessus de toutes ses soeurs.

- ref(XWindow) procedure Show;
- ref(XWindow) procedure Hide;

  Montre ou cache la fenêtre (ainsi que le label s'il y en a un).

- ref(XWindow) procedure SaveUnder;

  Demande au systeme de sauver les parties de fenêtres qui seront cachés par cette fenêtre. Ne fonctionne pas sur toutes les stations.

- ref(XWindow) procedure SetBackingStore;

  Demande au système de sauver le contenu graphique de la fenêtre si elle vient à être cachée par un autre fenêtre ou si on lui fait "hide". Ne fonctionne pas sur toutes les stations.

2.5.2 Le fond de la fenêtre

Fonctions qui modifient le fond de la fenêtre au complet.

- ref(XWindow) procedure Invert;

  Inverse ce qui est affiche dans la fenêtre.

- ref(XWindow) procedure Fill;

  Remplit la fenêtre avec la couleur de remplissage

- ref(XWindow) procedure Clear;

  Efface ce qui est affiché dans la fenêtre.

- procedure SetFancy

  Indique qu'on veut une présentation spéciale pour cette fenêtre. Elle prendra une certaine profondeur et une couleur grise. Cette commande fait en sorte que toutes les sous-fenêtres crées ensuite seront aussi "fancy".

- Boolean procedure IsFancy;

  Indique si une fenêtre est "fancy"

- ref(XWindow) procedure SetBackGround(fillname); text fillname;

  Change la couleur de fond de la fenêtre.

2.5.3 La bordure interne

Les quatre fonctions suivantes dessinent une bordure interne dont l'épaisseur est donnée par `depthborder` (c.f 2.1.1).

- ref(XWindow) procedure Thickborder;

  Bordure tout autour

- ref(XWindow) procedure Whiteborder;

  Bordure blanche tout autour.

- ref(XWindow) procedure Upborder;

  Demi-bordure au gauche et en haut.

- ref(XWindow) procedure Downborder;

  Demi-bordure au droite et en bas.

2.6 Fontes

Voici les différentes fonctions pouvant manipuler les fontes de la fenêtre. Les différentes fontes possibles varient d'un serveur à l'autre. Il est conseille ici que vous vous procurez une documentation supplémentaire pour avoir acces aux différentes qui vous sont accessible.

- ref(XWindow) procedure SetFont(font_name); text font_name;

  Change la fonte de la fenêtre. Il faut donner le nom complet.

- ref(XWindow) procedure SetTypeFace(typeface, size, bold, slanted); text typeface; integer size; Boolean bold, slanted;

  Change la fonte de la fenêtre. Il suffit de donner les 4 caractéristiques demandées.

- ref(XWindow) procedure SetSymbolFont(typeface, size);

  Comme `SetTypeFace`, sauf que bold et slanted sont à "true".

- text procedure CurrentFontname;

  Donne le nom complet de la fonte courante.

- Boolean procedure font_set;

  Indique si on a bel et bien un fonte pour cette fenêtre.

- integer procedure width_of_text(t); text t;

  Donne la taille en pixels que prendrait le texte t avec la fonte courante.

- integer procedure font_ascent;
- integer procedure font_descent;
- integer procedure font_height;
- integer procedure max_char_width;
- integer procedure font_lbearing;
- integer procedure font_rbearing;
- integer procedure font_minrank;
- integer procedure font_maxrank;
- integer procedure leftbearing_of_char(c); character c;
- integer procedure rightbearing_of_char(c); character c;
- integer procedure width_of_char(c); character c;
- integer procedure height_of_char(c); character c;
- integer procedure ascent_of_char(c); character c;
- integer procedure descent_of_char(c); character c;

  Donnent les differentes caractéristiques de la fonte courante.

- ref(XWindow) procedure SettNorskTekst;
- ref(XWindow) procedure SetEnglishText;

  Permettent de changer certain titres dans la fenêtre et les sous-fenêtres à venir. Les titres "cancel" "clear" "yes" "no" peuvent être ainsi remplacés par leur équivalent norvégiens. Le booléen Norsktekst indique si les textes doivent être donnés en norvégiens.

2.7 Messages

Voici les fonctions de message principales. Toutes ces fonctions sont des fonctions virtuelles. Dans la classe XWindow, elles ne font absolument rien. Elles sont redéfinies dans les sous-classes des sous-fenêtres (subwindow). Vous devriez vous sentir libre de redéfinir autant de ces fonctions que vous voulez dans votre sous-classe éventuelle de mainwindow,sidewindow ou simplewindow. Dans simplewindow, seule la fonction handle_configure a été redéfinie.

Si vous souhaitez redéfinir de ces fonctions dans une sous-classe d'une sous-fenêtre (c'est à dire une "sous-sous-classe" de subwindow), il est fort probable que vous éliminerez ainsi un comportement de la super-classe.

Certaines de ces fonctions ne seront appelées que sous certaines conditions (c.f 2.8 et 2.9)

Les messages sont envoyés par les fenêtres ou par le display.

2.7.1 Messages venant de fenêtres

Ces prochains messages ne proviennent pas du display mais d'autres fenêtres (de sous-fenêtres surtout).

- procedure Perform(heading); text heading
- procedure PerformIn(heading, wnd); text heading; ref(SubWindow) wnd

  Annonce une réponse d'une sous-fenêtre (de classe button ou menuwindow, c.f. 3.8 et 3.13). La redéfinition de `perform` peut entraîner des problèmes avec la fonction buttonchoice (c.f 2.8).

- procedure ClickInButton(b); ref(Button) b

  La meilleure fonction à implanter pour détecter les boutons cliqués (c.f 3.8). Si ce message n'est pas récupéré par la fenêtre parente, elle continuera a se propager dans la hiérarchie des fenêtres.

- procedure ClickInRadioPanel(rb); ref(RadioPanel) rb

  Annonce qu'on a choisi un radiobutton dans un des radiopanel de la fenêtre (c.f 3.21).

- procedure LineChoice(line); text line
- procedure LineChoiceIn(line, line_number, wnd); text line; integer line_number; ref(SubWindow) wnd

  Annonce la sélection d'une ligne dans un listwindow, scrollwindow ou menubutton (c.f 3.12 et 3.14 3.9).

- procedure InputInSubwindow(sw); ref(SubWindow) sw

  Annonce qu'on a changé le contenu d'une sous-fenêtre (c.f 3.11). La fenêtre modifiée peut être une sous-fenêtre à n'importe quel niveau de profondeur. Le message continue à se propager aux fenêtres parents jusqu'à ce qu'elle soit interceptée.

- procedure SizeChange_in_SubWindow(wnd); ref(SubWindow) wnd

  Annonce qu'une sous-fenêtre a changé de taille (c.f 3.20).

- procedure SetSizeSub;

  Fonction appelée à chaque fois qu'une fonction redéfinissant la taille est appelée (c.f 2.1.2). Ce message est envoyé de la fenêtre à elle-même.

2.7.2 Messages du display

Voici une courte description des causes des appels des messages envoyés par le display.

- procedure Refresh

  Le fond de la fenêtre a été effacé. Il faut le redessiner.

- procedure handle_key_down(key); character key
- procedure handle_key_up(key); character key

  Annonce l'enfoncement ou le relâchement d'une touche ayant un code ASCII.

- procedure handle_special_key_down(key_sym); integer key_sym
- procedure handle_special_key_up(key_sym); integer key_sym

  Même chose que les deux fonctions précédantes, sauf que c'est pour des touches spéciales (tels que crtl,shift,alt).

- procedure handle_button_down(button); integer button
- procedure handle_button_up(button); integer button
- procedure handle_button_click(button); integer button
- procedure handle_second_click(button); integer button

  Indique quel bouton de souris vient d'être enfonce, relâché, cliqué ou cliqué une deuxième fois. Si l'utilisateur fait un double-click, les deux messages `handle_button_click` et `handle_second_click` seront envoyés.

- procedure handle_pointer_motion(x, y); integer x, y

  Annonce la nouvelle position de la souris

- procedure handle_enter_window
- procedure handle_leave_window

  Annonce l'entrée ou la sortie de la souris de la fenêtre.

- procedure handle_destroy

  Annonce que la fenêtre a été détruite. N'a jamais été redéfinie dans la librairie.

- procedure handle_configure

  Annonce que la fenêtre a changé de taille.

- procedure handle_wakeup
- procedure handle_alarm

  Appelés à la suite des appels de `WakeUp` et `SetAlarm` (c.f 2.8).

2.8 Evénements

Ces différentes méthodes agissent directement avec le display. Ils permettent de générer un événement, d'en attendre, ou d'en traiter un.

- ref(XWindow) procedure Wait

  Par défaut, cette fonction ne fait rien. Cette fonction est redéfinie dans scrollwindow et button (c.f 3.8 et 3.14) ou elle attend après une réponse.

- text procedure ButtonChoice;

  Attente active qu'un des boutons de la fenêtre soit choisi. Cette attente active est arrêtee par la définition par défaut de la fonction `perform`. Si vous redéfinissez cette fonction et que vous utilisez ButtonChoice dans le fenêtre, il est fortement conseillé d'ajouter les lignes

  `chosen_button_heading :- heading;`
  `button_choice_made := true;` où heading est le paramètre de `perform`. De façon plus générale, si un événement doit casser l'attente active de buttonchoice, l'instruction "`button_choice_made := true;`" est la solution. `chosen_button_heading` sera la réponse de buttonchoice une fois l'attente terminée.

- procedure WakeUp;

  Poste un événement dans une file d'attente interne à windowtools. La réponse sera un appel à la fonction `handle_wakeup` une fois que tous les messages de X et des autres demandés à `handle_wakeup` déjà en attente soient deja traitées.

  Un appel à une telle fonction, suivie d'une boucle appelant `handle_event` sans cesse et s'arrêtant lors de l'appel de `handle_wakeup` peut être intéressante pour gérer des processus de fenêtres en parallèle.

- procedure SetAlarm(time); integer Time;

  Demande au display s'appeler `handle_alarm` dans un nombre de secondes donné par Time. Les valeurs nulles ou négatives entraînent un appel instantané.

- procedure handle_event;

  Traite un événement éventuel en attente. Cet événement peut être un événement d'alarme, ou ayant rapport à l'interface usager. Dans les deux cas, un message venant de X. Si aucun message de ce type est en attente, on traite alors les événements postés par les fenêtres grâce à la fonction wakeup.

- procedure handle_pending_events;

  S'assure que tous les événements envoyés par X sont traités.

2.9 Filtres de messages

Les fonctions suivantes permettent ou interdisent au display d'envoyer des fonctions de message aux fenêtres.

2.9.1 Filtres simples

Ces fonctions modifient l'accès aux fonctions parenthèsés qui les suivent dans la fenêtre concernée (et pas dans ses sous-fenêtres).

- procedure button_sensitize;
- procedure button_desensitize;

  (handle_button_down,handle_button_up, handle_button_click,handle_second_click)

- procedure key_sensitize;
- procedure key_desensitize;

  (handle_key_down,handle_key_up, handle_special_key_down,handle_special_key_up)

- procedure alarm_sensitize;
- procedure alarm_desensitize;

  (handle_alarm)

- procedure enter_and_leave_sensitize;
- procedure enter_and_leave_desensitize;

  (handle_enter_window,handle_leave_window)

- procedure pointer_motion_sensitize;
- procedure pointer_motion_desensitize;

  (handle_pointer_motion)

2.9.2 Filtres complexes

Les fonctions suivantes sont plus complexes. Elles peuvent modifier la permission d'envois de plusieurs fonctions messages et peuvent aussi se transmettre aux sous-fenêtres.

- ref(XWindow) procedure AllowInput;
- ref(XWindow) procedure DisallowInput;

  La fonction par défaut est de transmettre le même message à ses sous-fenêtres. Elles sont redéfinies dans les classes inputwindow,scrollwindow et listwindow (c.f 3.11,3.14 et 3.11)

- ref(XWindow) procedure AllowChoice;

- ref(XWindow) procedure DisAllowChoice;

  Par défaut, ces fonctions ne font rien. Les fenêtres scrollwindow et listwindow redéfinissent ces fonctions (c.f 3.12 et 3.14).

- ref(XWindow) procedure AllowButtonsInWindow;
- ref(XWindow) procedure DisallowButtonsInWindow;

  Permet ou interdit la sélection de tous les boutons de la fenêtre (c.f 3.8).

2.10 Contexte graphique

Le contexte graphique est un paquet de paramètres qui sont passés de façon intrinsèque à chaque fonction graphique. Les différentes couleurs possibles varient d'une station à l'autre. Cependant, en tout temps, il est possible d'utiliser des tons de gris. Il suffit de demander la couleur "grayXXX" où XXX est un nombre de 0 à 100, représentant le degré de noirceur.

- ref(XWindow) procedure SetLineWidth(linewidth); integer linewidth;

  Change la taille des lignes. Une taille de zéro donne une ligne mince et très rapide à dessiner.

- ref(XWindow) procedure SetBlackOnWhite;
- ref(XWindow) procedure SetWhiteOnBlack;

  Changent le foreground et le background du contexte graphique en même temps. Le background n'a aucun rapport avec la couleur de fond de la fenêtre.

- ref(XWindow) procedure SetDrawStipple(stipplename); text stipplename;
- ref(XWindow) procedure SetStipple(stp); ref(Bitmap) stp;

  Changent le stipple grâce à un nom ou un bitmap.

- ref(XWindow) procedure SetNoStipple;

  Arrête l'utilisation du stipple.

- ref(XGraphicalContext) procedure copygc;
- procedure restoregc(gc); ref(XGraphicalContext) gc;

  Permettent de sauvegarder le contexte graphique. Ainsi, vous pouvez modifier le contexte graphique courant à votre guise pour vos opérations graphiques. A la fin, il est important d'appeler `restoregc` pour faire revenir le contexte graphique à son état initial.

- ref(XGraphicalContext) procedure InvertGC;

  Inverse le fond de l'écran.

- ref(XGraphicalContext) procedure FillGC;

  Dessine avec la couleur de remplissage.

- ref(XGraphicalContext) procedure WhiteGC;
- ref(XGraphicalContext) procedure BlackGC;

  Dessine en noir ou blanc.

- ref(XWindow) procedure SetFill(fillname); text fillname;

  Change la couleur de remplissage.

- ref(XWindow) procedure SetForeground(foreground); text foreground;

  Change la couleur de dessinage.

2.11 Autres fonctions

On a ici les dernières sections trop petites pour être des sections à part.

2.11.1 Fenêtre principale

Lors de la création d'une fenêtre indépendante de la fenêtre principale (SideWindow, c.f 3.3), il faut tout de même y avoir une référence.

- ref(XWindow) procedure TopWindow

  Donne une référence sur la fenêtre principale.

2.11.2 Debugging

Ces différentes fonctions servent à fabriquer des traces dans le programme.

- procedure set_trace;
- procedure unset_trace;

  Demandent d'imprimer à la sortie standard tous les événements que le display reçoit.

- text procedure window_kind

  Fonction virtuelle donnant la classe exacte de la fenêtre. Devrait être ajouté à toute nouvelle classe de fenêtre. Sera appelée si on demande une trace des événements.

- procedure write_geometry;

  Demande d'imprimer à la sortie standard la taille et position de la fenêtre.

2.11.3 Souris

Accès à la position et à la forme du pointeur de la souris.

- ref(XWindow) procedure PlacePointer(x, y); integer x, y;

  Place le pointeur de souris relativement à la partie dessinable de la fenêtre.

- ref(XWindow) procedure PlacePointerRight;

  Place le pointeur à droite de la fenêtre.

- Boolean procedure PointerInside;

  Indique si la souris est à l'interieur de la fenêtre.

- ref(XWindow) procedure SetCursor(cursor_name);

  Change le curseur. On a deux curseurs disponibles : "xterm", un curseur pour l'edition de texte, et "arrow", ,une flèche,en temps normal.

2.11.4 Ecran

Les fonctions suivantes donnent des informations sur l'écran où apparaissent les fenêtres.

- integer procedure DisplayWidth;
- integer procedure DisplayHeight;

  Taille de l'écran en pixels;

- integer procedure DisplayWidthMM;
- integer procedure DisplayHeightMM;

  Taille de l'écran en millimètres.

- Boolean procedure HighResolutionScreen;

  Indique si on a plus de 87 pixels par pouces (environ 2.54 cm).

- integer procedure ScreenDepth;

  Indique le nombre de bits nécessaires pour coder une couleur.

2.11.5 Buffer de texte

Accès au buffer de texte du systeme. Ceci donne la possibilite d'échange de texte entre différents programmes ou fenêtres.

- text procedure FetchCutBuffer;
- procedure StoreInCutBuffer(t); text t;

3 Les sous-classes de XWindow

Dans cette partie, chaque section décrit une ou plusieurs sous-classes de XWindow. Les premières lignes donnent les entêtes des classes, permettant ainsi de trouver les variables d'initialisation et la super-classe des classes.

3.1 MainWindow

XWindow Class MainWindow(window_name); text window_name;

Les fenêtres de classe MainWindow et SimpleWindow créent automatiquement un display avant leur création (c.f 3.2). Ainsi, une telle fenêtre doit être la première fenêtre que vous créez. De plus, vous ne devez pas avoir plus d'une de ces fenêtres parce que plusieurs display seraient ainsi crées et les messages risqueraient de ne pas être transmis correctement.

3.2 SimpleWindow

MainWindow Class SimpleWindow

Ce type de fenêtre peut être une fenêtre principale au même titre que mainwindow. Noubliez pas, on doit avoir une et une seule fenêtre principale (voir 3.1)

Cette fenêtre permet la création rapide de sous-fenêtres et de leur donner des places automatiquement (c.f 2.2.2). Attention. Si vous y créez une fenêtre `scrollwindow`, le simplewindow s'efforcera de maximiser la taille de cette dernière à chaque changement de taille.

- procedure MakePromptandScrollWindows;

  Création simultanee de deux fenêtres

- text procedure ask_for_text(prompt); text prompt;
- integer procedure ask_for_int(prompt); text prompt;
- character procedure ask_for_char(prompt); text prompt;
- real procedure ask_for_real(prompt); text prompt;
- Boolean procedure ask_for_Bool(prompt); text prompt;

  Ces fonctions créent des promptwindow demandant l'information voulue

- text procedure be_om_text(prompt); text prompt;
- integer procedure be_om_int(prompt); text prompt;
- character procedure be_om_char(prompt); text prompt;
- real procedure be_om_real(prompt); text prompt;
- Boolean procedure be_om_Bool(prompt); text prompt;

  Version norvégienne des 5 dernières fonctions.

- ref(XWindow) procedure MakeEmpty;
- ref(SimpleWindow) procedure outimage;
- ref(SimpleWindow) procedure outtext(t); text t

- ref(SimpleWindow) procedure setpos(pos); integer pos;
- ref(SimpleWindow) procedure outline(line); text line;
- ref(SimpleWindow) procedure outchar(c); character c;
- ref(SimpleWindow) procedure outint(i, w); integer i, w;
- ref(SimpleWindow) procedure outfix(r, d, w); real r; integer d, w;

  Ces fonctions n'affectent que le scrollwindow créé par MakeScrollWindow. Si la fenêtre scrollwindow n'existe pas avant l'appel d'une de ces fonctions, elle sera créee (voir 3.14)

3.3 SideWindow

MainWindow Class SideWindow(TopMainWindow); ref(MainWindow) TopMainWindow;

Ce type de fenêtre permet de créer des fenêtres indépendantes de la fenêtre mère sans causer de problèmes. On peut en créer autant qu'on veut. La variable `TopMainWindow` peut être correctement initialisée par un appel à `topwindow` (c.f 2.11.1).

3.4 GraphPage

SideWindow Class GraphPage

Cette fenêtre contient une fenêtre de classe graphwindow. Pour comprendre l'utilité des fonctions, voir la description de cette classe (3.24).

- procedure DefineCorners(mnx, mny, mxx, mxy); real mnx, mny, mxx, mxy;
- procedure DefineCoordinates(mnx, mny, mxx, mxy); real mnx, mny, mxx, mxy;
- ref(XWindow) procedure DrawLineScaled(x1, y1, x2, y2); real x1, y1, x2, y2;
- ref(XWindow) procedure DrawPointScaled(x, y); real x, y;
- ref(XWindow) procedure MakeEmpty;

3.5 PopupWindow

SideWindow Class PopupWindow

Ce type de fenêtre se crée sans demander à l'utilisateur où on veut la placer.

3.6 PopupMenu

PopupWindow Class PopupMenu(NoticeWindow);ref(XWindow) NoticeWindow

Cette fenêtre permet la sélection d'un item (ligne de texte) à partir d'une fenêtre indépendante des autres.

`NoticeWindow` est la fenêtre à laquelle on renvoie les messages `LineChoice et LineChoiceIn` lors de la sélection et par rapport à laquelle il est possible de se positionner. Etant donné que cette fenêtre n'est pas une sous-fenêtre (SubWindow), `LineChoiceIn` donne "`none`" comme pointeur de fenêtre.

Cette fenêtre est utilisée par MenuButton (c.f. 3.9)

- procedure FindPosition(XNotice,YNotice);integer XNotice,YNotice;

  Positionne la fenêtre relativement à `NoticeWindow`.

- procedure PutLine(Line);text Line;

  Ajoute une ligne de sélection.

- procedure SetDefault(idef);integer idef;

  Décide quelle sera la ligne de sélection par défaut. Cette méthode doit impérativement être appelée une fois que toutes les lignes sont entrées.

3.7 Subwindow

XWindow Class Subwindow(parent); ref(XWindow) parent;

Toute fenêtre faite à partir de cette classe est une sous-fenêtre. C'est à dire qu'elle considère sa fenêtre mère comme un écran en ce qui concerne son positionnement.

- integer procedure label_width;

  Donne la taille du label. Les labels ne servent que dans les sous-fenêtres.

3.8 Button

SubWindow Class Button(heading); text heading;
Button Class FancyButton;

La seule véritable différence entre ces deux classes est la présentation.

On peut recupérer la sélection d'un bouton par les fonctions `Perform`,`PerformIn` et `ClickInButton`.

Le FancyButton est le type de bouton utilisé dans les fenêtres "fancy". Vous pouvez en conclure qu'on ne peut faire `SetFancy` sur un bouton.

- text procedure heading_and_key;

  Donne le texte complet qui sera affiché dans le bouton.

- text procedure current_heading;

  Donne le titre du bouton.

- ref(Button) procedure SetShortcut(k); character k;

  Assigne une touche permettant la sélection rapide du bouton.

- ref(Button) procedure Allow;
- ref(Button) procedure Disallow;
- Boolean procedure IsAllowed;

  Permission et interdiction la sélection du bouton. Ces fonctions changent sa présentation.

- ref(Button) procedure SetActive;
- ref(Button) procedure SetInactive;

  Un bouton "active" ne renvoie pas de message à sa fenêtre mère. Ca peut être intéressant pour avoir un bouton après lequel on attend (utilisation de wait).

- ref(Button) procedure SetActiveDisallow;
- ref(Button) procedure SetInactiveAllow;

  Rend le bouton actif en faisant un `disallow` sur les autres boutons frères, ou le contraire.

- ref(Button) procedure Click;

  Agit comme si on avait cliqué sur le bouton.

- ref(XWindow) procedure Wait;

  Commence une boucle d'attente jusqu'à ce qu'on sélectionne le bouton. Cette boucle est très importante car elle traite les événements venant du programme (c.à.d une boucle appelant handle_event). On peut avoir plus d'une boucle imbriqué à la fois.

- Boolean procedure IsWaiting;

  Dit si le bouton est toujours en attente.

- procedure StopWaiting;

  Force l'arrêt de la boucle d'attente du bouton. Si c'est la dernière boucle dans tout le programme, ce dernier s'arrêtera aussitôt.

3.9 MenuButton

Button Class MenuButton

Ce bouton permet l'utilisation d'un PopupMenu (c.f 3.6) et de se souvenir de sa dernière sélection. En tout temps, il affichera la dernière sélection (ou celle par défaut).

Pour chaque item de menu, il est possible d'attribuer des clefs. La clef de l'item choisi peut être récupéré par `Perform` et `PerformIn`. Sinon, l'item choisi est toujours donné par `LineChoice` et `LineChoiceIn`.

- procedure SetMenu(choix,default);text choix,default;

  Initialisation du menu.

  Le paramètre `choix` contient les différentes entrées du menu. Chaque entrée est soit un item, soit une étiquette suivie d'un item. Dans le deuxième cas, les deux éléments sont séparés par le caractère ':'. L'étiquette servira de clef si elle existe. Sinon, l'entrée servira de clef. Les entrées sont séparées par un séparateur qui est défini comme la première lettre du texte.

Le paramètre `defaut` est une clef dans choix qui représente le choix par défaut du menu. Si `defaut` ne correspond à aucune clef, la première entrée sera la sélection par défaut.

Voici un exemple: `SetMenu(''+t1:premier+deuxieme+t3:troisieme'',''t3'');`

- text procedure CurrentChoice;

  Sélection courante.

3.10 MultipleChoiceWindow

SubWindow Class MultipleChoiceWindow

Cette fenêtre permet la sélection du contenu d'une variable texte.

- ref(MultipleChoiceWindow) procedure AddChoice(t); text t;

  Ajoute une ligne à choisir.

- ref(MultipleChoiceWindow) procedure SetChoice(t); text t;

  Change la sélection courante.

- ref(MultipleChoiceWindow) procedure MakeStepBoxes;

  Ajoute des boutons permettant la navigation à travers les choix.

- procedure PreviousChoice;

  (bouton de gauche)

- procedure FirstChoice;

  (bouton du milieu)

- procedure NextChoice;

  (bouton de droite)

  Navigation à travers les différents choix. La sélection se fait en boucle. Ainsi, le dernier élément précède le premier. Les boutons mentionnés permettent d'appeler ces fonctions avec la souris.

- text procedure CurrentChoice;

  La sélection courante. Par défaut, c'est la première ligne entrée.

3.11 InputWindow

SubWindow Class InputWindow

Cette classe n'est pas un classe à instancier. Elle est abstraite.

Dès que du texte modifié par l'utilisateur, un message `InputInSubWindow` est envoyé à la fenêtre parente.

- ref(InputWindow) procedure LeftJustify;
- ref(InputWindow) procedure RightJustify;
- ref(InputWindow) procedure CenterJustify;

  Centre le texte

- ref(XWindow) procedure AllowInput;
- ref(XWindow) procedure DisallowInput;

  Permission d'entrée d'information venant du clavier ou de la souris.

- ref(InputWindow) procedure NoEcho;
- ref(InputWindow) procedure Echo;

  Détermine si on affiche le texte entŕe ou une série d'étoiles.

- ref(XWindow) procedure MakeEmpty;

  Efface le contenu de la fenêtre.

3.12 ListWindow

InputWindow Class ListWindow
ListWindow Class SelectionListWindow

ListWindow offre deux fonctionnalités exclusifs:

- Edition de texte (traitement de texte simple).
- Sélection de ligne (comme dans un menu).

Si aucune d'entre elles n'est choisie, la fenêtre servira ne fera alors qu'afficher du texte.

3.12.1 Permissions

Ces fonctions décident quelle est la fonctionnalié utilisée.

- ref(XWindow) procedure AllowChoice;
- ref(XWindow) procedure DisallowChoice;

  Permission de sélection de texte.

- ref(XWindow) procedure AllowInput;
- ref(XWindow) procedure DisallowInput;

  Permission d'édition de texte.

SelectionListWindow est une variante de ListWindow qui se comporte différemment. Si vous êtes intréressés qu'à la selection de texte comme dans un menu, cette fenêtre risque d'être la solution. Cette fenêtre sert en fait à la fabriquation des menus déroulants. Un avertissement : le double-click y est détecté. Comme il n'y a pas eu d'établissement d'un protocole de "feedback" dans le cas où une fenêtre reçoit un double-click, la fenêtre renvoie handle_second_click dans la fenêtre parente. Ca peut causer des problèmes si cette fenêtre parente réagit a un double-click.

3.12.2 Ajout de texte

Par ces méthodes, vous pouvez entrer du texte dans la fenêtre.
- ref(ListWindow) procedure outtext(t); text t;
- ref(ListWindow) procedure outint(i,w); integer i,w;
- ref(ListWindow) procedure outchar(C); character C;
- ref(ListWindow) procedure outfix(r, dig, w); real R;integer dig, w;
- ref(ListWindow) procedure PutText(t); text t;
- ref(ListWindow) procedure setpos(pos); integer pos;
- ref(ListWindow) procedure outimage;
- ref(ListWindow) procedure PutImage;

  Edition de texte avec un tampon de texte interne à la fenêtre.

- ref(ListWindow) procedure PutLine(line); value line; text line;

  Ajoute une ligne à la fin.

- ref(ListWindow) procedure PutLineNr(lnr, line);

  Insère une ligne à la ligne lnr.

- ref(ListWindow) procedure PutVisibleLineNr(lnr, line); integer lnr; text line;

  Insere une ligne. "lnr" désigne le numéro de ligne visible à ce moment là.

- ref(ListWindow) procedure InitPutLine(line); value line; text line;

  Ajoute une ligne à la fin sans rafraîchir la fenêtre. Interéssant lors du premier remplissage de la fenêtre. On peut ensuite faire un seul "refresh".

- ref(XWindow) procedure MakeEmpty;

  Vide le contenu de la fenêtre

- Boolean procedure IsEmpty;

  Indique si la fenêtre est vide.

3.12.3 Accès au texte

Ces fonctions nous permettent d'obtenir le contenu du texte de la fenêtre.
- text procedure GetLine(lnr); integer lnr;
- text procedure GetVisibleLine(lnr); integer lnr;

  Donne une ligne de texte. La deuxième fonction commence a compter à partir de la première ligne visible.

- integer procedure info_top;

  Donne le nombre de lignes entrées.

- Text Procedure CurrentText;

  Première ligne sélectionnée (ou ligne selectionnée si on est en mode de sélection).

- Integer procedure Current_Line_nr;

  Numéro de la première ligne sélectionnée (ou ligne selectionnée si on est en mode de sélection).

3.12.4 Modification de la présentation du texte

Ces méthodes forcent la fenêtre à afficher certaines parties tu texte de façon différente.
- procedure InvertSelection;
- ref(ListWindow) procedure InvertLine(wLn); integer wLn;

  Inverse la sélection courante ou une ligne particulière.

- ref(ListWindow) procedure GrayLine(lnr); integer lnr;

  Met une ligne en gris, interdisant la sélection de celle-ci.

- ref(ListWindow) procedure BlackLine(lnr); integer lnr;

  Contraire de grayline

- Boolean procedure gray(n); integer n;

  Indique si la ligne est grise ou non.

3.12.5 Taille de la fenêtre

On a une fonction supplémentaire pour modifier la taille de la fenêtre.
- ref(ListWindow) procedure SetTextSize(ncolumns, nrows); integer ncolumns, nrows;

  Change la taille de la fenêtre pour qu'elle puisse contenir le nombre de rangées (lignes) et de colonnes (nombre de caracteres par ligne) spécifiées.

3.12.6 Scrolling (déplacement)

Ces fonctions servent à nous déplacer dans le texte entré dans la fenêtre.
- ref(ListWindow) procedure ScrollLineUp;
- ref(ListWindow) procedure ScrollLineDown;
- ref(ListWindow) procedure Scroll(n); integer n;

  Va vers le bas de n lignes (n peut être négatif pour remonter).

- ref(ListWindow) procedure ScrollPage(forward); Boolean forward;

  Déplacement de plusieurs lignes

- ref(ListWindow) procedure ScrollToBottom;
- ref(ListWindow) procedure ScrollToTop;

  Déplacement au début et à la fin

- ref(ListWindow) procedure ScrollToLine(line_number); integer line_number;

  Déplacement pour avoir la ligne au centre de la fenêtre.

- ref(ListWindow) procedure ScrollLineTop(line_number); integer line_number;

  Déplacement pour avoir la ligne en haut de la fenêtre

- ref(ListWindow) procedure ScrollTo(fraction); real fraction;

  Fraction doit être entre 0 et 1. C'est la fraction du texte à laquelle on veut se rendre.

3.13 MenuWindow

SubWindow Class MenuWindow

Ce type de fenêtre permet la créationd'un menu à partir de boutons. Tout bouton sélectionné dans cette fenêtre sera rapporté à la fenêtre parente comme s'il lui appartenait (c.f 3.8). Bref, Cette classe permet une extension du nombre de boutons de sa fenêtr parente.
- ref(MenuWindow) procedure SetHorizontal;
- ref(MenuWindow) procedure SetVertical;

  Décision de l'arrangement des boutons qui seront ajoutés ensuite.

- ref(Button) procedure MakeButton(heading); text heading;

  Création d'un bouton. La sélection future d'un tel bouton sera transmise à la fenêtre mère comme si le bouton lui appartenait directement. La sélection d'un bouton est récupérable par `Perform` et `ClickInButton`.

3.14 ScrollWindow

SubWindow Class ScrollWindow

Cette fenêtre permet la sélection et l'édition de texte. Lorsqu'une ligne est sélectionnee, la fenêtre renvoie deux messages à la fenêtre mère : `linechoice` et `linechoicein`.

- ref(ScrollWindow) procedure MakeWaitButton(heading); text heading;

  Crée un bouton qui permettera d'attendre après une réponse.

- ref(XWindow) procedure Wait;

  Fait attendre la fenêtre après une réponse. Ne fonctionne que si un bouton a été créé par MakeWaitButton (c.f 3.8).

- ref(ScrollWindow) procedure MakeHideButton;

  Crée un bouton qui permettera d'ignorer la fenêtre.

- ref(ScrollWindow) procedure SetHeadTypeface(typeface, size, bold, slanted); text typeface; integer size; Boolean bold, slanted;

  Fonction supplémentaire pour changer la fonte de la fenêtre titre.

Les trois prochaines méthodes servent à manipuler le menu venant avec le scrollwindow (c.f 3.13).

- ref(ScrollWindow) procedure SetHorizontal;
- ref(ScrollWindow) procedure SetVertical;
- ref(Button) procedure MakeButton(heading); text heading;

Les prochaines méthodes (mise à part les deux premières) sont très très proches de celles de `listwindow` (c.f 3.12 pour des descriptions).

- ref(ScrollWindow) procedure RefreshLines;

  Fait un "refresh" pour les lignes de texte. A appeler après avoir fait des appels à `InitPutLine`.

- integer procedure GetNumberOfLines;

  Donne le nombre de lignes entrées dans la fenêtre. (très proche de `info_top` de `listwindow`).

- ref(XWindow) procedure AllowInput;
- ref(XWindow) procedure DisallowInput;
- ref(XWindow) procedure AllowChoice;
- ref(XWindow) procedure DisallowChoice;
- ref(ScrollWindow) procedure InitPutLine(line); value line; text line;
- ref(ScrollWindow) procedure GrayLine(lnr); integer lnr;
- ref(ScrollWindow) procedure BlackLine(lnr); integer lnr;
- ref(ScrollWindow) procedure PutLine(line); text line;

- ref(ScrollWindow) procedure PutLineNr(lnr, line); integer lnr; text line;
- ref(ScrollWindow) procedure PutVisibleLineNr(lnr, line); integer lnr;text line;
- ref(ScrollWindow) procedure outline(line); text line;
- ref(ScrollWindow) procedure outtext(t); text t;
- ref(ScrollWindow) procedure outchar(c); character c;
- ref(ScrollWindow) procedure outint(i, w); integer i, w;
- ref(ScrollWindow) procedure outfix(r, d, w); real r; integer d, w;
- ref(ScrollWindow) procedure PutText(t); text t;
- ref(ScrollWindow) procedure setpos(pos); integer pos;
- ref(ScrollWindow) procedure outimage;
- ref(ScrollWindow) procedure PutImage;
- text procedure GetLine(lnr); integer lnr;
- text procedure GetVisibleLine(lnr); integer lnr;
- ref(ScrollWindow) procedure ScrollLineUp;
- ref(ScrollWindow) procedure ScrollLineDown;
- ref(ScrollWindow) procedure ScrollToTop;
- ref(ScrollWindow) procedure ScrollToBottom;
- ref(ScrollWindow) procedure ScrollToLine(line_number); integer line_number;
- ref(ScrollWindow) procedure ScrollPage(forward); Boolean forward; procedure ScrollTo(q); real q;

3.15 MenuScrollWindow

ScrollWindow Class MenuScrollWindow
MenuScrollWindow Class DirectoryMenu(DirName);text DirName;

Ces deux classes permettent la sélection d'une ligne de texte qu'elles rapportent par `LineChoice` et `LineChoieIn`. On peut aussi obtenir la réponse par `WaitMenu`.

DirectoryMenu n'a pas besoin d'être initialisé en aucune façon. Il suffit de donner un répertoire par `DirName` et la fenêtre se remplira d'elle-même. Un nom de répertoire incorrect (`none` par exemple), causera la lecture du répertoire courant. Elle donnera un nom de fichier complet (i.e répertoire inclu) comme réponse. Un exemple d'utilisation de cette classe se trouve en 4.2.

- procedure SetMenu(question,choix,default);text question,choix,default;

  Initialise le menu. Les rôles de `choix` et de `defaut` sont bien expliqués dans la description de `SetMenu` de la classe MenuButton (c.f. 3.9).

- text procedure WaitMenu;

  Attend après une réponse avant de continuer l'exécution du programme. C'est une fonction très similaire au `Wait` de la classe Button (c.f 3.8).

- text procedure menu(question,choix,default);text question,choix,default;

  Exécute `SetMenu`, puis `WaitMenu`.

3.16 ItemWindow

InputWindow Class ItemWindow
ItemWindow Class ItemWindow IntItemWindow
ItemWindow Class ItemWindow CharItemWindow
ItemWindow Class ItemWindow BoolItemWindow
ItemWindow Class ItemWindow IntItemWindow
ItemWindow Class ItemWindow TextItemWindow

Itemwindow est une classe abstraite d'une famille de fenêtres pouvant receuillir des types de données.

Chaque sous-classe ont leur `put` et `get`. Il suffit d'ajouter leur prefixe distinctif à la fin de ces commandes. Ainsi un `intitemwindow` possède les méthodes `procedure putint(i);integer i;` et `integer procedure getint;`

Le intitemwindow possède une fonction supplémentaire permettant de mettre des bornes aux réponses possibles:

`ref(IntItemWindow) procedure setlimits(mn, mx); integer mn, mx;`

Pour le `boolitemwindow`, `makeempty` affiche `false`, alors que pour les autres fenêtres, on vide.

Si la fenêtre est vide lorsqu'on lui demande son entrée, la réponse sera "false","caractère nul","0","0.0" ou "notext", en fonction du type de la sous-classe.

3.17 ParameterList

SubWindow Class ParameterList

C'est une fenêtre qui peut contenir une collection de fenêtres (voir 3.16). Elle va gérer leur création et leur positionnement. Les différentes fonctions créent différents types de fenêtre d'items. Elles vont toujours placer la dernière fenêtre en dessous des autres.

Les paramètres donnés ont la meme signification.

lb : pour label qui est le titre de la fenêtre
init : la valeur initiale à y mettre
nc : la largeur de la fenêtre en caractères.

- ref(IntItemWindow) procedure MakeIntPara(lb, init, nc); text lb; integer init, nc;
- ref(BoolItemWindow) procedure MakeBoolPara(lb, init); text lb; Boolean init;
- ref(CharItemWindow) procedure MakeCharPara(lb, init); text lb; character init;
- ref(RealItemWindow) procedure MakeRealPara(lb, init, nd, nc); text lb; real init; integer nd, nc;
- ref(TextItemWindow) procedure MakeTextPara(lb, init, nc); text lb, init; integer nc;

3.18 ParameterWindow

SubWindow Class ParameterWindow(para_label); text para_label;
ParameterWindow Class IntParameterWindow(nc);integer nc;
ParameterWindow Class CharParameterWindow
ParameterWindow Class BoolParameterWindow

ParameterWindow Class RealParameterWindow(nc);integer nc;
ParameterWindow Class TextParameterWindow(nc);integer nc;

Cette classe abstraite permet l'arrivée d'une autre famille de fenêtre paramètre (voir 3.16).

Les sous-classes intparameterwindow,realparameterwindow et textparameterwindow prennent une variable d'initialisation supplémentaire : c'est un entier qui détermine le nombre maximum de caractères que ces fenêtres peuvent contenir.

La véritable différence ici, c'est que ces fenêtres ont un bouton "ok" permettant ainsi une forme de confirmation. Tant que le bouton ok n'a pas été cliqué, la valeur affichée n'est pas considèrée.

Chaque sous-classe permet l'accès et la modification de sa valeur grâce à `current_value` et `set_current_value`. La valeur de retour de current_value et le paramètre de set_current_value seront de type correspondant à la sous-classe. Ainsi, un IntParameterWindow aurait les fonctions `integer procedure current_value` et `procedure set_current_value(val);integer val;`

3.19 ParameterMenu

SubWindow Class ParameterMenu

Comme pour listwindow (voir 3.12), parametermenu permet de manipuler une collection de fenêtres contenant des paramètres (voir 3.18).

Les paramètres donnes ont la même signification. lb : pour label qui est le titre de la fenêtre nc : la largeur de la fenêtre en caractères.

- ref(IntParameter) procedure MakeIntPara(lb, nc); text lb; integer nc;
- ref(BoolParameter) procedure MakeBoolPara(lb); text lb;
- ref(CharParameter) procedure MakeCharPara(lb); text lb;
- ref(RealParameter) procedure MakeRealPara(lb, nc); text lb; integer nc;
- ref(TextParameter) procedure MakeTextPara(lb, nc); text lb; integer nc;

3.20 PromptWindow

SubWindow Class PromptWindow

Cette fenêtre sert à poser une question nécessitant qu'une seule valeur comme réponse. Elle offre la possibilité de s'agrandir toute seule si la fenêtre de réponse est pleine.

- ref(PromptWindow) procedure MakeOKButton;
- ref(PromptWindow) procedure MakeCancelButton;
- ref(PromptWindow) procedure MakeClearButton;
- ref(PromptWindow) procedure MakeHideButton; Création de boutons de contrôle. Il faut au moins avoir le OkButton.
- ref(PromptWindow) procedure Echo;
- ref(PromptWindow) procedure NoEcho;

Il est possible de demander à la fenêtre de ne pas afficher à l'écran la valeur donnée par l'utilisateur

Les méthodes commençant par "be_om" sont les versions norvégiennes de celles commenç par "ask_for". Voici la signification de leur paramètres:

prompt: la question demandée (ou le label)
default: la valeur donnée par défaut
max_length: la nombre maximum de caractères accpetables.

- Boolean procedure ask_for_Bool(prompt); text prompt;
- text procedure ask_for_text(prompt); text prompt;
- text procedure ask_for_text_def(prompt, default); text prompt, default;
- text procedure ask_for_text_lim(prompt, max_length);
- text procedure ask_for_text_def_lim(prompt, default, max_length);
- integer procedure ask_for_int(prompt); text prompt;
- integer procedure ask_for_int_def(prompt, default);
- real procedure ask_for_real(prompt); text prompt;
- character procedure ask_for_char(prompt); text prompt;
- character procedure ask_for_char_def(prompt, default);
- Boolean procedure be_om_Bool(prompt); text prompt;
- text procedure be_om_text(prompt); text prompt;
- integer procedure be_om_int(prompt); text prompt;
- real procedure be_om_real(prompt); text prompt;
- character procedure be_om_char(prompt); text prompt;

3.21 RadioPanel

SubWindow Class RadioPanel

Le radiopanel est encore une autre forme de sélection. Il comporte plusieurs boutons. Un seul peut être sélectionné à la fois. Tous les boutons doivent avoir un texte différent.

La sélection d'un bouton par l'usager fait envoyer à la fenêtre mère un message `ClickInRadioPanel`.

- ref(RadioPanel) procedure SetHorizontal;
- ref(RadioPanel) procedure SetVertical;

  Décison si les boutons seront placés verticalement ou horizontalement.

- text procedure CurrentChoice;

  Texte du bouton choisi

- ref(RadioPanel) procedure AddButton(t); text t;

  Ajoute un bouton.

- ref(RadioPanel) procedure SetButton(t); text t;

  Sélectionne le bouton avec le texte t.

- ref(RadioPanel) procedure AddSetButton(t); text t;

  Ajoute et sélectionne un bouton

3.22 MessageWindow

SubWindow Class MessageWindow

Ces fonctions créent des fenêtres de messages. Attention. Lorsque ces fenêtres ont fini leur travail, elles sont détruites. Tout pointeur dessus serait ainsi érroné, une fois disparues de l'écran.

- ref(MessageWindow) procedure Message(mess); text mess;

  Création d'une fenêtre message.

- ref(MessageWindow) procedure MessageWait(mess, time); text mess; integer time;
- ref(MessageWindow) procedure TimedMessage(mess, time); text mess; integer time;

  Les deux dernières fonctions sont identiques. Elles créent une fenêtre message qui se détruit au bout d'un laps de temps. Si le temps demandé est negatif (ou nul), la fenêtre sera instantanément détruite.

3.23 Drawwindow

SubWindow Class DrawWindow

Ce type de fenêtre permet de dessiner à léchelle. Il suffit de donner les échelles avec `SetScales` et d'utiliser les méthodes de dessin propres à cette classe.

- ref(XWindow) procedure SetScales(xmn, ymn, xmx, ymx, imn, jmn, imx, jmx); real xmn, ymn, xmx, ymx; integer imn, jmn, imx, jmx;

  Détermine les valeurs minimales et maximales des coordonnées dessinées par les fonctions de dessin suivants, puis les coordonnées du point d'origine, suivi du coin opposé à l'origine (en pixels), dans lequel on dessinera.

- ref(XWindow) procedure DrawLineScaled(x1, y1, x2, y2); real x1, y1, x2,y2;
- ref(XWindow) procedure DrawPointScaled(x, y); real x, y;
- ref(XWindow) procedure DrawCircleScaled(cx, cy, r); real cx, cy, r;
- ref(XWindow) procedure DrawArcScaled(x1, y1, x2, y2, ang1, ang2); real x1, y1, x2, y2, ang1, ang2;
- ref(XWindow) procedure DrawTextScaled(x, y, t, hpos, vpos); real x, y; text t; integer hpos, vpos;

  Fonctions permettant de dessiner à l'echelle.

- integer procedure hor_scale(x); real x;
- integer procedure ver_scale(y); real y;
- real procedure hor_invscale(i); integer i;
- real procedure ver_invscale(j); integer j;

  Convertir des coordonnées virtuelles en réelles (pixels), ou le contraire.

- integer procedure hor_relscale(x); real x;

- integer procedure ver_relscale(y); real y;

  Convertir une distance virtuelle en réelle (pixel).

3.24 Graphwindow

DrawWindow Class GraphWindow

Ce type de fenêtre permet deux modes de dessin definis par les deux méthodes suivantes.

- procedure DefineCorners(mnx, mny, mxx, mxy); real mnx, mny, mxx, mxy;

  Définit le point d'origine en bas à gauche et s'assure que toute la fenêtre servira à dessiner.

- procedure DefineCoordinates(mnx, mny, mxx, mxy); real mnx, mny, mxx, mxy;

  Transforme la fenêtre en graphe avec repère cartésien et trace des lignes de coordonnées. On ne dessinera pas dans la zone reservée à ces lignes.

4 Code

Dans cette partie, on donne du code en exemple. Il est important de noter qu'il existe plusieurs fichiers de code que vous pouvez aussi regarder librement. Il devrait en exister une liste tout près de la provenance de ce fichier.

4.1 Hello World

Ce code sert surtout à montrer à quel point il faut peu de code pour afficher “Hello World” à l'écran avec windowtools.

```
begin
external class windowtools;   !appel de la librairie;

   inspect new simplewindow(''Test'') do !Creation d'une fenetre principale;
      begin
         ref(button) quit;
         quit :- MakeButton(''Quit'');    !Creation du bouton de sortie;
         DrawCenterText(''Hello World''); !Affichage du texte;
         quit.wait;   !Debut de la boucle d'attente d'evenements;
      end;
end.
```

4.2 Edition de texte

Ce code-ci est un éditeur de texte. Le programme demande un nom de fichier à l'usager et l'éditera si le fichier existe.

```
begin
   external class windowtools;

   simplewindow class MyWindow;
   begin
      ref(button) quit,save;   !boutons de sortie et de sauvegarde du fichier;
      ref(scrollwindow) edit;  !fenetre d'edition;
      text filename;           !nom du fichier d'entree et/ou sortie;

      boolean procedure readfile(fname);TEXT fname;
      !Procedure qui  lit un fichier et le met dans edit et qui
       indique a la fin si le fichier demande existe;
      begin
         ref(InFile) f;
         boolean opened;
         f:-NEW InFile(filename);
         opened := f.Open(Blanks(500));
         IF opened then
            begin

               f.InImage;
               while not f.Endfile DO  !On met le fichier ligne par
                                        ligne dans edit;
                  begin
                     edit.initputline(f.Image.strip);
                     f.InImage;
                  end;
               edit.refreshlines;    !On rafra\^{\i}chit la fen\^etre;
               f.Close;
            end;

         readfile:=opened;
      end;

      procedure savefile(fname); text fname;
      !Sauve le contenu de edit dans le fichier;
      begin
         integer i,nlines=20;
         text t;
         ref(OutFile) output;
         output:-new OutFile(fname);
         output.Open(Blanks(500));
         for i:=1 step 1 until edit.GetNumberOfLines do
            begin
```

```
         t:-edit.getline(i);
         output.OutText(t);
         output.OutImage;
      end;
   output.Close;
   MakeMessageWindow.TimedMessage(``File Saved'',3);
end;


procedure createnewfile;
begin
   filename :- ask_for_text(``Filename'');  !Nouveau fichier;
   if filename=/=notext then  !ouverture du fichier;
   begin
      if ReadFile(filename) then
         MakeMessageWindow.TimedMessage(``File Exists'',3);

      edit.allowinput;
      save.allow;
   end;
end;

PROCEDURE opennewfile;
begin
   ref (DirectoryMenu) menu;
   menu :- new directorymenu(this XWindow,notext);
   menu.setsize(400,300);
   menu.placecentered;
   menu.show;
   filename :- menu.Waitmenu;
   if filename =\= notext and then  ReadFile(filename) then
      begin               !ouverture du fichier;
         edit.allowinput;
         save.allow;
      end;
   menu.destroy;          !destruction du menu inutile;
end;

procedure ClickInButton(b);ref(button) b;
!Recupere l'enfoncement du bouton enfonce et agit en consequence;
if b==save then savefile(filename)
else
if b==newfile then createnewfile
else
```

```
      if b==openfile then opennewfile;

      quit :- makebutton(``Quit'');             !creation des sous-fenetres;
      save :- makebutton(``Save'').Disallow;    !interdiction de sauver
                                               avant d'avoir un nom de fichier;

      newfile:-makebutton(``New File'');        !boutons d'ouverture de fichier;
      openfile:-makebutton(``OpenFile'');
      edit :- makescrollwindow;
      filename :- ask_for_text(``Filename'');   !demande du nom de fichier;

      if not ReadFile(filename) then          !ouverture du fichier;
         MakeMessageWindow.TimedMessage(``Nouveau Fichier'',3);

      edit.allowinput;         !permission pour editer le contenu de edit;
      save.allow;              !permission de sauver;
      quit.Wait;               !boucle d'attende d'evenement;
   end;

   new Mywindow(``Edit File''); !creation de la fenetre principale;
end.
```