MASTER EN SCIENCES INFORMATIQUES

Didacticiel de fonctions mathématiques simples et leurs représentations graphiques

Nguyen, Sivilai

Award date:
1996

Awarding institution:
Universite de Namur

FACULTES UNIVERSITAIRES NOTRE-DAME DE LA PAIX - NAMUR

FUNDP

INSTITUT D'INFORMATIQUE
Rue Grandgagnage 21 - 5000 Namur
Tél. : 081 / 72 49 83

Didacticiel de fonctions mathématiques simples et leurs représentations graphiques

Mémoire présenté en vue de l'obtention
du titre de Licenciée et Maître en Informatique

Promoteur
M. Claude CHERTON

Septembre 1996

Sivilai NGUYEN

Remerciements

Tout d'abord, je voudrais remercier le C.E.T.D. (Centre d'Enseignement et de Traitements Différenciés), en particulier Madame Yvette Heyters et Madame Chantal Blomart de m'avoir consacré une partie de leur temps pour discuter de mon sujet de mémoire. Je voudrais également remercier Paul, un élève du centre, de m'avoir fait découvrir son univers de travail.

Ensuite, je tiens à remercier mon promoteur de mémoire, Monsieur Claude Cherton de m'avoir consacré une grande partie de son temps pour la réalisation de ce mémoire. Ses conseils précieux et judicieux, ses remarques pertinentes et ses encouragements m'ont énormément aidée tout au long de mon travail.

Enfin, je voudrais faire un clin d'oeil à ma famille, en particulier à mes parents, ma soeur et mon frère pour leur compréhension, leur soutien constant et leur affection tout au long de mes années d'études.

Plan

1. Introduction

Présentation du sujet du mémoire : poser le problème, le situer dans son contexte et définir des conditions de travail.

2. Le projet

Définition du logiciel : quel est son objectif, quelles sont les lignes directrices qui nous guideront lors de son élaboration et quel type d'activités offrira-t-il à l'utilisateur ?

3. Analyse fonctionnelle

Etude complète des différentes fonctionnalités du logiciel sous différents aspects : l'aspect mathématique, graphique, interactif et handicap d'une part, l'interface et les aides du logiciel d'autre part.

4. Analyse conceptuelle

Proposition d'une architecture logique du projet en se basant sur l'étude des fonctionnalités : définition et description des différentes unités et fonctions du logiciel.

5. Implémentation

Implémentation orientée objet du projet : définition et description des différents objets de notre logiciel. Ces objets seront accompagnés de leurs attributs et de leurs méthodes.

6. Création de l'interface

Dans un premier temps, présentation visuelle et définition des fonctionnalités de quelques fenêtres du logiciel et dans un second temps l'enchaînement de quelques unes d'entre-elles.

7. Conclusion

8. Bibliographie

9. Table des matières

10. Table des figures

11. Annexes

Les annexes concernant l'analyse mathématique et graphique du logiciel d'une part, et les explications du terme « infirmité motrice cérébrale » (I.M.C.) d'une part.

12. Code source

Listing du code source du programme (à l'état actuel et sans commentaire).

1. Introduction

1. Introduction

1.1 Le contexte du mémoire

1.2 L'objectif du mémoire

1.3 Le C.E.T.D.

1.4 Les futurs utilisateurs

1.5 Les contraintes générales

1.6 L'approche Orientée Objet

1.7 Le langage de programmation

1.1 Le Contexte du mémoire

Nous sommes dans une classe en secondaire supérieur. La matière enseignée est l'étude de *fonctions réelles à une variable.* La base de ces études de fonctions est la résolution d'exercices.

La difficulté majeure dans ces études de fonctions est la compréhension des liens existant entre une fonction algébrique et sa représentation graphique. Comment expliquer et faire comprendre ces liens aux élèves ?

Les enseignants de mathématiques du Centre d'Enseignement et de Traitements Différenciés (C.E.T.D.) sont également confrontés à ce problème et qui plus est de l'enseigner à des élèves *infirmes moteur d'origine cérébrale*[1] (I.M.C.).

Selon nous, une des façons de bien comprendre ces liens serait de manipuler directement les fonctions mathématiques et de voir les effets sur leurs graphes et inversement. Pour cela, nous pensons que l'approche informatique pourrait aider les élèves dans leur travail. En effet, d'une part, l'enseignement assisté par ordinateur (E.A.O.) pourrait apporter divers avantages sur le plan pédagogique et d'autre part, l'ordinateur pourrait être considéré comme un outil de support très utile pour les élèves handicapés moteur.

Quels sont les avantages apportés par les didacticiels ?

- Les didacticiels permettent une plus grande autonomie et individualisation du choix de l'apprenant. Celui-ci a la possibilité de se former à son rythme, en fonction de sa disponibilité, sur le sujet qui l'intéresse ou qu'il doit approfondir.

- L'acquisition des connaissances se fait de manière interactive. Les didacticiels permettent de revenir en arrière et de demander des informations à toute question ou action réalisée par l'apprenant tout en demandant sa participation active.

- L'aspect ludique de certains didacticiels peut inciter l'apprenant à vouloir apprendre.

- La combinaison de différentes techniques multimédias peut favoriser le processus de mémorisation. Ainsi une présentation vidéo combinant textes écrits et parlés, graphiques et images constituent différentes formes de mémoire. Et le fait de répéter plusieurs fois la même chose en est une également.

- L'apprenant suit un processus d'apprentissage et peut à tout moment contrôler ses connaissances et ainsi tester ses performances.

Les didacticiels peuvent cependant atténuer la relation particulière entre l'enseignant et l'enseigné. C'est pourquoi, une combinaison des deux : l'auto-apprentissage des élèves à travers des didacticiels dans un contexte scolaire pourrait préserver cette relation privilégiée entre l'enseignant et l'apprenant.

[1] Voir Annexe A.

1.2 L'objectif du mémoire

Le but de ce mémoire est donc d'essayer d'apporter une solution à ce problème d'étude de fonctions et de l'implémenter. L'implémentation du logiciel doit poursuivre deux objectifs, il doit être pratique et didactique. Pratique dans le sens où le logiciel est avant tout destiné à des élèves handicapés moteur, nous devons donc penser à l'environnement de travail de ces élèves qui n'est pas le même que celui d'un élève normal (nous pensons notamment aux problèmes de saisie de données). Et didactique dans le sens où le logiciel doit se dérouler dans un cadre pédagogique, ce qui signifie qu'il doit poursuivre une fonction d'enseignement.

La réalisation de ce mémoire se fera en deux parties: l'analyse du problème et le développement du logiciel proprement dit.

La partie concernant l'analyse du problème a pour objet de bien définir les besoins de l'utilisateur. Ces besoins constitueront pour nous l'objectif à atteindre. Sur base de cet objectif, nous tracerons les lignes directrices qui nous permettront d'élaborer une étude complète des fonctionnalités du logiciel. Cette étude des fonctionnalités abordera les différents aspects entrant en ligne de compte lors de la réalisation du logiciel, à savoir l'aspect mathématique, l'aspect graphique, l'aspect interactif, sans oublier l'interface et les aides du logiciel. Un autre problème que nous devons aborder dans cette analyse est celui du balayage. Le balayage est destiné aux élèves ne disposant pas de l'usage de la souris. Il doit leur permettre d'utiliser avec autant d'aisance que possible tout le logiciel.

1.3 Le C.E.T.D.

Le Centre d'Enseignement et de Traitements Différenciés (C.E.T.D.) est un institut médico-pédagogique. Ce centre accueille une centaine d'enfants et d'adolescents âgés de 1 à 21 ans qui souffrent d'affections neurologiques (I.M.C.).

Ce centre est également doté d'un service informatique qui a pour objet de favoriser l'utilisation de nouvelles technologies par le développement de logiciels, par la réalisation d'interfaces, par la maintenance du matériel mais surtout par la formation des élèves à la maîtrise de ces nouveaux outils.

1.4 Les futurs utilisateurs

Les premiers utilisateurs du logiciel seraient les élèves du C.E.T.D. Mais étant donné le caractère didactique de ce dernier, nous pourrions aussi bien envisager son utilisation par une personne normale que par une personne handicapée moteur.

1.5 Les contraintes générales

Le logiciel doit fonctionner sur un PC sous MS-Windows version 3.1 ou version ultérieure.

1.6 L'approche Orientée Objet

L'objectif de notre projet étant la réalisation d'un outil didactique, l'aspect interactif y prendra une place primordiale. La conception de notre logiciel doit par conséquent avoir une approche non séquentielle mais bien dynamique des choses et nous pensons que l'approche Orientée Objet se prête bien à notre idée.

En effet, l'approche d'analyse Orientée Objet fait appel aux principes de base suivants :

- Le monde est modélisé sous forme d'objets, d'attributs des objets et de relations entre objets.
- Les objets sont organisés en classes et héritent des propriétés de leur classe et des classes de niveau supérieur.
- Les propriétés comportementales des objets sont modélisées et encapsulées à l'intérieur des objets.
- Le mécanisme d'échange d'informations entre objets est unifié : il s'effectue sous la forme de messages échangés entre un objet émetteur et un ou plusieurs objets récepteurs.

1.7 Le langage de programmation

Le langage de programmation que nous avons choisi pour implémenter notre logiciel est le langage **Borland Pascal pour Windows version 7.0**.

La programmation orientée objet est la suite logique de l'approche d'analyse orientée objet et introduit les concepts de base suivants :

- Les ***objets***[2], qui sont semblables à des enregistrements mais qui contiennent leurs propres procédures et fonctions, appelées « méthodes ».
- Les ***types d'objet***[3], qui définissent des types de données objet et leurs capacités.
- L'***encapsulation***[4], qui relie des procédures et des fonctions dans une structure avec les données qu'elles traitent.
- Les ***messages***[5], qui sont des instructions envoyées aux objets leur demandant d'exécuter leurs méthodes prédéfinies.
- L'***héritage***[6], par lequel les caractéristiques d'un type objet sont transmises à ses types objet héritiers.
- Le ***polymorphisme***[7], qui est la capacité pour des objets différents de répondre, à leur manière propre, à la même commande de programme.
- ***L'extensibilité du code***[8], par laquelle des unités déjà compilées de types objet peuvent être utilisées comme base pour créer et utiliser de nouveaux types objet qui étaient inconnus lors de la compilation.

[2] BORLAND PASCAL 7 *par la pratique*, Scott D. Palmer, Sybex 1992 - Page 529
[3] Idem - Page 529
[4] Idem - Page 529
[5] Idem - Page 529
[6] Idem - Page 529
[7] Idem - Page 530
[8] Idem - Page 530

2. *Le projet*

2. Le projet

2.1 L'objectif du logiciel

2.2 Les lignes directrices

2.3 Les activités proposées

2.4 Illustration des activités proposées

2.1 L'objectif du logiciel

L'objectif de notre didacticiel est d'aider les élèves à comprendre et à assimiler le plus aisément possible les notions de base relatives à quelques familles de fonctions simples ainsi qu'à leurs représentations graphiques.

Les familles envisagées sont :

- Les fonctions linéaires de la forme $y = ax + b$;
- Les fonctions du second degré de la forme $y = ax^2 + bx + c$;
- Certaines fonctions trigonométriques :

  - $k * \sin(ax+b)$
  - $k * \cos(ax+b)$
  - $k * \text{tg}(ax+b)$

Nous nous sommes restreints à l'étude de ces quelques familles de fonctions pour trois raisons :

- La première concerne directement les élèves du C.E.T.D. En effet, de ce qui est sorti de notre discussion avec les professeurs du centre, c'est ce que les élèves voient principalement dans leur cours de mathématiques.

La deuxième concerne notre objectif en général. Nous pensons que pour l'atteindre, partir de ces trois types de fonctions peut constituer une bonne base à la compréhension des élèves du secondaire en général, ainsi qu'à celle des élèves handicapés moteur en particulier. Pour aller plus loin, il est vrai qu'il serait intéressant d'envisager l'étude d'autres familles de fonctions telles que les fonctions rationnelles. Mais nous pensons que cela pourrait faire l'objet d'un autre sujet de mémoire vu l'étendue des possibilités envisageables.

- Enfin, la troisième raison concerne le délai de temps qui nous est réservé pour la réalisation de ce mémoire.

2.2 Les lignes directrices

Nous recherchons avant tout à donner une idée intuitive aussi claire que possible du lien existant entre une fonction exprimée algébriquement et sa représentation graphique. Pour ce faire, nous chercherons à donner une vue aussi dynamique que possible des choses. En particulier, nous essayerons que les modifications apportées aux paramètres définissant algébriquement une fonction soit directement observable sur le graphique correspondant. De même, nous tenterons d'autoriser certaines transformations effectuées directement sur le graphe en montrant leur effet sur l'expression algébrique.

Il est clair que l'un des avantages essentiels de l'emploi de l'ordinateur dans ce contexte est sa capacité d'interactivité. Cependant, le logiciel devant être utilisé par des handicapés moteur, l'une des priorités que nous chercherons à observer est de minimiser les manipulations nécessaires à son emploi. Pour ce faire, nous veillerons à :

1. Réaliser une interface aussi conviviale que possible.

2. Limiter la généralité des expressions étudiées chaque fois que cela ne nuit pas à la compréhension de l'élève.

3. Eviter les possibilités d'interaction qui n'apportent pas vraiment quelque chose sur le plan didactique.

Une interface adaptée à un handicapé ne l'étant souvent pas pour un autre, il serait donc intéressant de permettre une configurabilité du logiciel aussi souple que possible.

2.3 Les activités proposées

Le type d'activité que nous voulons proposer à l'élève est la résolution d'exercices suggérés par le logiciel. Ce seront donc des exercices de nature dirigée, c'est-à-dire que l'élève n'aura pas la possibilité d'aller éditer lui-même la fonction à étudier mais devra choisir un des énoncés présentés par le logiciel pour commencer son exercice.

Lors d'un exercice, ce que nous voulons faire apparaître à l'écran sont les valeurs des paramètres de l'équation algébrique, les propriétés mathématiques qui lui sont liées et son dessin graphique. Ceci dans le but de montrer le lien dynamique qui existe entre les valeurs des paramètres d'une équation algébrique, le calcul de ses propriétés mathématiques et son dessin.

Les différents types d'interactions lors d'un exercice seront les suivants :

- Affichage de l'équation, de ses paramètres, de ses propriétés mathématiques et de son dessin graphique correspondant.
- Modification des valeurs des paramètres avec répercussion sur les propriétés mathématiques et sur le dessin graphique.
- Modification du graphe avec répercussion sur les paramètres et les propriétés mathématiques.
- Modification de l'équation avec répercussion sur les paramètres, les propriétés mathématiques et le graphe.

Il est donc important que l'affichage de l'équation, des paramètres, des propriétés mathématiques et du dessin graphique soit continuellement présente et visible à l'écran.

Pour réaliser un exercice, nous proposerons à l'élève de :

- ♦ Soit commencer un nouvel exercice
- ♦ Soit continuer un exercice existant

Et à tout moment, il aura la possibilité :

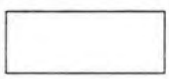
- ♦ D'enregistrer l'exercice en cours sur disque ou disquette
- ♦ De terminer l'exercice et quitter le programme

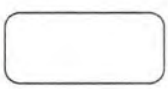
2.4 Illustration des activités proposées

2.4.1 Légende des schémas

Les schémas que nous allons présenter se lisent de la manière suivante :

Une flèche indique une action effectuée soit par l'utilisateur soit par le programme

Un rectangle  indique une opération effectuée par l'utilisateur

Un rectangle arrondi  indique une opération effectuée par le programme

Remarque : Le terme « utilisateur » sous-entend aussi bien l'élève que l'enseignant.

2.4.2 Commencer un nouvel exercice

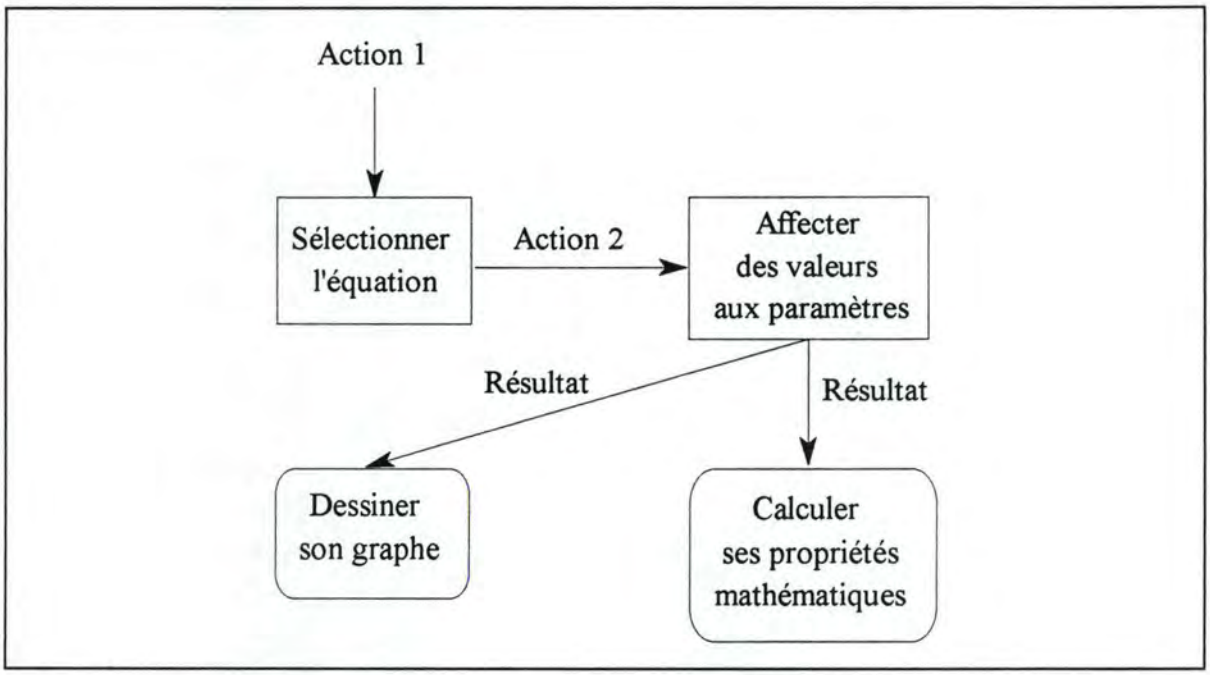


Figure 2.1 : Déroulement d'un nouvel exercice

Action 3 : Modifier les valeurs des paramètres de l'équation étudiée

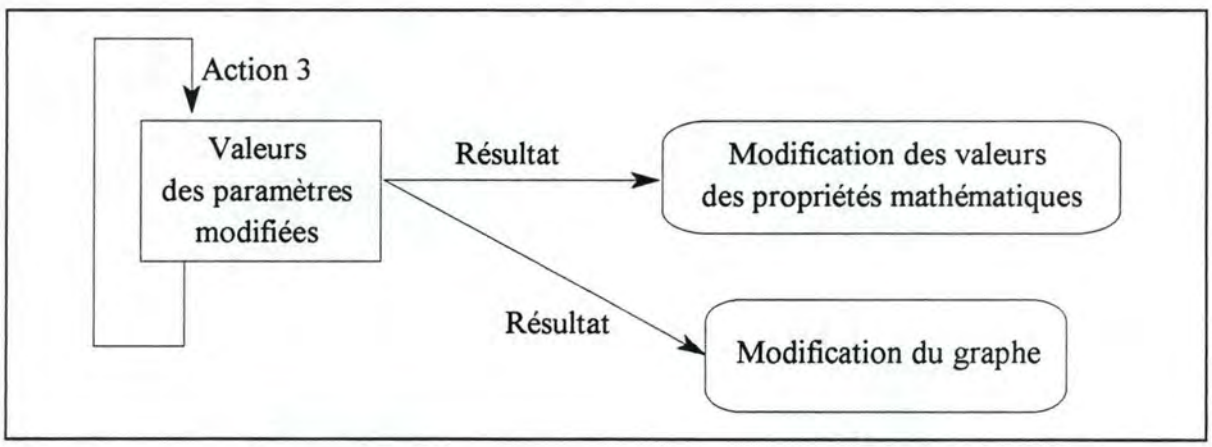


Figure 2.2 : Déroulement lors d'une modification des valeurs des paramètres

2.4.3 Continuer un exercice existant

Action 1 : Aller rechercher le fichier d'exercice

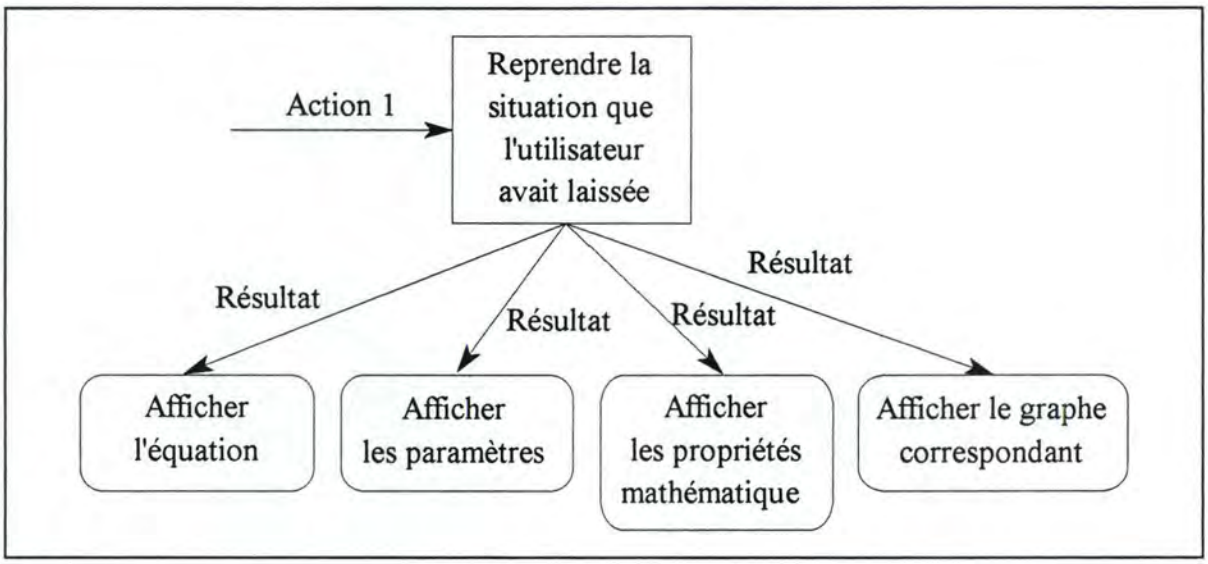


Figure 2.3 : Reprise d'un exercice

Action 2 : Modifier les valeurs des paramètres de l'équation en cours

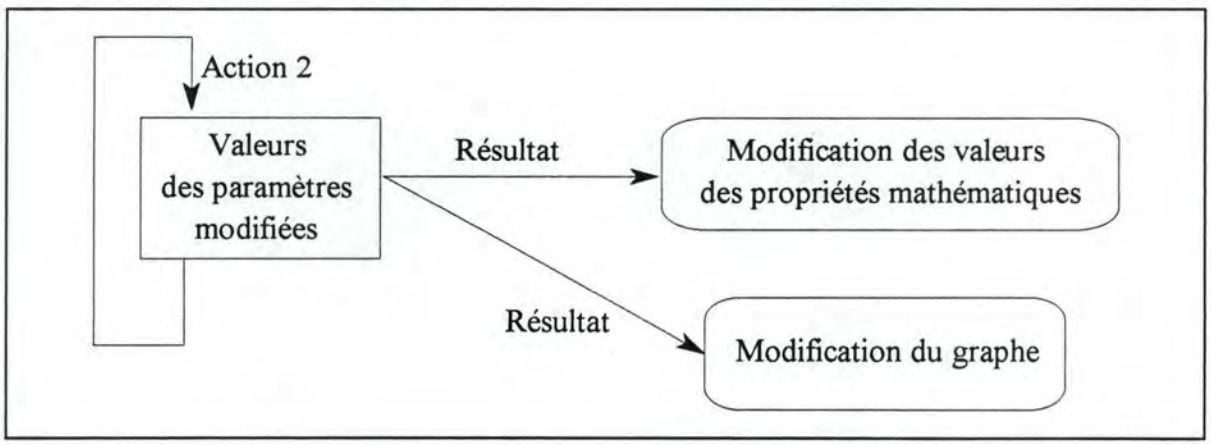


Figure 2.4 : Modification des valeurs des paramètres d'un exercice en cours

3. Analyse fonctionnelle

3. Analyse fonctionnelle

3.1 Introduction

L'étude des fonctionnalités du logiciel nous permettra de réfléchir à la manière de proposer une solution à notre problème de départ et de réaliser une architecture logique de cette solution. Pour que cette étude soit complète, nous aborderons le problème sous différents aspects : nous commencerons par la partie mathématique et graphique; ensuite, nous analyserons l'aspect interactif et handicap de l'élève, et nous terminerons par l'interface et les aides de notre logiciel.

3.2 Etude des fonctionnalités

3.2.1 Partie mathématique

Au niveau mathématique, nous avons fait pour chaque famille de fonctions leur étude et ce en tenant compte du contenu du cours de mathématiques des élèves.

3.2.1.1 Equations et propriétés

- La droite

Quant à la droite, nous l'avons défini sous sa forme générale :

◊ la droite quelconque ($y = ax + b$)

Les propriétés mathématiques qui lui sont attachées sont :

◊ le coefficient angulaire;
◊ l'angle avec l'axe des X;
◊ l'intersection avec l'axe des X et l'axe des Y.

- La parabole

La forme de représentation de l'équation est : $y = ax^2 + bx + c$

Pour la parabole, nous avons les propriétés suivantes :

◊ l'intersection avec l'axe des X et l'axe des Y;
◊ le sommet;
◊ l'axe de symétrie.

- Les fonctions trigonométriques

Les courbes trigonométriques seront représentées sous la forme :

k * sin (ax + b)
k * cos (ax + b)
k * tg (ax + b)

Pour les fonctions trigonométriques, nous analyserons les propriétés suivantes :

◊ la période;
◊ la parité;
◊ l'axe de symétrie;
◊ l'intersection avec l'axe des X et l'axe des Y;
◊ les sommets de la courbe.

Une étude complète et détaillée de toutes les familles de fonctions que nous avons envisagées se trouve dans l'annexe B.

3.2.2 Partie graphique

- La droite

  ◊ Bouger un point quelconque de la droite en gardant fixe le point d'intersection avec l'axe des X.
  ◊ Faire une translation de la droite.

- La parabole

  ◊ Déplacer le sommet de la parabole en gardant fixe les points d'intersection avec l'axe des X.
  ◊ Déplacer verticalement ou horizontalement la courbe.

  ◊ Afficher l'axe de symétrie.
  ◊ Afficher la tangente au sommet.
  ◊ Afficher la tangente en un point de la courbe.

- Les courbes trigonométriques

  ◊ Faire une translation de la courbe par rapport à l'axe des Y.

  ◊ Afficher les tangentes verticales de la fonction tangente.
  ◊ Afficher la tangente d'un sommet donné.

Dans l'annexe C se trouvent les dessins illustrant chacun des cas que nous venons de traiter.

3.2.3 Partie interaction

Le fait que le logiciel est destiné à des handicapés moteur implique non seulement la nécessité d'une interface adéquate mais influence la conception des fonctionnalités elles-mêmes. En effet, dans le cas d'un utilisateur normal, celui-ci peut effectuer un grand nombre de manipulations simples ou complexes (la frappe d'un texte par exemple) sans que cela lui demande un effort considérable, mais il n'en va plus de même dans notre cas. Nous devrons donc toujours garder cette idée en tête et concevoir le logiciel de manière telle que les manipulations effectuées par l'élève soient minimales sans pour autant diminuer ses possibilités de compréhension.

3.2.3.1 Saisie de l'énoncé

Pour la saisie de l'énoncé d'un exercice, l'élève doit sélectionner une des formes d'équations générales proposées dans la partie mathématique (voir le point 3.2.1).

Le fait que l'on ne permet pas à l'élève d'aller éditer lui-même son énoncé est dû à un de nos objectifs à savoir la minimisation des manipulations de l'élève.

3.2.3.2 Détermination des valeurs de paramètres

Une fois que de l'énoncé est affiché, il faut encore déterminer les valeurs des paramètres. Deux principes seront ici à la base du choix de la façon de procéder : minimiser le nombre de manipulations et permettre une visualisation dynamique de la relation entre l'expression algébrique et la représentation graphique.

Pour ce faire, nous envisageons la possibilité suivante : permettre à l'élève de sélectionner le paramètre à modifier. Une fois sélectionné, la machine modifiera elle-même les valeurs et l'élève n'aura plus qu'à confirmer la modification.

L'aspect dynamique de toutes ces actions réside dans le fait que pour chaque modification des valeurs des paramètres, l'utilisateur peut l'observer directement sur le graphique. En effet, si l'élève doit lui-même éditer son énoncé, nous devrons lui fournir un éditeur de texte spécial. Or, le fait d'aller éditer soi-même son énoncé d'exercice ou le sélectionner dans un menu ne change en rien la compréhension de l'élève.

3.2.3.3 Calcul des propriétés mathématiques

Une étude de fonctions s'accompagne généralement d'une série de calculs de propriétés mathématiques telles que l'intersection avec les axes. Nous avons vu dans la partie mathématique (voir point 3.2.1) que pour une famille de fonctions donnée, nous avons la possibilité de faire plusieurs calculs. En temps normal, l'élève fait ses calculs et trace son dessin graphique en fonction des résultats de ces derniers. Dans le cas qui nous occupe, nous voyons trois façons de procéder :

1. La machine a la charge de tous les calculs;
2. L'élève indique à la machine les différents calculs à effectuer.

La première solution permet de réduire le nombre de démarches à faire par l'élève mais peut réduire également son niveau de compréhension. La deuxième solution semble être la meilleure car l'élève demandera à la machine d'exécuter les calculs qui l'intéressent. De plus, elle permet de réduire les manipulations de l'élève.

Pour montrer le lien dynamique existant entre les valeurs des paramètres d'une équation et ses propriétés mathématiques, nous suggérons que la machine prenne en charge les différents calculs et affiche les résultats à l'écran. A chaque modification de valeurs des paramètres, elle les recalcule et les affiche.

3.2.3.4 Modification du dessin sur l'écran graphique

Nous avons dit qu'une des façons de comprendre le lien existant entre un graphe et sa fonction algébrique est de les manipuler directement. Nous venons de voir la modification de la fonction par ses paramètres (voir point 3.2.3.2). Maintenant, comment envisager les modifications sur graphe ?

Pour une personne handicapée, une des façons de se déplacer sur l'écran est le balayage automatique, c'est-à-dire que tous les éléments d'une fenêtre sont rendus actifs par machine de telle sorte que l'utilisateur n'aura qu'à cliquer sur un bouton pour confirmer sa sélection.

Dans le problème qui nous occupe, nous voyons deux types de modification sur les graphes, une modification ponctuelle (déplacement d'un point de la droite par exemple) et une modification de l'ensemble de la courbe (le déplacement d'une parabole vers le haut par exemple).

Pour la modification ponctuelle et globale, nous voyons deux solutions :

1. Définir une liste d'objets de dessin à sélectionner.
2. Pour un objet de dessin sélectionné, fournir un système de déplacement par directions

La première solution nous permet d'optimiser le déplacement et la deuxième nous semble être plus naturelle. Evidemment la vitesse de ces balayages sera modifiable en fonction du choix de l'utilisateur lors de la configuration de son interface.

Concrètement, cela devrait se dérouler de la manière suivante : lorsque l'utilisateur veut poser son curseur à un endroit précis, un système de balayage se met en marche et balaye les différents objets de dessin à l'écran. Pour un objet de dessin sélectionné, un système de direction se met en marche permettant ainsi à l'utilisateur de sélectionner le sens de déplacement de son curseur (gauche, droite, bas, haut). Enfin un système de validation permettra de clôturer ou d'annuler la modification en cours.

3.2.4 Partie handicap

3.2.4.1 Le balayage automatique

Comme nous venons de préciser plus haut (voir point 3.2.3.4), le balayage automatique des éléments d'une fenêtre est une manière de procéder pour se déplacer à l'écran.

Vu que ce sont les éléments d'une fenêtre qui seront balayés, il serait intéressant de trouver un moyen d'optimiser le balayage pour chaque élève en particulier.

Un autre point important est d'essayer de rendre possible l'adaptation du balayage en fonction de la facilité d'un élève à appuyer sur un bouton. Ainsi, plus un élève a de facilité à appuyer sur un bouton, plus il est intéressant de diminuer le temps d'accès à un élément particulier.

Donc, deux points à retenir lors de l'implémentation du balayage :

1) La possibilité d'optimiser le balayage c'est-à-dire chercher la façon la plus efficace pour balayer une fenêtre dont un exemple serait le balayage des éléments par groupe, par ligne ou par colonne.

2) La possibilité d'adapter la vitesse de balayage lors de la configuration de l'interface de l'élève.

3.2.4.2 Choix dans les menus

Dans la mesure du possible, nous chercherons à ce que les balayages se fassent sur les représentations visuelles des éléments à choisir et non sur une palette de balayage spéciale qui se superposerait à la vue de l'écran. Par exemple, dans le cas du choix d'un paramètre dans la fenêtre des paramètres, on peut imaginer que ce sont les lignes définissant les paramètres elles-mêmes qui sont alternativement mises en évidence. Bien sûr, cela ne sera pas toujours possible car il ne faut pas encombrer l'écran de choses qui ne sont pas suffisamment utiles.

3.2.4.3 Personnalisation de l'interface

Une de nos lignes directrices est de permettre une configuration de l'interface aussi souple que possible.

Donc, avant de commencer tout exercice, nous envisageons de laisser à l'enseignant la possibilité de configurer l'interface de l'élève. Pour ce faire, une fenêtre présentera les différentes possibilités concernant le balayage des écrans. L'enseignant aura les possibilités suivantes :

1. Choisir la vitesse de balayage;
2. Sauvegarder les options pour chaque élève.

D'une part, nous pourrions que l'enseignant choisisse pour l'élève la même vitesse de balayage pour toutes les fenêtres, d'autre part il serait intéressant que l'utilisateur puisse choisir pour chaque type de fenêtre une vitesse de balayage différente. En d'autres termes, nous pourrions imaginer que l'utilisateur a besoin d'un balayage plus lent pour taper ses commentaires et un balayage plus rapide pour sélectionner un point ou une courbe dans la fenêtre des graphes.

Dans ce cas, dans la fenêtre des options de l'interface, nous devrons laisser à l'utilisateur le choix de plusieurs vitesses de balayage pour les différentes fenêtres en question.

Cependant, l'utilisateur peut ne pas vouloir configurer lui-même son interface et demander au système de le faire. Par défaut, la machine affectera une certaine vitesse de balayage pour les différents types de fenêtres.

Les différents types de fenêtres seraient :

- Une fenêtre
- Une boîte de dialogue
- Une boîte de message

3.2.5 Partie interface

3.2.5.1 Le but recherché

Etant donné le caractère didactique de notre logiciel, nous voulons par l'intermédiaire de notre interface, aider les élèves à acquérir une certaine adaptation à notre logiciel. Cette adaptation dépend de l'habitude de l'élève à utiliser un logiciel.

En effet, si nous habituons les élèves à une certaine présentation de nos fenêtres, ils s'adapteront plus vite au logiciel, ce qui ne sera pas le cas si nous leur présentons des fenêtres toutes différentes les unes des autres et sans aucune structure de cohérence. Par exemple, tantôt le bouton « OK » se trouve en bas à gauche de la fenêtre et tantôt en haut à droite, ou encore tantôt le bouton de validation par défaut est le bouton « OK » et tantôt le bouton « Annuler ».

Afin d'éviter cela, nos fenêtres seront créées selon une certaine structure de présentation que nous essayerons de respecter dans la mesure du possible.

3.2.5.2 Type de fenêtre standard

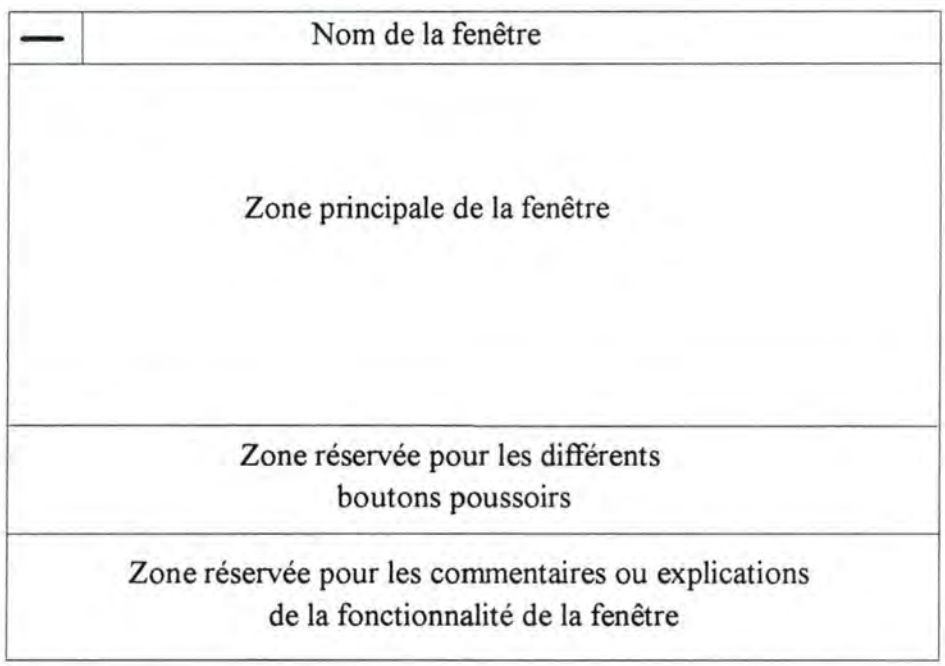


Figure 3.1 : Type de fenêtre standard

3.2.5.3 Explication de sa structure

La plupart de nos fenêtres auront les caractéristiques suivantes : elles porteront un nom et peuvent être fermées à tout instant. Elles seront réparties en trois zones bien définies. Juste après le titre de la fenêtre sera sa zone principale, elle présentera les différentes actions que l'utilisateur peut effectuer comme sélectionner un élément dans une boîte liste. La zone en dessous sera réservée aux boutons poussoirs tels que le bouton « OK », le bouton d'« Annulation » ou le bouton d'« Aide ». Et la dernière zone sera destinée à donner quelques commentaires ou explications aidant l'utilisateur dans son action. Afin de faciliter davantage le travail de l'utilisateur, un bouton poussoir par défaut sera défini, le bouton de validation « OK » par exemple. De cette façon, l'utilisateur peut acquérir une certaine habitude après un certain temps d'utilisation du logiciel.

3.2.5.4 Les menus

Nous devons envisager deux types de présentation des menus. L'un pour l'enseignant et l'autre pour l'élève.

Pour l'enseignant, nous avons les possibilités suivantes :

- la barre de menus
- les menus déroulants
- les boîtes de dialogue; etc.

Pour l'élève, une palette d'actions remplacera le menu principal. Sur cette palette d'actions seront fournies les actions de bases lui permettant de commencer un exercice, de continuer un exercice en cours, de sauver son travail sur disque et de quitter le programme. En plus de l'enseignant, deux actions supplémentaires devront être ajoutées :

- l'accès à la fenêtre des paramètres en vue de modifications;
- l'accès à la fenêtre des graphes en vue de modifications.

3.2.5.4.1 Comment choisir dans les menus ?

Pour l'enseignant : par clavier et par souris.
Pour l'élève : par balayage automatique.

Concrètement, le balayage passe en revue avec une certaine vitesse tous les éléments d'un menu. Quand l'élève veut sélectionner un élément, il attend que celui-ci soit mis en évidence et au moyen d'un bouton (un contact) le sélectionne. Dans chaque menu, nous devrons également penser à fournir une touche lui permettant de revenir en arrière.

Nous devrons donc fournir à l'élève :

- une touche d'action
- une touche de retour en arrière

La touche d'action peut soit déclencher un autre balayage, soit exécuter une commande.

3.2.6 Partie environnement

3.2.6.1 Chargement d'un fichier

Dans le cas de chargement de fichiers, nous devons aussi envisager deux présentations différentes à l'écran. Une pour l'enseignant et une autre pour l'élève.
Pour l'enseignant, le système classique de chargement de fichier devra être mis en place, c'est-à-dire qu'avec la souris, l'utilisateur pourra aller sélectionner dans le menu l'option de chargement afin de sélectionner le fichier de son choix. Quant à l'élève, les mêmes options lui seront proposées à la différence près que tout se passera à l'aide d'un balayage.
Rappelons que dans un système de balayage, tous les éléments à balayer doivent être affichés.

3.2.6.2 Sauvetage d'un fichier

Pour sauver un fichier en cours, nous avons plusieurs possibilités. En effet, pour un fichier donné, nous avons deux cas : soit le fichier porte déjà un nom soit ce fichier n'a pas encore de nom. Dans le cas où le fichier a déjà un nom, nous pouvons le sauver de deux façons, c'est-à-dire soit proposer à l'utilisateur de le sauver sous le même nom soit le sauver sous un autre nom. Dans le cas où le fichier n'a pas encore de nom, nous devons proposer à l'utilisateur de le sauver sous un nouveau nom.

Fichier → avec nom	→	sauver sous le même nom
	→	sauver sous un autre nom
→ sans nom	→	sauver sous un nouveau nom

Maintenant que nous avons vu les différentes possibilités de sauvetage d'un fichier, nous devons encore voir comment le faire concrètement c'est-à-dire comment tout cela va se présenter à l'écran.

Rappelons que dans notre cas, l'enseignant et l'élève devraient pouvoir faire cette opération de sauvegarde. Pour le professeur, aucun problème ne se pose car lorsqu'il veut sauver un fichier, il lui suffit, avec la souris par exemple, d'aller dans la barre de menus et sélectionner la commande « sauver » ou « sauver sous ».

Tout ce mécanisme doit être revu pour l'élève. Rappelons que nous voulons minimiser les manipulations de l'élève, alors une solution à ce problème serait d'envisager des noms de fichiers par défaut, c'est-à-dire que l'élève n'aura pas à éditer le nom de son fichier vu que la machine attribuera de manière automatique un nom à tous ces fichiers. Donc, quelque part, nous devrons gérer la liste des noms de fichiers par défaut. De cette façon, l'élève n'aura qu'à valider son choix. Dans ce cas, nous n'aurons pas à nous poser la question comment gérer la saisie de données et la fin de cette saisie.

Nous proposons la solution suivante : à chaque exercice correspond un fichier (par exemple le premier exercice sera sauvé dans le fichier portant le nom *exercice1*, le deuxième *exercice2*, etc.). Lorsque pour un exercice donné, l'élève veut apporter des modifications (par exemple en changeant les valeurs de paramètres, donc l'exercice sera modifié au niveau calcul et dessin graphique. Nous devrons proposer à l'élève de sauvegarder ses modifications.

Dans le cas où celui-ci accepte de garder trace de ses modifications, nous devons sauver l'exercice sous un nom par défaut (par exemple, en effectuant son premier exercice, l'élève verra son exercice sauvé sous le nom «*exercice1*» et lorsqu'il sauve ses modifications, celles-ci seront sauvées sous le nom par défaut «*exercice1a*», etc.). Ainsi, la sauvegarde sera prise en charge par la machine et non l'utilisateur (dans le cas de l'élève).
Pour ces différentes opérations, il serait judicieux et intéressant de donner une aide à l'utilisateur.

3.2.7 Partie Aide

3.2.7.1 Aide sur les actions

Ce que nous voulons c'est proposer une aide sur chaque action effectuée par l'élève de telle sorte que ce dernier sera guidé tout au long de son travail. Par exemple, lors de l'affichage d'une boîte de dialogue, donner un commentaire ou une explication.

3.2.7.2 Aide « on line »

Il serait intéressant de proposer une aide sur :

1. La présentation du logiciel;
2. L'explication de la mise en route du logiciel;
3. L'explication des fonctionnalités du logiciel;
   (comment choisir dans les menus par exemple);
4. L'explication des termes mathématiques
   (donner la définition du coefficient angulaire et donner son interprétation graphique par exemple).

4. Analyse conceptuelle

4. Analyse conceptuelle

4.1 Introduction

Après l'analyse fonctionnelle de notre projet, nous allons maintenant passer à son analyse conceptuelle. De cette analyse conceptuelle sera élaborée une solution fonctionnelle détaillée mais encore indépendante de tout moyen de réalisation, c'est-à-dire sans se préoccuper encore du langage de programmation.

4.2 Modèle « Structuration des traitements »

4.2.1 Explication du modèle

Nous allons, pour faire l'analyse conceptuelle, nous baser sur le modèle de «Structuration des traitements»[9]. Le modèle de structuration des traitements permet de représenter les fonctionnalités à différents niveaux de détails. En d'autres termes, ce modèle nous permettra de structurer notre projet par raffinements successifs. Cette façon de travailler nous permet d'avoir une décomposition par partition de notre projet. En effet, celui-ci peut être vu comme une décomposition en un certain nombre d'unités. Chaque unité se compose d'une série de fonctions. Et chaque fonction est constituée d'un certain nombre de traitements. De plus, pour relier les différentes unités du projet, nous ajouterons à ce modèle la relation « utilise ».

Cette relation « utilise » s'explique par l'exemple suivant : Deux unités A et B peuvent avoir besoin d'une même fonction Z. Nous pouvons voir cela de deux façons : soit nous décidons de définir la fonction Z comme étant une fonction de l'unité B et faire de A un descendant de B, de cette façon A hérite de B et peut donc utiliser toutes ses fonctions; soit nous décidons que les deux unités A et B sont indépendantes et définissons la relation « utilise » entre elles. Nous pouvons ainsi avoir des relations entre les unités. Ceci est également valable au niveau des fonctions.

Ainsi, notre projet sera décomposé en une série d'unités dont un exemple serait la gestion des paramètres. Cette unité sera à son tour découpée en un certain nombre de fonctions dont un exemple serait la fonction d'affichage des paramètres et la fonction de sélection d'un paramètre. Une fonction est constituée de traitements dont un traitement serait la lecture de la liste des paramètres ou l'écriture des valeurs de paramètres dans une fenêtre.

[9] Conception Assistée des Systèmes d'Information, Méthode-Modèles-Outils, F. Bodart et Y. Pigneur, 2°édition, MASSON 1989.

4.2.2 La raison de notre choix

Le fait que nous ayons choisi le modèle de « Structuration des traitements » pour faire notre analyse conceptuelle s'explique par la raison suivante :

Nous avons choisi la programmation Orientée Objet pour l'implémentation de notre logiciel. Par les concepts que la programmation Orientée Objet met en avant tels que le concept d'objet, d'encapsulation, d'héritage, et ainsi de suite, elle permet une représentation arborescente des choses. Par exemple, un objet A est descendant d'un objet B et par conséquent hérite de toutes les méthodes de B. C'est ce qu'offre le modèle de « Structuration des traitements ». En effet, une unité X peut reprendre une ou plusieurs fonctions d'une unité Y, ce qui peut être illustré par la relation d'héritage ou la relation « utilise ». C'est pourquoi, nous pensons que ce modèle nous permettra de représenter l'architecture de notre projet.

4.2.3 Représentation du modèle

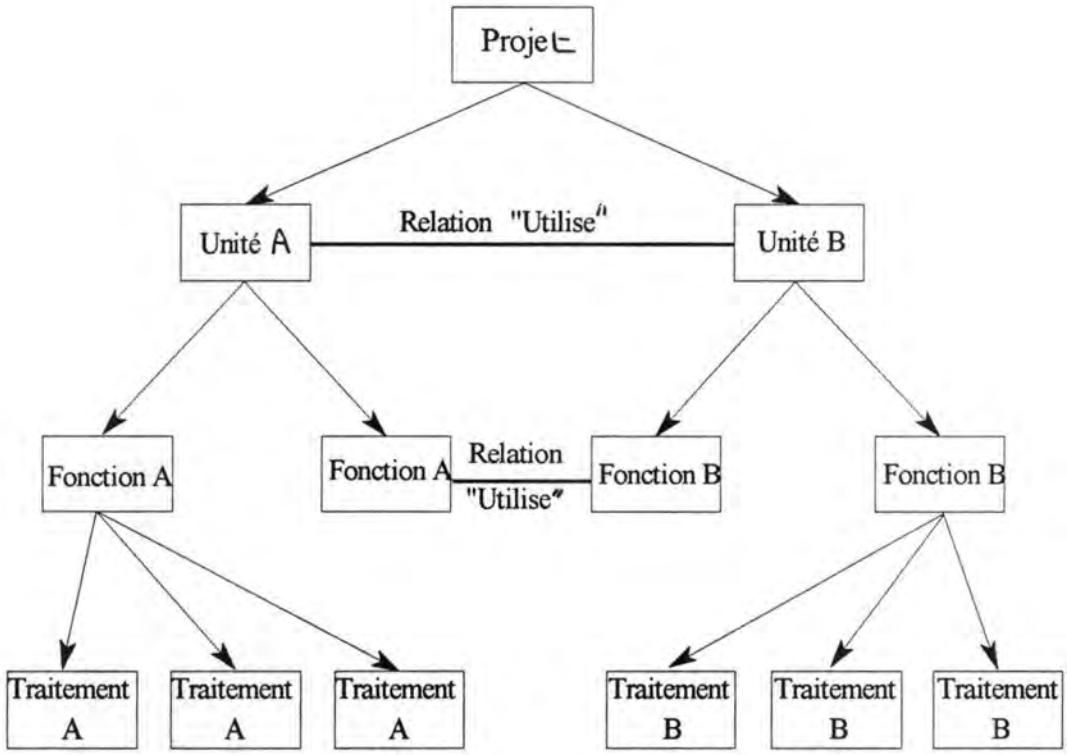


Figure 4.1 : Modèle de « Structuration des traitements » avec la relation « utilise »

4.3 Architecture logique

4.3.1 Les unités du projet

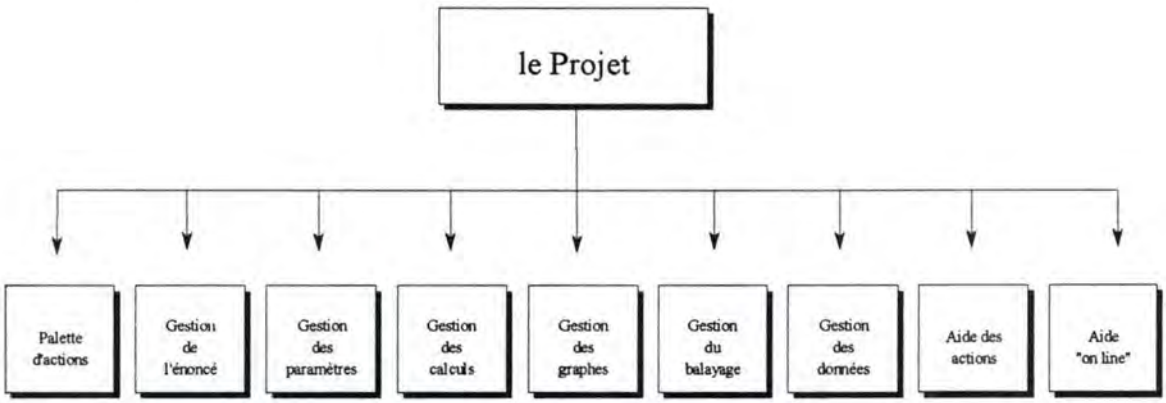


Figure 4.2 : Les unités du projet

4.3.1.1 La palette d'actions

La palette d'actions jouera le rôle du menu principal de l'élève. Elle devra lui permettre de commencer un nouvel exercice, continuer un exercice en cours, de le sauver sur disque et de quitter le programme. En plus, elle devra permettre l'accès à la fenêtre des paramètres et à la fenêtre des graphes en vue d'actions de sélections et de modifications.

4.3.1.2 La gestion de l'énoncé

Un énoncé est sous forme d'équation algébrique. Les différentes formes d'équations sont prédéfinies, les enseignants et les élèves n'ont plus qu'à sélectionner une des formes d'équations proposées afin de commencer leur exercice. Cette gestion devra donc afficher la liste des énoncés et en permettre la sélection pour l'afficher dans la fenêtre des énoncés.

4.3.1.3 La gestion des paramètres

Cette partie a pour but de gérer l'affectation et la modification des valeurs des paramètres d'une équation algébrique. Par défaut, la machine affectera des valeurs aux paramètres et à l'aide de la souris ou du balayage, permettre à l'utilisateur d'apporter des modifications.

4.3.1.4 La gestion des propriétés mathématiques

Dans cette partie, la machine a la charge de tous les calculs. A l'affichage d'un énoncé, elle calculera les valeurs des propriétés mathématiques qui lui sont attachées et refaire les calculs lors de chaque modification des valeurs des paramètres.

4.3.1.5 La gestion des graphes

Cette partie aura la charge de permettre à l'utilisateur d'aller sélectionner un des objets de dessin dans la fenêtre des graphes et d'en modifier la position. Pour garder l'aspect dynamique du logiciel, elle doit aussi communiquer avec le gestionnaire des paramètres et le gestionnaire des énoncés, de même qu'avec le gestionnaire de calculs lors de chaque modification de dessin.

4.3.1.6 La gestion du balayage

Certains handicapés moteur ont du mal à utiliser la souris ou en sont incapables. Ce gestionnaire permet de la remplacer par un affichage dynamique des objets à sélectionner. Il essaie d'être le plus souple possible afin de s'adapter aux besoins de chaque élève. Elle s'occupera d'une part de la configuration des vitesses de balayage et d'autre part de sa sauvegarde sur disque.

4.3.1.7 La gestion des données

Cette partie s'occupe de la sauvegarde des paramètres, des dessins et des calculs lors de chaque modification ou d'affectation à la demande de l'utilisateur. Elle s'occupe également de charger ou de sauver un exercice effectué par l'utilisateur. En outre, elle aura la charge des transferts de données entre les différentes unités.

4.3.1.8 L'aide des actions

Cette partie aura pour objet de donner des explications lors de chaque action exécutée par l'utilisateur. Ce qui permettra une plus grande aisance d'utilisation du logiciel. Ces explications seront de nature « ponctuelles », c'est-à-dire qu'elles ne seront affichées à l'écran que lors de l'activation des commandes.

4.3.1.9 L'aide « on line »

L'aide a la charge d'informer l'utilisateur sur :

- La présentation du logiciel
- L'explication de la mise en route du logiciel
- L'explication des fonctionnalités du logiciel
- L'explication des termes mathématiques

4.3.2 Les fonctions du projet

Nous allons maintenant présenter et détailler l'ensemble des fonctions des différentes unités de notre projet.

4.3.2.1 L'unité: La palette d'actions

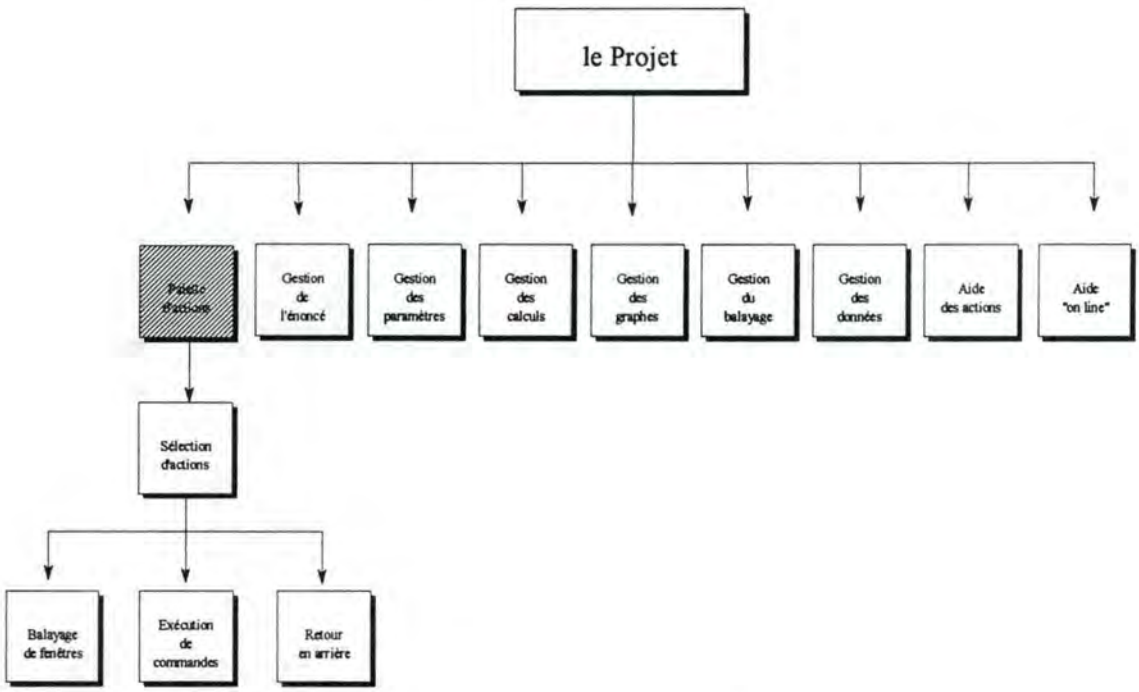


Figure 4.3 : Les fonctions constituant l'unité de la palette d'actions

4.3.2.1.1 La fonction: Sélection d'actions

Cette fonction devra permettre à l'élève d'accéder aux commandes de base du logiciel par le balayage des outils composant la palette d'actions. Le fait de sélectionner une actions peut entraîner trois autres actions, à savoir le balayage d'une autre fenêtre, l'exécution d'une commande bien précise et le retour en arrière, par exemple si l'action déclenche le balayage d'une boîte de dialogue, la touche de retour en arrière devra permettre de fermer la boîte de dialogue et de revenir à la fenêtre précédente.

4.3.2.2 L'unité: Gestion de l'énoncé

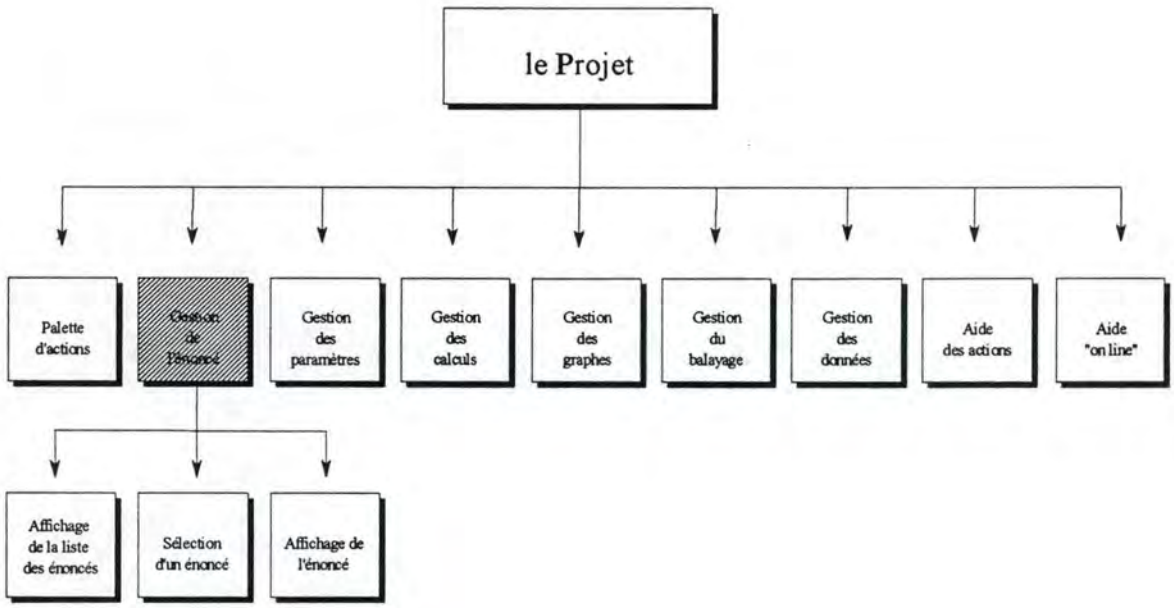


Figure 4.4 : Les fonctions constituant l'unité « Gestion de l'énoncé »

4.3.2.2.1 La fonction: Affichage de la liste des énoncés

Cette fonction présente à l'utilisateur une fenêtre dans laquelle sont affichés une série d'équations à sélectionner.

4.3.2.2.2 La fonction: Sélection d'un énoncé

Cette fonction permet à l'utilisateur de sélectionner une des formes d'équation.

4.3.2.2.3 La fonction: Affichage de l'énoncé

Après avoir choisi une équation, cette fonction l'affiche dans une fenêtre.

4.3.2.3 L'unité: Gestion des paramètres

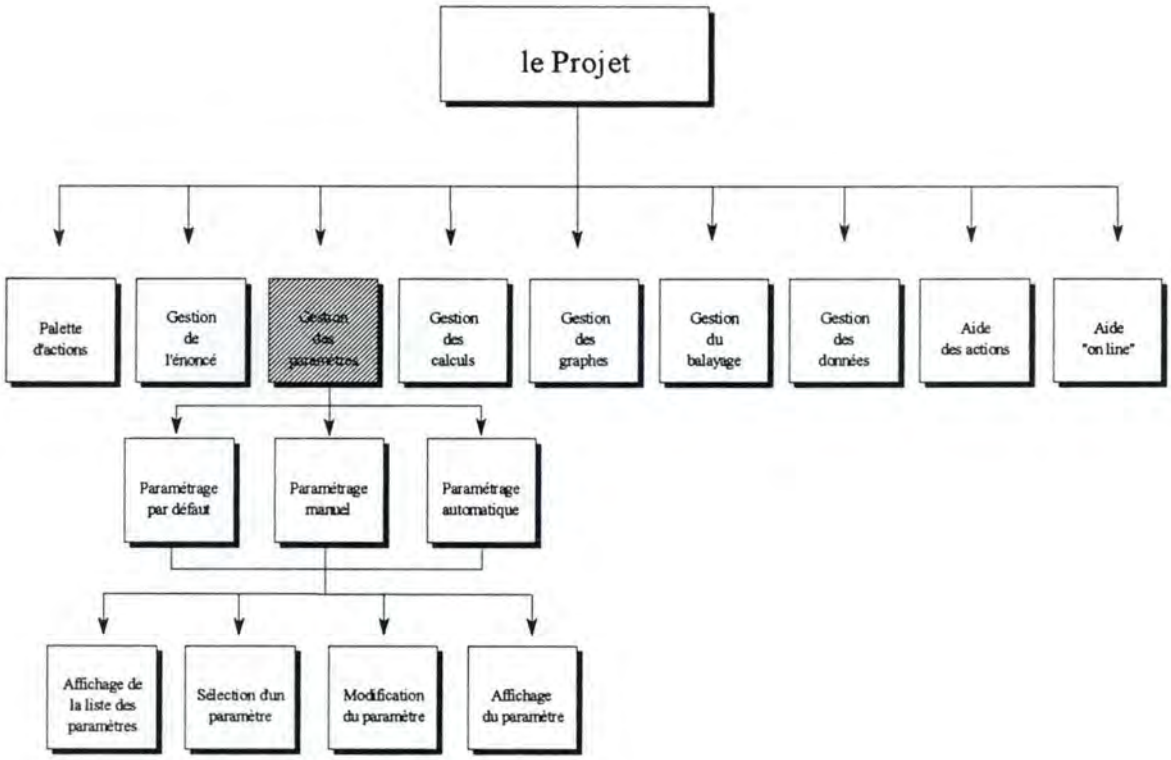


Figure 4.5 : Les fonctions constituant l'unité « Gestion des paramètres »

4.3.2.3.1 La fonction: Paramétrage par défaut

La machine affecte par défaut des valeurs aux paramètres. Ces valeurs seront différentes en fonction du type de fonction à analyser.

4.3.2.3.2 La fonction: Paramétrage manuel

Cette fonction donne l'accès à l'enseignant à la fenêtre des paramètres.

4.3.2.3.3 La fonction: Paramétrage automatique

Cette fonction devra balayer les paramètres de l'équation en vue de modifications par l'élève.

Pour chaque type de paramétrage, ces fonctions devront également pourvoir afficher la liste des paramètres, en permettre la sélection, la modification et l'affichage d'un paramètre donné.

4.3.2.4 L'unité: Gestion des propriétés mathématiques

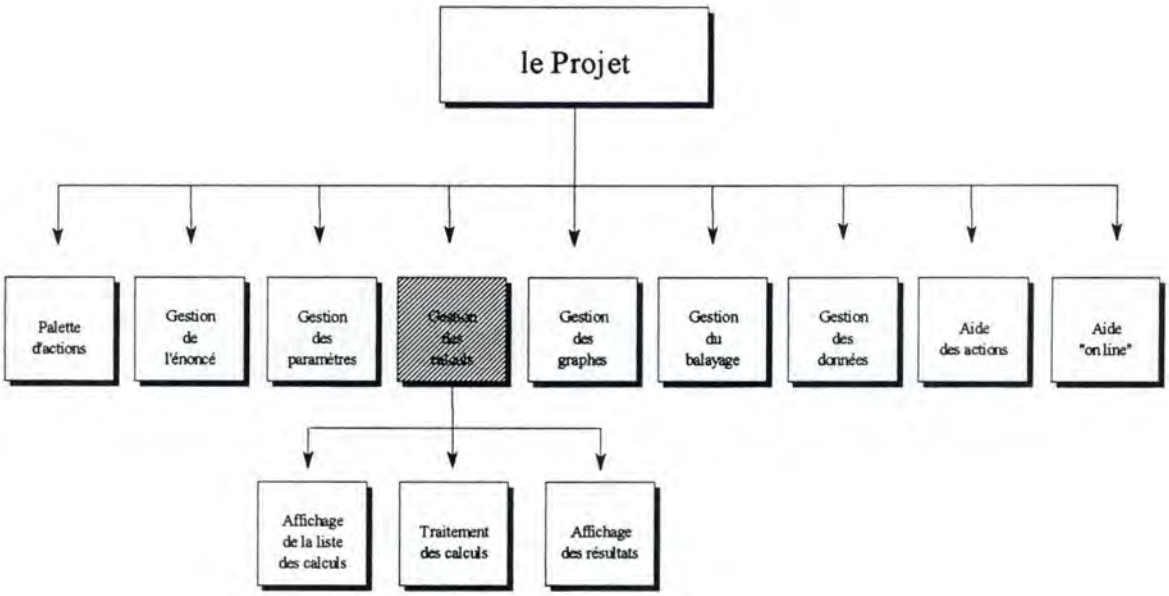


Figure 4.6 : Les fonctions constituant l'unité « Gestion des propriétés mathématiques »

4.3.2.4.1 La fonction: Affichage de la liste des calculs

Cette fonction a pour but d'afficher dans une fenêtre la liste des différentes propriétés mathématiques de l'équation en cours.

4.3.2.4.2 La fonction: Traitement des calculs

En fonction des modifications des valeurs des paramètres, recalculer les propriétés mathématiques de l'équation algébrique en cours.

4.3.2.4.3 La fonction: Affichage des résultats

Cette fonction affiche le résultat des calculs dans sa fenêtre.

4.3.2.5 L'unité: Gestion des graphes

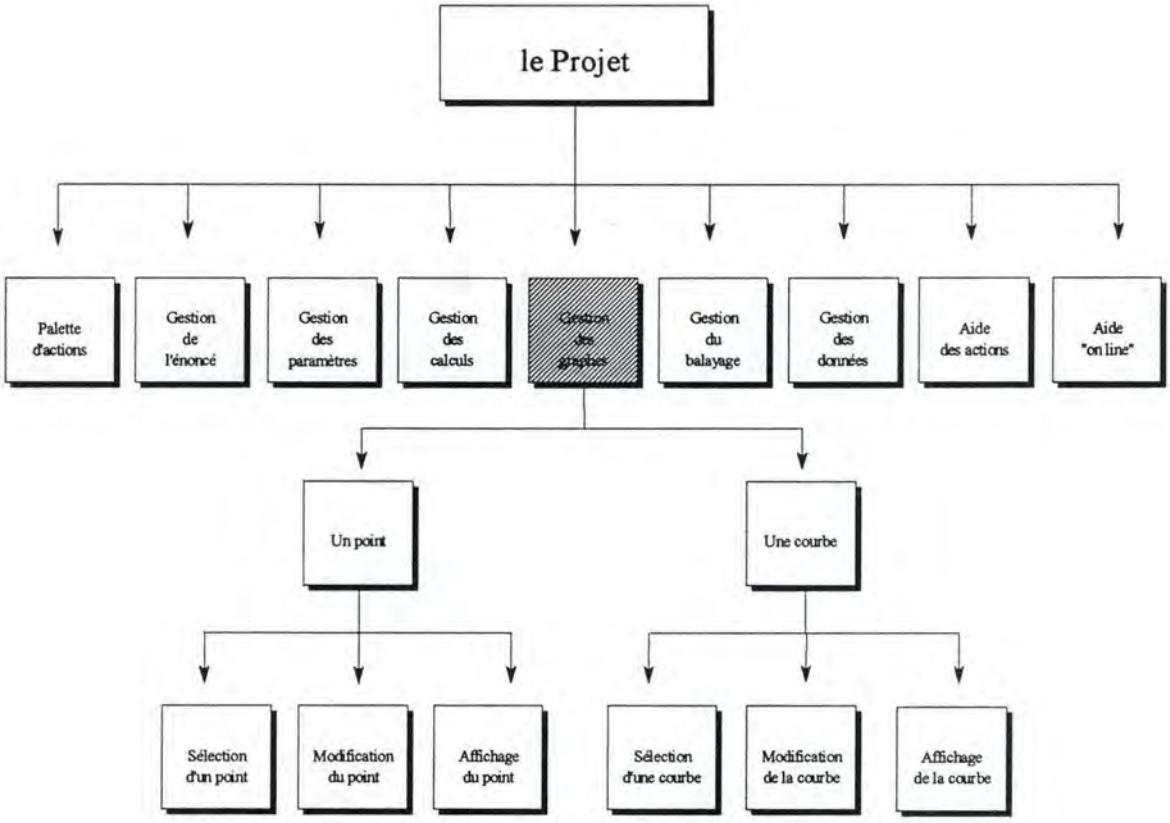


Figure 4.7 : Les fonctions constituant l'unité « Gestion des graphes »

L'ensemble de ces fonctions gère les différents objets dessins créés. Il s'agit de les mémoriser après leur création. Un objet dessin, après sa création, peut se trouver dans deux états : normal ou sélectionné. Ces fonctions s'occupent également des objets dessins qui se trouvent dans l'état « sélectionné », elles en permettent leur déplacement. Elles devront par conséquent créer et gérer une liste de points et une liste de courbes.

4.3.2.5.1 La fonction: Sélection d'une courbe

Cette fonction se charge de la sélection d'une des courbes de dessins qui sont activés par machine.

4.3.2.5.2 La fonction: Modification de la courbe

Cette fonction se charge de gérer la modification d'une courbe, c'est-à-dire effacer la courbe sélectionnée et mémoriser les nouvelles coordonnées de la courbe modifiée.

4.3.2.5.3 La fonction: Affichage de la courbe

Cette fonction se charge de redessiner la courbe modifiée avec ses nouvelles coordonnées.

4.3.2.5.4 La fonction: Sélection d'un point

Cette fonction s'occupe aussi de la sélection mais d'un point dans ce cas-ci.

4.3.2.5.5 La fonction: Modification du point

Fonction semblable à celle précédente mais au niveau d'un point.

4.3.2.5.6 La fonction: Affichage du point

Cette fonction se charge de redessiner le point modifié avec ses nouvelles coordonnées.

4.3.2.6 L'unité: Gestion du balayage

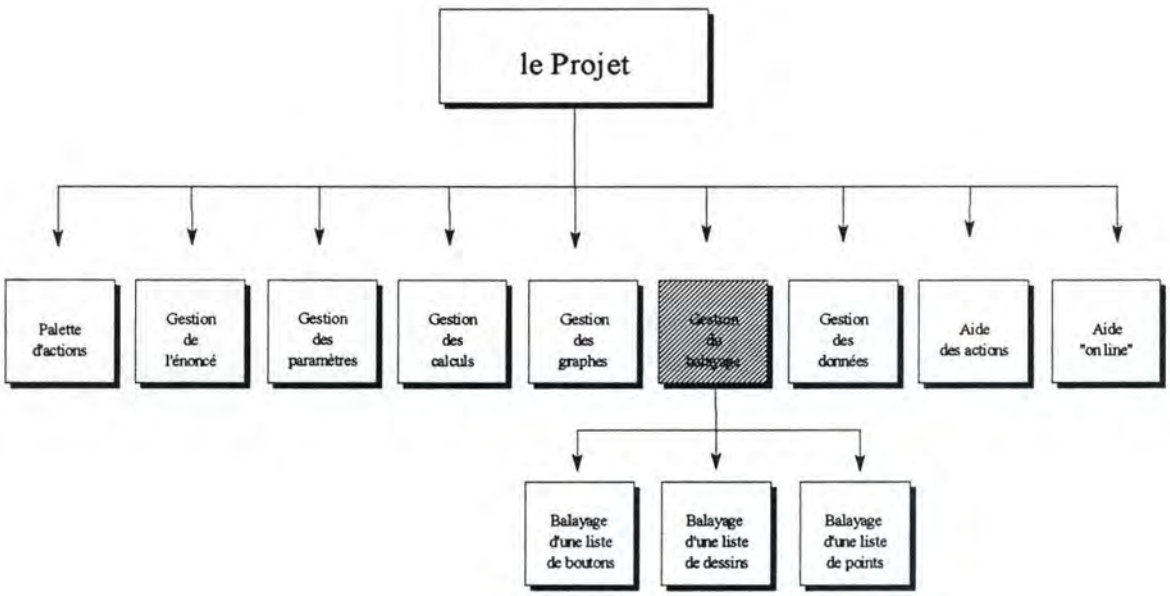


Figure 4.8 : Les fonctions constituant l'unité « Gestion du balayage »

4.3.2.6.1 La fonction: Balayage d'une liste de boutons

Cette fonction devra balayer les différentes fenêtres destinées à l'élève, à savoir la palette d'actions, les fenêtres (notamment celle des paramètres et celle des graphes), les boîtes de dialogue et les boîtes de messages.

4.3.2.6.2 La fonction: Balayage d'une liste de dessins

Cette fonction s'occupera du balayage de la liste des dessins graphiques afin que l'élève puisse les sélectionner en vue de modifications.

4.3.2.6.3 La fonction: Balayage d'une liste de points

Cette fonction est semblable à celle précédente mais au niveau de la liste des points.

4.3.2.7 L'unité: Gestion des données

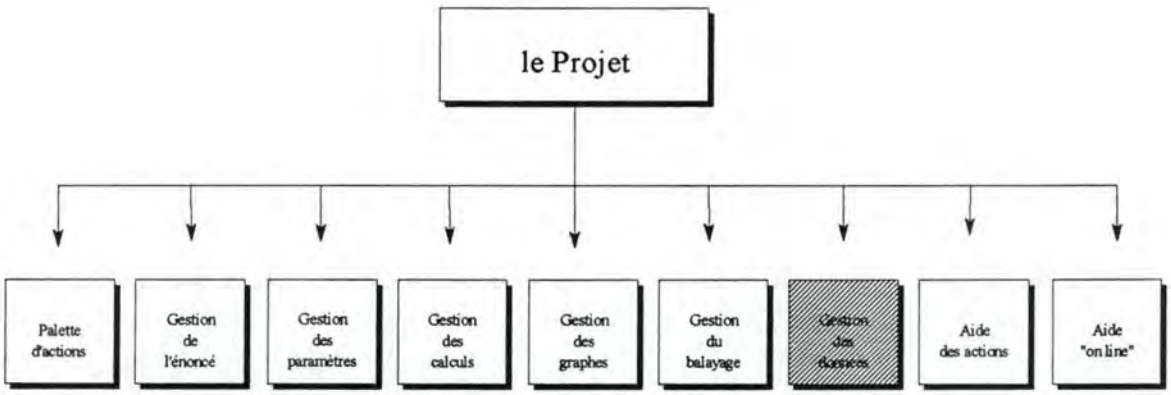


Figure 4.9 : Les fonctions constituant l'unité « Gestion des données »

La fonction principale de cette unité est le stockage de données en vue de transfert, de distribution et d'échanges d'informations entre les différentes unités.

4.3.2.8 L'unité: Aide des actions

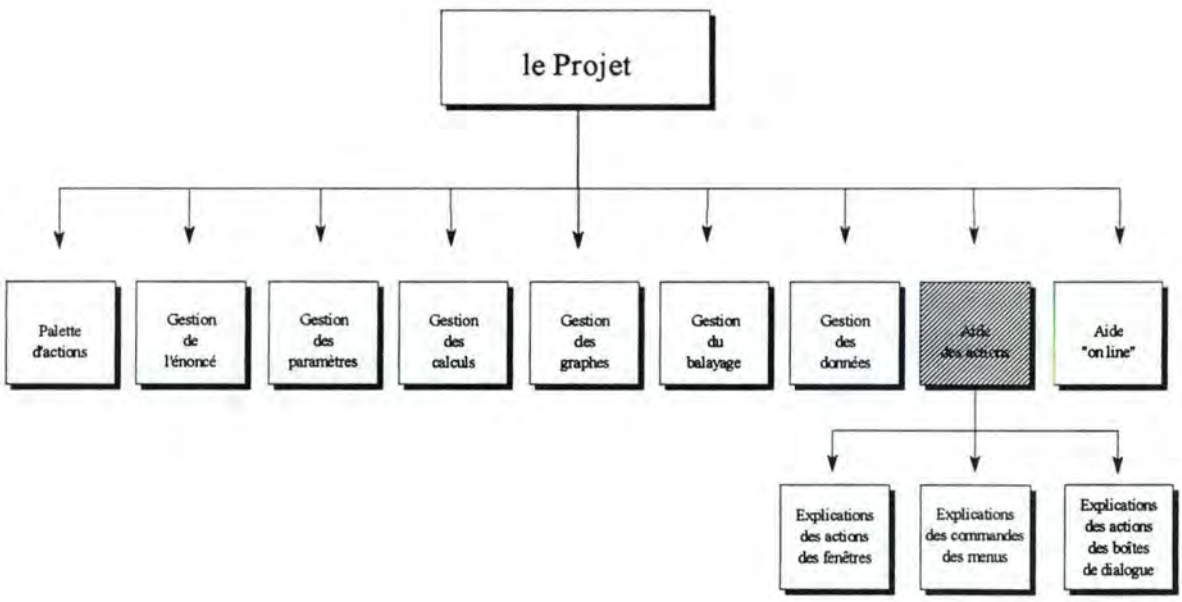


Figure 4.10 : Les fonctions constituant l'unité « Aide des actions »

Les trois fonctions ont la charge d'afficher les explications à différents niveaux : explications des commandes de la barre de menus, des actions des fenêtres et celles des boîtes de dialogue.

4.3.2.9 L'unité: Aide « on line »

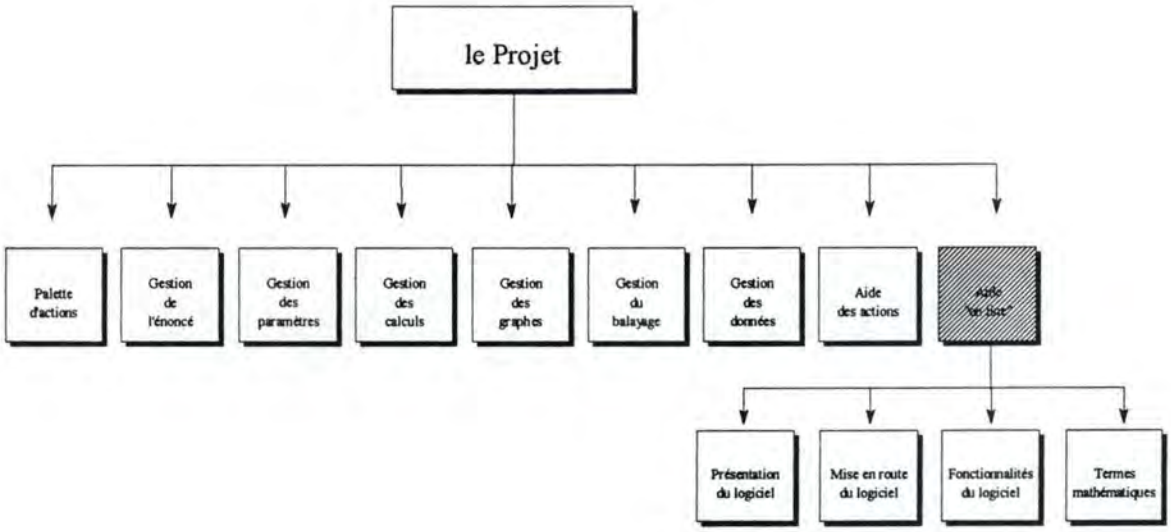


Figure 4.11 : Les fonctions constituant l'unité « Aide on line »

4.3.2.9.1 La fonction: Présentation du logiciel

Cette fonction affiche à l'écran une brève présentation du logiciel, notamment en expliquant rapidement son objectif.

4.3.2.9.2 La fonction: Mise en route du logiciel

Cette fonction affiche le mode d'emploi du logiciel.

4.3.2.9.3 La fonction: Fonctionnalités du logiciel

Cette fonction affiche à l'écran tout ce que le logiciel permet de faire lors d'une étude de fonctions mathématiques.

4.3.2.9.4 La fonction: Termes mathématiques

Cette fonction affiche à l'écran les différents termes mathématiques utilisés dans le logiciel ainsi que leur définition.

4.3.3 Liens entre les unités

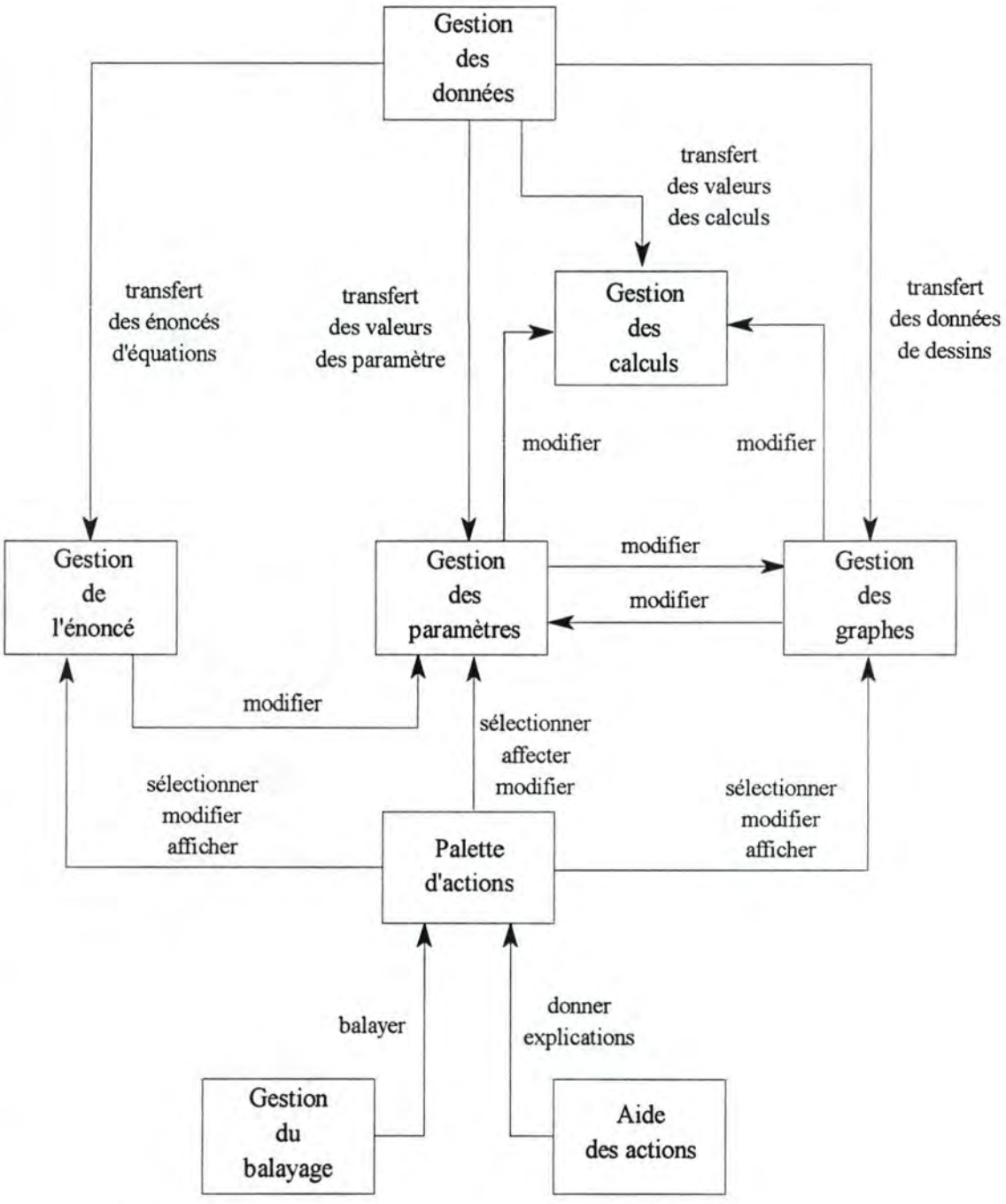


Figure 4.12 : Graphe des liens existant entre les unités de l'application

4.3.4 Les traitements du logiciel

Les traitements du logiciel seront définis et détaillés dans la partie implémentation. Ces traitements seront définis en termes de procédures ou méthodes. Pour cela, nous définirons tout d'abord les différents objets des unités du logiciel et à partir de là, nous en détaillerons leurs différentes méthodes.

5. Implémentation

5. Implémentation

5.1 Introduction

5.2 Les traitements du projet

5.2.1 Unité : Gestion des énoncés

5.2.2 Unité : Gestion des paramètres

5.2.3 Unité : Gestion des calculs mathématiques

5.2.4 Unité : Gestion des dessins graphiques

5.2.5 Unité : Gestion des balayages

5.2.6 Unité : Gestion des données

5.2.7 Unité : Palette d'actions

5.2.8 Unité : Gestion des aides des actions

5.2.9 Unité : Gestion des aides « on line »

5.1 Introduction

La partie implémentation de notre projet concerne les traitements des différentes unités de notre projet.

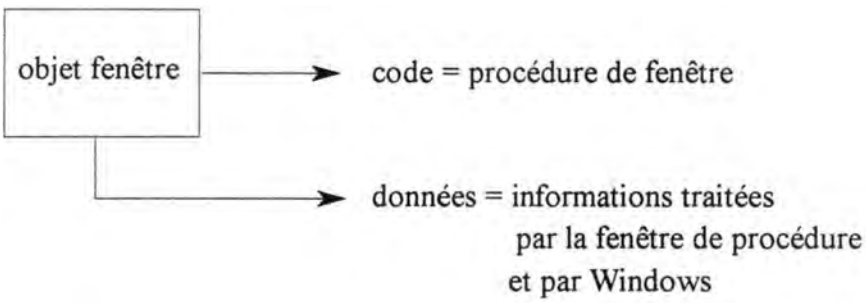
Cette partie a pour objectif de définir et de décrire les différents traitements nécessaires aux fonctions de notre projet. Pour cela, nous devrons avant tout définir les objets de notre logiciel, une fois les objets définis, nous décrirons ses différents champs et méthodes. Cette partie abordera donc tout l'approche orientée objet de notre logiciel.

5.1.1 Objet fenêtre et procédure de fenêtre

Une fenêtre créée dans le programme (l'objet fenêtre visuelle à l'écran) est associée à une procédure de fenêtre.

Une fenêtre est créée à partir d'un modèle appelé « classe de fenêtre » (exemple : TWindow) qui identifie la procédure de fenêtre. Et cette procédure de fenêtre traite les messages de l'objet fenêtre visuelle à l'écran.

L'objet fenêtre visuelle à l'écran est constitué de code et de données : l'objet est la fenêtre que l'on voit affichée à l'écran, le code est la procédure de fenêtre attachant à l'objet fenêtre et les données sont les informations traitées par la procédure de fenêtre et les informations que gère Windows.



5.1.2 Convention de notation

Chaque procédure de fenêtre correspondant à un objet donné sera notée de la manière suivante :

P + nom de l'objet	variable pointeur pointant vers un objet d'un certain type
T + nom de l'objet	le type de l'objet en question
nom de l'objet	soit le mot en entier si ce dernier est court soit une abréviation ou un préfixe si le mot est long ou composé

Exemple :

PFenParamètres = TFenParamètres
signifie que la variable pointeur « PFenParamètres » pointe vers un objet de type TFenParamètres

TFenParamètres = OBJECT (TWindow)
signifie que l'objet fenêtre des paramètres « TFenParamètres » est de type TWindow (un objet de type fenêtre)

5.1.3 Notion de « handle »

Un « handle » est un entier (16 bits) identifiant un objet.

Exemple :
Hdlg = un handle identifiant un objet boîte de dialogue
HWnd = un handle identifiant un objet fenêtre
HDC = un handle identifiant un contexte de périphérique

5.2 Les traitements du projet

5.2.1 Unité : Gestion des énoncés

5.2.1.1 Objet : Fenêtre d'affichage des énoncés

But : Créer un objet fenêtre permettant d'afficher l'énoncé d'exercice.

PFenEnonEq = TFenEnonEq
TFenEnonEq = OBJECT (TWindow)

Méthodes : Constructeur Init
Destructeur Done
Initialisation de la fenêtre SetUpWindow

Méthode de création de la fenêtre des énoncés

Appeler la méthode de création Init héritée de TWindow (HWnd, Titre)
Modifier les attributs des coordonnées de la fenêtre (x, y, w, h)
Modifier les attributs du style de la fenêtre
Créer la boîte d'édition permettant d'afficher l'énoncé de l'exercice (VarPoint)

Méthode d'initialisation de la fenêtre SetUpWindow

Appeler la méthode SetUpWindow héritée de TWindow
Mettre à jour la fenêtre des énoncés

Méthode de destruction de la fenêtre des énoncés

Détruire la variable pointeur (VarPoint) pointant vers la boîte d'édition de la fenêtre
Appeler la méthode de destruction Done héritée de TWindow

5.2.1.2 Objet : Boîte de dialogue de sélection des énoncés

But : Permettre à l'utilisateur de sélectionner son équation en vue de réaliser son exercice.

PDiaFamFonc = TDiaFamFonc
TDiaFamFonc = OBJECT (TDialog)

Méthodes : Procédure d'initialisation de la boîte de dialogue
Procédure correspondant au bouton OK de la boîte de dialogue
Procédure correspondant au bouton CANCEL de la boîte de dialogue
Procédure d'initialisation des paramètres
Procédure correspondant à la boîte liste des droites
Procédure correspondant à la boîte liste des paraboles
Procédure correspondant à la boîte liste des courbes trigonométriques
Procédure de balayage des boîtes listes
Procédure de déclenchement d'horloge
Procédure de destruction de l'horloge

Remarque :

Pour les boîtes listes, il y a les méthodes :

D'initialisation de la boîte liste (insérer des items dans la liste)
De sélection d'item et de l'afficher dans une fenêtre

Méthode d'initialisation de la boîte de dialogue SetUpWindow

Appeler la méthode SetUpWindow héritée de TDialog
[Initialiser l'horloge]
Initialiser la boîte liste des droites (insérer des items dans la boîte liste qui lui est destinée)
Initialiser la boîte liste des paraboles droites (insérer des items dans la boîte liste qui lui est destinée)
Initialiser la boîte liste des courbes trigonométriques droites (insérer des items dans la boîte liste qui lui est destinée)
[Initialiser les champs de l'objet]

Méthode correspondant au bouton OK de la boîte de dialogue

Voir si un item est sélectionné dans une des boîtes listes
Si c'est le cas
Alors l'afficher dans la fenêtre des énoncés
Sinon fermer la boîte de dialogue sans rien faire
Appeler la méthode correspondant au bouton OK héritée de TDialog

Remarque : Ceci dans le cas sans balayage. Dans le cas où le balayage est activé, le fait que l'élève clique sur le bouton OK stoppe ce dernier (détruire l'horloge).

Méthode correspondant au bouton CANCEL de la boîte de dialogue

Appeler la méthode correspondant au bouton CANCEL héritée de TDialog

Méthode d'initialisation des valeurs des paramètres des équations (fenêtre des paramètres)

Si c'est une équation de droite
Alors
Mettre
paramètreA ← 1
paramètreB ← 1
paramètreC ← 0

Si c'est une équation du second degré ou trigonométrique
Alors
Mettre
paramètreA ← 1 (paramètreA ≠ 0)
paramètreB ← 1
paramètreC ← 1

Positionner les barres de défilement correspondant aux valeurs des paramètres
Afficher les valeurs des paramètres dans la fenêtre des paramètres

Méthode correspondant à la boîte liste des droites

Aller voir quel item est sélectionné
L'afficher dans la fenêtre des énoncés
Mettre à jour la zone client de la fenêtre des graphes
Initialiser les valeurs des paramètres (voir la méthode précédente)
Afficher le dessin graphique correspondant

Idem pour la méthode correspondant à la boîte liste des paraboles

Idem pour la méthode correspondant à la boîte liste des courbes trigonométriques

Méthode de balayage des boîtes listes de la boîte de dialogue

Paramètres : Handle (identifiant) de la boîte liste (HDlg)
L'indexe d'un item de la boîte liste (ValeurIndexItem)

Sélectionner l'item dont la valeur de l'indexe se trouve dans « ValIndexItem » de la boîte liste dont l'identifiant est « HDlg »

Méthode de déclenchement de l'horloge (pour le balayage)

```
Aller à la boîte liste des droites
  Aller rechercher le nombre d'items dans la liste
    (NombreItems1 ← nombre d'items de la liste)
  Balayer le premier item de la liste
  Si on n'est pas encore au bout de la liste
    Alors
      Balayer le reste des items de la liste
    Sinon
      Aller à la boîte liste des paraboles
        Aller rechercher le nombre d'items dans la liste
          (NombreItems2 ← nombre d'items de la liste)
        Balayer le premier item de la liste
        Si on n'est pas encore au bout de la liste
            Alors
              Balayer le reste des items de la liste
            Sinon
              Aller à la boîte liste des courbes trigonométriques
                  Aller rechercher le nombre d'items dans la liste
                    (NombreItems3 ← nombre d'items de la liste)
                  Balayer le premier item de la liste
                  Si on n'est pas encore à la fin de la liste
                    Alors
                      Balayer le reste des items de la liste
                    Sinon
                      Revenir à la première boîte liste des droites
                      Et recommencer la balayage
                  Fin Si
        Fin Si
    Fin Si
```

Remarque : Cette méthode ne fait que balayer une liste d'éléments dans différentes boîtes listes. Il faut encore voir le cas où l'élève clique sur le bouton OK (voir méthode à la page 53)

Méthode de destruction de la boîte de dialogue

[Relâcher les différents handles des différents contrôles de la boîte de dialogue]
Appeler la méthode de destruction WMDestroy héritée de TDialog

5.2.2 Unité : Gestion des paramètres

5.2.2.1 Objet : Fenêtre des paramètres

But : Créer une fenêtre contenant un titre et un menu système
Détruire cette fenêtre à la fin de l'application
Ajouter du texte statique, des boîtes d'édition et des barres de défilement
Les dessiner dans la fenêtre
Traiter les messages des barres de défilement

PFenParamètres = TFenParamètres
TFenParamètres = OBJECT(TWindow)

Champs : des variables pointeurs de type PStatic

Méthodes : Contructeur Init
Destructeur Done
Procédure d'initialisation de la fenêtre SetUpWindow
Procédures de réponses aux messages des barres de défilement
(3 fois, une pour chaque barre de défilement)
Procédure de balayage des barres de défilement
Procédure de déclenchement de l'horloge WMTimer
Procédure de destruction de l'horloge WMDestroy

Méthode de création de la fenêtre des paramètres

Appeler la méthode de création Init héritée de TWindow
Modifier les attributs des coordonnées de la fenêtre (dimensions en x, y, w, h)
Modifier les attributs du style de la fenêtre
Créer les textes statiques (un titre par paramètre)
Créer les boîtes d'édition (une par paramètre)
Créer les barres de défilement (un par paramètre)

Méthode d'initialisation de la fenêtre SetUpWindow

Appeler la méthode SetupWindow héritée de TWindow
Initialiser le positionnement des barres de défilement
Déterminer les intervalles des valeurs des barres de défilement
Initialiser les boîtes d'édition avec les valeurs par défaut des paramètres de l'équation en cours

[Si on doit faire le balayage des barres de défilement, il faut initialiser ici l'intervalle de temps pendant lequel un élément est balayé] → SetTimer

Méthode de réponse aux messages des barres de défilement

Regarder le positionnement de la barre de défilement en cours
En retirer sa valeur et l'afficher dans la boîte d'édition correspondant
Affecter cette valeur au paramètre correspondant (paramètre X ← valeur Y)
Mettre à jour la zone client de la fenêtre des graphes
Retracer le dessin graphique avec la nouvelle valeur du paramètre
Mettre à jour la zone client de la fenêtre des calculs des propriétés
Afficher les nouveaux résultats des calculs mathématiques

Remarque :

Cette méthode est valable pour un barre de défilement correspondant à un paramètre, il faut donc autant de traitement de messages de barres de défilement qu'il y a de paramètres. Dans notre cas, nous aurons trois barres de défilement correspondant aux trois paramètres.

Méthode de balayage d'un barre de défilement

Paramètre : le handle du barre de défilement, ici son identifiant (HDlg)

Aller rechercher l'identifiant du barre de défilement
(HDlg ← identifiant du barre de défilement)
Aller positionner le barre de défilement à la valeur du paramètre correspondant
Afficher cette valeur dans la boîte d'édition correspondante

Méthode de déclenchement de l'horloge WMTimer

Si on n'est pas à la borne supérieure de l'intervalle du barre de défilement
Alors
Incrémenter la valeur du paramètre de 1
Balayer le barre de défilement (voir méthode précédente)
Sinon
Réinitialiser la valeur du paramètre
Et recommencer le balayage

Méthode de destruction de l'horloge (WMDestroy)

Détruire l'horloge qu'on a défini
Appeler la méthode WMDestroy héritée de TWindow

Méthode de destruction de la fenêtre des paramètres

Appeler la méthode Done héritée de TWindow

5.2.3 Unité : Gestion des calculs mathématiques

5.2.3.1 Objet : Fenêtre des calculs mathématiques

But :

- Calculer les différentes propriétés mathématiques des équations d'une droite, d'une parabole et celles d'une courbe trigonométrique.
- Les afficher dans la fenêtre des calculs.
- Au cas de modification d'une des valeurs des paramètres de l'équation, mettre à jour les calculs et réafficher les nouveaux résultats.

PFenCalculsProp = TFenCalculsProp
TFenCalculsProp = OBJECT (TWindow)

Champs : CalculsPropDC : HDC
le handle du contexte de périphérique de la fenêtre des calculs

Méthodes : Constructeur Init
Destructeur Done
Procédure d'initialisation de la fenêtre SetUpWindow
Procédure de calculs des propriétés d'une droite
Procédure de calculs des propriétés d'une parabole
Procédure de calculs des propriétés d'une courbe trigonométrique
Procédure d'affichage des différents calculs dans la fenêtre

Méthode de création de la fenêtre des calculs

Appeler la méthode de création Init héritée de TWindow
Modifier les attributs des coordonnées de la fenêtre
Modifier les attributs du style de la fenêtre

Méthode d'initialisation de la fenêtre SetUpWindow

Appeler la méthode SetUpWindow héritée de TWindow
Mettre à jour la fenêtre
Aller rechercher le handle du contexte de périphérique de la fenêtre en vue d'afficher les résultats des calculs
(CalculsPropDC ← handle de la fenêtre)

Méthode de calculs des propriétés d'une droite

Calculer le coefficient angulaire
(CoefAng ← -b/a)
Calculer l'intersection avec l'axe des X
(InterX ← [-b/a,0])
Calculer l'intersection avec l'axe des Y
(InterY ← [0,c])
Afficher les différents résultats des calculs

Méthode de calculs des propriétés d'une parabole

Calculer le discriminant (r_0)
r_0 ← (b*b) - (4*a*c)

Calculer l'intersection avec l'axe des X
Tester la valeur de r_0
Si $r_0 > 0$
InterX1 ← -b + SQRT(a) / 2*a
InterX2 ← -b - SQRT(a) / 2*a
Si $r_0 = 0$
InterX0 ← -b / 2*a
Si $r_0 < 0$
Il n'y a pas d'intersection avec l'axe des X

Calculer l'intersection avec l'axe des Y
InterY ← c

Calculer le sommet de la courbe
SommetX ← -b / 2*a
SommetY ← [(b*b) - (4*a*c)] / 4*a

Calculer l'axe de symétrie
AxeSym ← -b / 2*a

Afficher les différents résultats des calculs

Méthode de calculs des propriétés d'une courbe trigonométrique

Cette partie n'a pas encore été implémentée.

Méthode d'affichage des calculs

Dans le cas où c'est une équation de droite
Appeler la méthode de calculs des propriétés d'une droite

Dans le cas où c'est une équation d'une parabole
Appeler la méthode de calculs des propriétés d'une parabole

Dans le cas où c'est une équation d'une courbe trigonométrique
Appeler la méthode de calculs des propriétés d'une courbe trigonométrique

Méthode de destruction de la fenêtre

Relâcher le handle du contexte de périphérique de la fenêtre (CalculsPropDC)
Appeler la méthode de destruction Done héritée de TWindow

5.2.4 Unité : Gestion des dessins graphiques

5.2.4.1 Objet : Fenêtre des dessins graphiques

But : Créer une fenêtre en vue d'afficher des dessins graphiques

PFenGraphe = TFenGraphe
TFenGraphe = OBJECT (TWindow)

Méthodes : Constructeur Init de la fenêtre
Procédure de mise à jour de la fenêtre SetUpWindow
Procédure d'affichage des dessins graphiques

Méthode de création de la fenêtre

Appeler la méthode de création Init héritée de TWindow (HWnd, Titre)
Modifier les attributs des coordonnées de la fenêtre (x, y, w, h)
Modifier les attributs du style de la fenêtre

Méthode de mise à jour de la fenêtre SetUpWindow

Appeler la méthode de mise à jour SetUpWindow héritée de TWindow
Mettre à jour la fenêtre de dessins

Méthode d'affichage des dessins graphiques

Demander à l'objet de dessin de s'afficher

5.2.4.2 Objet : Dessiner la fonction

But : En fonction de l'énoncé de l'équation en fournir son dessin graphique

PDessinFonction = TDessinFonction
TDessinFonction = OBJECT (TObject)

Méthodes : Procédure de dessin d'une droite
Procédure de dessin d'une parabole
Procédure de dessin d'une courbe trigonométrique
Procédure d'affichage de dessins

Méthode correspondant au dessin d'une droite

Tracer les axes (en connaissant les dimensions de la fenêtres des graphes)
Tracer les graduations
Tracer la courbe (en connaissant l'équation en cours et les valeurs de ses paramètres)
Tracer les propriétés mathématiques de la courbe (en connaissant les valeurs des paramètres)

Méthode correspondant au dessin d'une parabole

Idem que la méthode précédente

Méthode correspondant au dessin d'une courbe trigonométrique

Idem que la méthode précédente

5.2.5 Unité : Gestion des balayages

But 1 :

- Gérer la hiérarchie de choix dans les menus qui sont balayés.

Chaque menu (celui destiné à l'élève) voit ses différentes actions balayées. Chaque action balayée et sélectionnée peut déclencher trois autres types d'actions : soit le balayage d'un sous-menu, soit l'exécution de l'action elle-même, soit l'action correspond à une touche arrière permettant à l'élève de revenir à la fenêtre précédente.

Pour bien gérer une telle hiérarchie, trois règles sont à respecter :

- ♦ les menus ne doivent pas être trop longs;
- ♦ les actions les plus fréquentes doivent être placées en tête de la liste;
- ♦ on doit permettre une pause assez long sur la première action de telle sorte que l'élève ait le temps de faire son choix.

Il nous faudrait donc quatre méthodes :

- La méthode permettant de passer au sous-menu suivant
  Elle doit stopper le balayage de son menu
  Et déclencher le balayage du sous-menu appelé
- La méthode permettant de revenir en arrière
  Elle doit stopper le balayage (détruire l'horloge)
  Fermer la fenêtre et revenir à la fenêtre précédente
- La méthode d'exécution d'action
  Elle doit exécuter l'action en question

- La méthode remplaçant le bouton gauche de la souris, elle jouera le rôle de contact ou de clic permettant à l'élève de faire son choix dans les menus balayés.

Au dessus de tout ceci se trouve une pile, c'est-à-dire un pointeur contenant les différents identifiants. En appelant la méthode d'ouverture de sous-menu, elle ajoute un élément (identifiant du menu) dans sa liste et en appelant la méthode de retour en arrière, elle retire de l'identifiant du menu courant de la liste.

But 2 :

- Créer une boîte de dialogue permettant à l'utilisateur d'aller sélectionner une vitesse de balayage pour chaque type de fenêtres définis (une fenêtre, une boîte de dialogue, une boîte de message).

Remarque : Cette boîte de dialogue ressemble beaucoup à la fenêtre des paramètres.

De cette boîte de dialogue, on en retire les valeurs de vitesse des balayages.

5.2.6 Unité : Gestion des données

Cette unité a pour but de définir les différentes variables globales partagées par les différentes unités de l'application.

5.2.7 Unité : Palette d'actions

Le but de cette unité est :

1) Créer une fenêtre vide ayant un titre et un menu système
2) Créer une série de boutons graphiques (icônes graphiques)
   Chaque icône graphique correspond à une action.
3) Les sauver dans une liste [une collection]
4) Afficher ces boutons graphiques dans la palette
5) Doter ces boutons graphiques d'une méthode de réponse aux messages du bouton gauche de la souris (se mettre en mode sélectionné)
6) Dessiner un rectangle autour d'un bouton graphique s'il est sélectionné (c'est-à- dire s'il répond au message du bouton gauche de la souris)

5.2.7.1 Objet : La fenêtre de la palette d'actions

But : Créer une fenêtre vide afin de la remplir de boutons graphiques et de les sélectionner.

PFenPaletteOutil = TFenPaletteOutil
TFenPaletteOutil = OBJECT (TWindow)

Méthodes : Procédure de création d'une fenêtre vide
Procédure de remplissage de la fenêtre palette d'actions
Procédure de sélection d'un bouton graphique de la fenêtre palette

Méthode de création d'une fenêtre vide

Appeler la méthode de création Init héritée de TWindow (HWnd, Titre)
Modifier les attributs des coordonnées de la fenêtre (x, y, w, h)
Modifier les attributs du style de la fenêtre

Méthode de remplissage de la fenêtre d'outils graphiques

Définir une liste (collection) de boutons graphiques
Aller rechercher dans la liste l'identifiant les boutons graphiques
Leur demander de s'afficher dans la palette

Méthode de sélection d'un bouton graphique de la fenêtre palette

Demander au bouton graphique de la palette qui a reçu un message du bouton gauche de la souris de se mettre en mode de sélection (dessiner un rectangle autour du bouton graphique)

5.2.7.2 Objet : La fenêtre outil de la palette

But : Afficher les boutons graphiques dans la palette d'actions

Remarque : Chaque outil correspond à un bouton graphique

PFenOutilPalette = TFenOutilPalette
TFenOutilPalette = OBJECT(TButton)

Méthodes : Procédure de création de boutons graphiques
Procédure de sélection d'un bouton graphique

Méthode de création de boutons graphiques

Se positionner dans la fenêtre palette d'actions
Créer le bouton graphique à cette position (le dessiner)

Méthode de sélection d'un bouton graphique

Aller rechercher l'identifiant du bouton graphique (en vue de dessiner un rectangle autour de lui)

5.2.7.3 Objet : La palette d'outils

But : remplir sa fenêtre des boutons graphiques

PPaletteOutil = TPaletteOutil
TPaletteOutil = OBJECT (TFenPaletteOutil)

Méthodes : Constructeur Init (ajouter un bouton graphique dans la palette)

Méthode de création d'outils dans la palette d'actions

Appeler la méthode de TFenPaletteOutil qui remplit la palette en lui donnant le bouton graphique (en connaissant son identifiant)

5.2.7.4 Objet : Les outils de la palette

But : répondre aux messages du bouton gauche de la souris c'est-à-dire se mettre en mode sélectionné.

Remarque : Chaque outil correspond à un bouton graphique

POutilPalette = TOutilPalette
TOutilPalette = OBJECT (TFenOutilPalette)

Méthodes : Procédure de réponse aux messages du bouton gauche de la souris (WMLButtonDown)

Méthode de réponse aux messages du bouton gauche de la souris

Appeler la méthode de sélection de TFenOutilPalette qui fournit l'identifiant du bouton graphique (en vue de dessiner un rectangle autour de lui)

5.2.8 Unité : Gestion des aides des actions

5.2.8.1 Objet : Fenêtre d'explication des actions

But : Le but de ce objet est de créer une fenêtre permettant d'afficher du texte en vue de donner des explications ponctuelles sur les actions entreprises par l'utilisateur.

PFenExplications = TFenExplications
TFenExplications = OBJECT (TWindow)

Champs : Le handle du contexte de périphérique de la fenêtre (HDC)

Méthodes : Construction de la fenêtre (constructeur Init)
Destruction de la fenêtre (destructeur Done)

Méthode de création de la fenêtre

Appeler la méthode de création Init héritée de TWindow (HWnd, Titre)
Modifier les attributs des coordonnées de la fenêtre (x, y, w, h)
Modifier les attributs du style de la fenêtre

Méthode de destruction de la fenêtre

Relâcher le handle du contexte de périphérique (HDC) de la fenêtre
Appeler la méthode de destruction Done héritée de TWindow

5.2.9 Unité : Gestion des aides « on line »

Cette partie n'a pas encore été implémentée.

6. Création de l'interface

6. Création de l'interface

6.1 Introduction

Notre but est de faire une présentation des fenêtres de notre application et une présentation des enchaînements de ces fenêtres.

Etant donné que nos futurs utilisateurs sont des personnes handicapées moteur, nous allons concevoir notre interface en deux temps : premièrement, l'interface destinée à l'élève et deuxièmement, celle destinée à l'enseignant. Pour chaque type d'interface, nous ferons tout d'abord une présentation visuelle, ensuite une présentation des fonctionnalités et enfin nous présenterons l'enchaînement des fenêtres.

Pour la présentation des fenêtres, nous essayerons de suivre un ordre chronologique du déroulement de l'application.

6.2 Le but recherché

Rappelons que nous voulons aider les élèves à acquérir une certaine adaptation voire une certaine réflexe lors de l'utilisation de notre logiciel. Par conséquent, nous devons suivre une certaine structure de cohérence lors de la création de notre interface (voir points 3.2.5.1. et 3.2.5.2.).

6.3 Différents types de fenêtres

6.3.1 Type de fenêtre standard

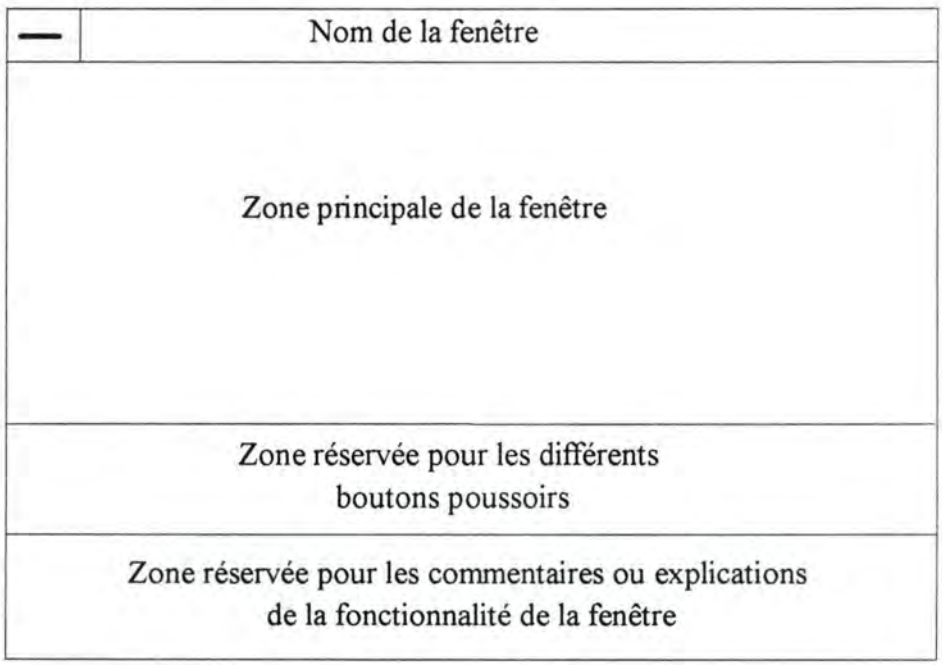


Figure 6.1 : Type de fenêtre standard

Chaque fenêtre aura un nom et peut être fermée à tout instant. Elle sera répartie en trois zones bien définies : juste après le titre de la fenêtre sera sa zone principale, elle présentera les différentes actions que l'utilisateur peut effectuer comme sélectionner un élément dans une boîte liste. La zone en dessous sera réservée aux boutons poussoirs tels que le bouton « OK », le bouton « Annuler » ou le bouton « Aide ». Et la dernière zone sera destinée à donner quelques commentaires ou explications aidant l'utilisateur dans son action. Afin de faciliter le travail de l'utilisateur, un bouton poussoir par défaut sera défini, le bouton de validation « OK » par exemple ou le bouton « Retour Arrière ». De cette façon, l'utilisateur peut acquérir une certaine habitude après un certain temps d'utilisation du logiciel.

6.3.2 Premier type de fenêtre

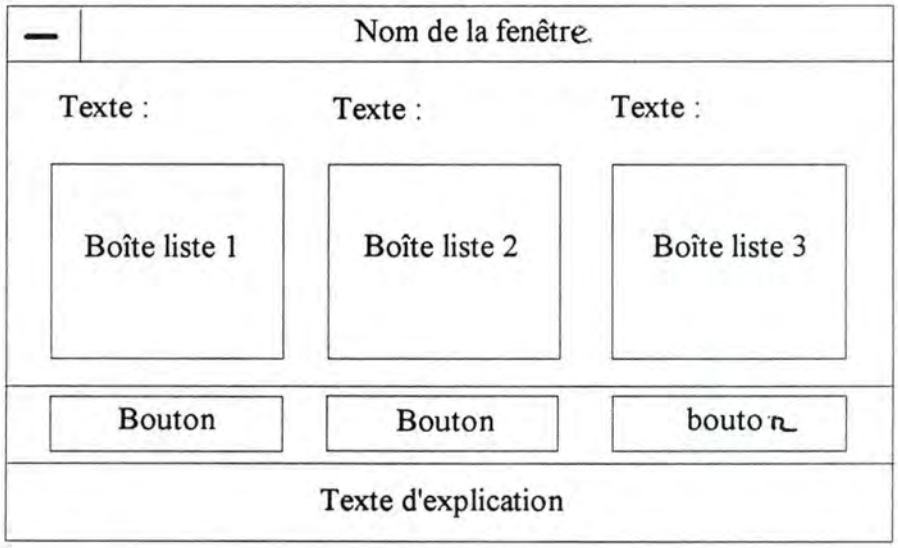


Figure 6.2 : Présentation du premier type de fenêtre

Les boîtes de dialogue auront cette structure de fenêtre. Elles permettront à l'utilisateur de sélectionner un des éléments des boîtes listes et de valider son choix. Pour l'aider dans sa tâche, une explication sera affichée en bas de la fenêtre.

6.3.3 Deuxième type de fenêtre

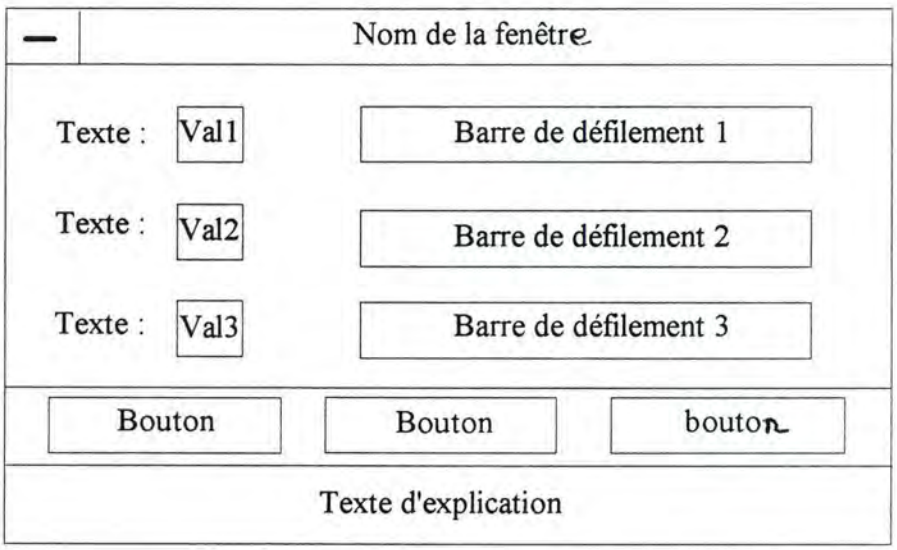


Figure 6.3 : Présentation du deuxième type de fenêtre

Ce type de fenêtre servira également lors de la création de boîtes de dialogue. A la différence de la précédente, celle-ci permettra à l'utilisateur de sélectionner une valeur à l'aide des barres de défilement. Et comme la fenêtre précédente, des boutons poussoirs lui permettront de valider ou d'annuler son choix. Et s'il rencontre des problèmes dans son action, un texte explicatif est présent pour l'aider.

6.3.4 Troisième type de fenêtre

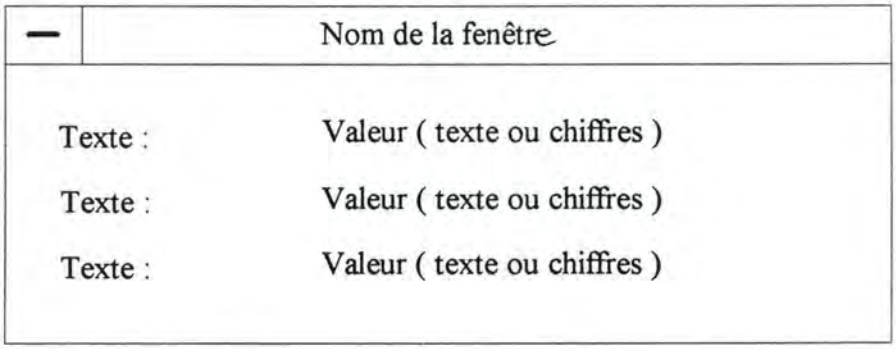


Figure 6.4 : Présentation du troisième type de fenêtre

Ce type de fenêtre ne permet pas une interaction active de l'utilisateur : en effet, elle ne sert qu'à afficher des résultats des calculs de propriétés mathématiques qui accompagnent l'équation algébrique en cours. Et à chaque modification, que ce soit au niveau des paramètres ou du graphique, le résultat des calculs sera modifié et réaffiché en fonction des modifications des valeurs des paramètres.

6.3.5 Quatrième type de fenêtre

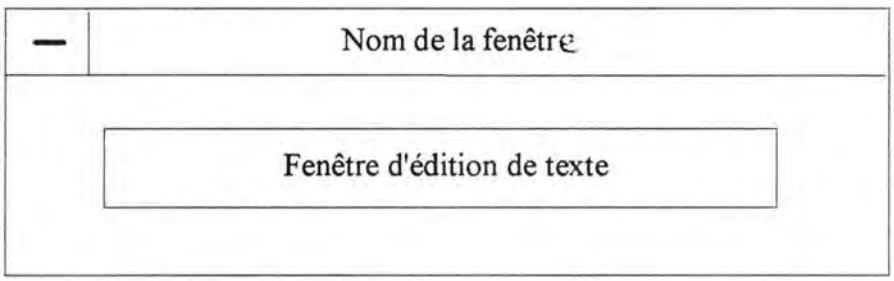


Figure 6.5 : Présentation du quatrième type de fenêtre

Ce type de fenêtre affiche l'énoncé de l'exercice que l'utilisateur a sélectionné dans une boîte de dialogue. Comme la précédente, elle ne permet pas une interaction active de la part de l'utilisateur.

6.3.6 Cinquième type de fenêtre

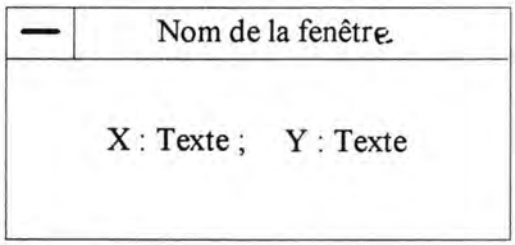


Figure 6.6 : Présentation du cinquième type de fenêtre

Cette fenêtre permet d'afficher les coordonnées d'un point sur l'écran graphique. A chaque déplacement du point sur l'écran graphique, ce dernier voit ses coordonnées changées.

6.3.7 Sixième type de fenêtre

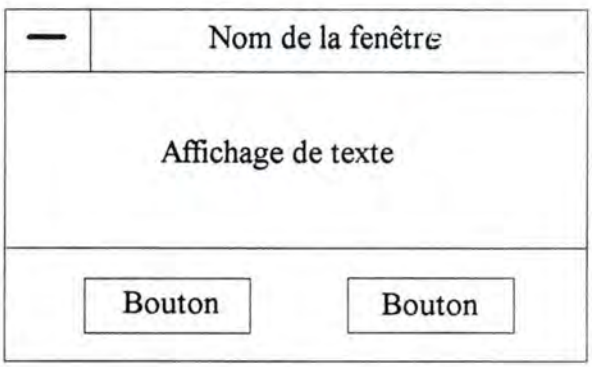


Figure 6.7 : Présentation du septième type de fenêtre

Ce type de fenêtre remplace la boîte de message que l'on a l'habitude de rencontrer sous Windows. En effet, à la différence d'une boîte de message normale, celle-ci sera dotée en plus d'un balayage automatique.

6.3.8 Septième type de fenêtre

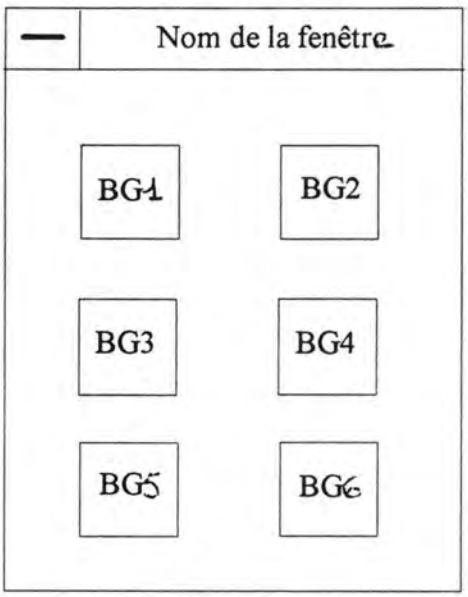


Figure 6.8 : Présentation du sixième type de fenêtre

Cette fenêtre joue un rôle très important pour l'élève. Elle peut être vue comme la palette du menu principal de l'élève. Effectivement, cette palette sera constituée de boutons graphiques qui reprendra les principales commandes de la barre du menu principal de l'application. Cette fenêtre est réservée à l'élève handicapé moteur. Pour que ce dernier puisse l'utiliser, chaque bouton graphique sera balayé par le système.

6.4 L'interface de l'élève

6.4.1 Présentation et fonctionnalités des fenêtres

Nous allons maintenant présenter quelques fenêtres de notre application pour illustrer les types de fenêtres que nous avons définis.

6.4.1.1 La sélection de l'énoncé d'un exercice

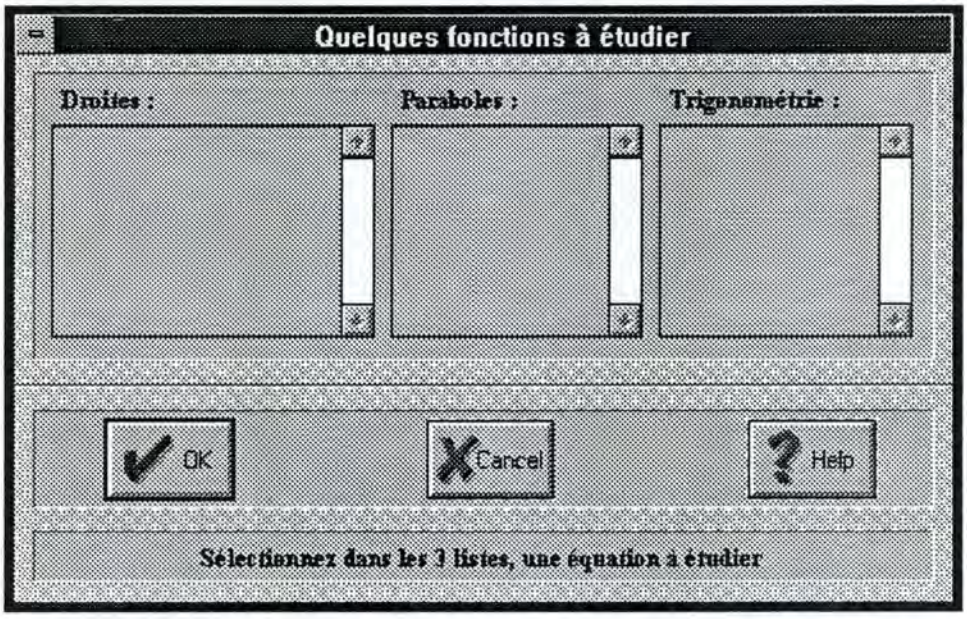


Figure 6.9 : La sélection de l'énoncé d'un exercice

Ayant choisi de commencer un nouvel exercice, c'est la fenêtre ci-dessus qui se présentera à l'utilisateur. Il devra parmi les familles de fonctions, choisir celle qu'il veut étudier.

Le bouton « OK » lui permet de valider son choix, le bouton « CANCEL » lui permet de revenir à la fenêtre précédente et le bouton « Help » lui permet d'aller consulter l'aide du logiciel quand il l'estime nécessaire.

6.4.1.2 La fenêtre d'affichage de l'énoncé

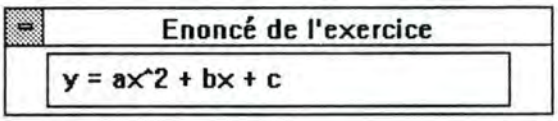


Figure 6.10 : La fenêtre d'affichage de l'énoncé d'exercice

6.4.1.3 La fenêtre des paramètres

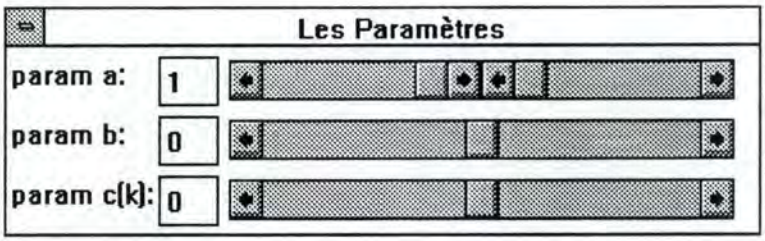


Figure 6.11 : La fenêtre des paramètres

Cette fenêtre permet d'affecter ou de modifier les valeurs des paramètres de la fonction que l'utilisateur est en train de traiter. Elle permet à l'enseignant d'affecter ou de modifier manuellement les valeurs des paramètres. Quant à l'élève, par le déclenchement du balayage à partir de la palette d'actions, il peut aller affecter ou de modifier des valeurs aux paramètres.

6.4.1.4 Fenêtre des calculs mathématiques

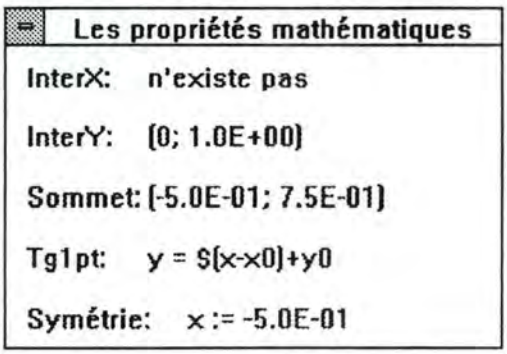


Figure 6.12 : Présentation de la fenêtre des calculs mathématiques

6.4.1.5 La fenêtre des graphes

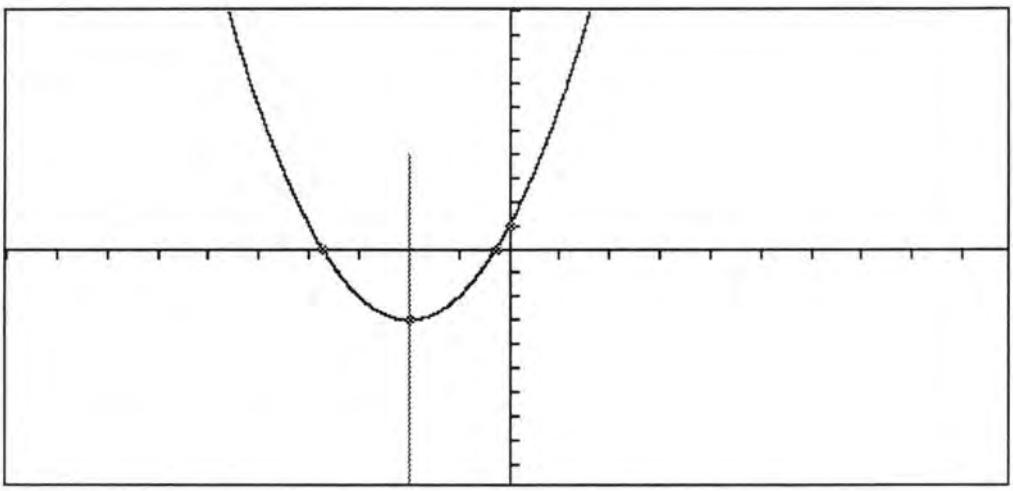


Figure 6.13 : Présentation de la fenêtre des graphes

Dans l'état actuel de notre application, cette fenêtre ne permet pas encore une interaction active de l'utilisateur. En effet, elle n'affiche que les dessins graphiques correspondant à l'équation en cours. Lors des modifications des valeurs des paramètres, elle modifie son dessin graphique de manière dynamique.

6.4.1.6 La fenêtre des coordonnées

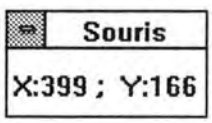


Figure 6.14 : Présentation de la fenêtre des coordonnées

Cette fenêtre affiche les coordonnées de la souris lorsque l'utilisateur clique sur le bouton gauche de la souris dans la fenêtre des graphes.

6.4.1.7 La palette d'actions

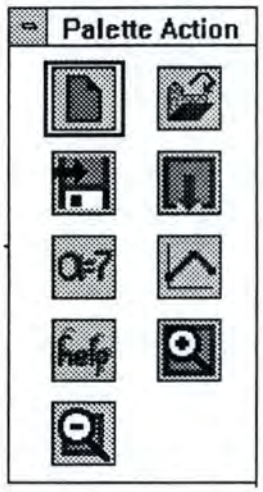


Figure 6.15 : Présentation de la palette d'actions

Cette palette est composée de boutons graphiques qui permettent d'exécuter des actions bien précises. Ces boutons graphiques reprennent quelques commandes de la barre du menu principal de l'application. En effet, le premier bouton permet de « Commencer » un nouvel exercice, le deuxième d'« Ouvrir » un exercice existant, le troisième de« Sauver » un exercice sur disque, le quatrième de « Quitter » le programme, le cinquième de déclencher le balayage de la fenêtre des paramètres en vue d'affecter ou de modifier leurs valeurs, lors des modifications des valeurs des paramètres, le dessin graphique se modifie également, de même que les valeurs de ses propriétés mathématiques. Le sixième bouton déclenche le balayage de la fenêtre des graphes en vue de sélectionner les objets dessins pour les déplacer dans la fenêtre. Les trois derniers boutons correspondent au bouton d'aide de l'application, au bouton de visualisation plus grand et plus petit du dessin graphique.

6.4.1.8 La fenêtre des messages

Figure 6.16 : Présentation de la fenêtre des messages

6.4.1.9 Fenêtre de configuration des balayages

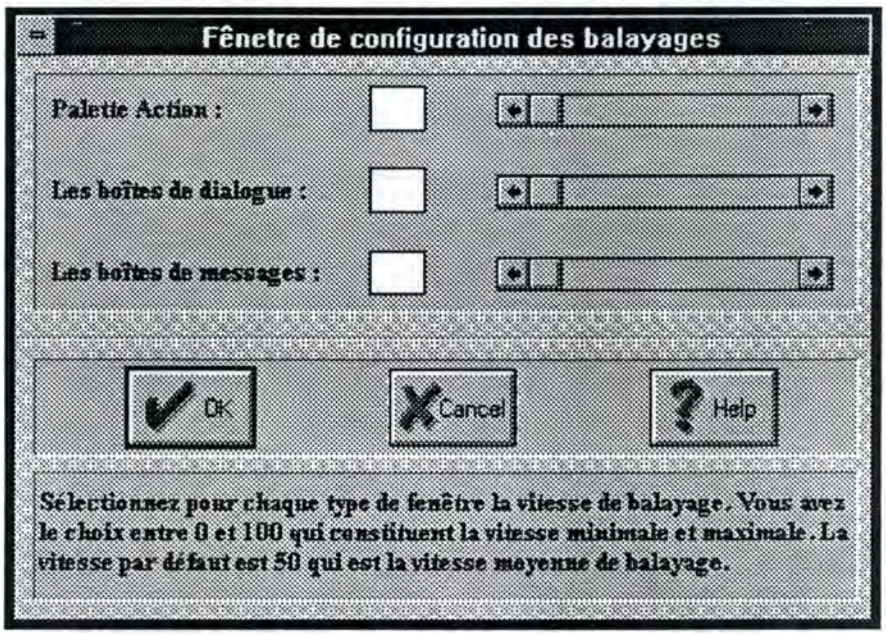


Figure 6.17 : Présentation de la fenêtre de configuration des balayages

Cette fenêtre offre les mêmes fonctionnalités que la fenêtre des paramètres. Ce qui les différencie, c'est leur objectif, l'une permet la manipulation des paramètres et l'autre permet la configuration des vitesses de balayage. Elles se différencient également dans leur classe, l'une est de classe des boîtes de dialogue et l'autre celle des fenêtres.

6.4.1.10 Fenêtre d'ouverture de fichiers

Figure 6.18 : Présentation de la fenêtre d'ouverture de fichiers

Cette fenêtre propose les mêmes fonctionnalités que la fenêtre d'ouverture de fichiers de Windows à la différence près que celle-ci est dotée d'un balayage.

La fenêtre de sauvegarde des fichiers est identique à cette dernière.

6.4.2 Enchaînement des fenêtres

6.4.2.1 Introduction

Maintenant que nous avons présenté quelques unes de nos fenêtres avec leurs fonctionnalités, nous allons mettre en schémas les enchaînements de toutes ces fenêtres.

Notre objectif est, d'une part, de mettre sur graphes les différents enchaînements de nos fenêtres afin d'en définir l'ordre d'apparition. De cette manière, nous aurons une vision claire et complète de notre interface. D'autre part, ces graphes nous permettront de définir les liens logiques existant entre toutes les fenêtres, ce qui ne pourra que nous faciliter dans notre tâche de programmation.

6.4.2.2 Légendes des graphes d'enchaînement

Pour chaque graphe, nous donnons la signification suivante :

Le rectangle représente une fenêtre et il porte le même nom que la fenêtre.

La flèche signifie « un item de menu est sélectionné ». Par exemple la sélection d'un des boutons poussoirs d'une fenêtre.

Le texte sans cadre reprend l'intitulé des items des menus.

6.4.2.3 Commencer un nouvel exercice

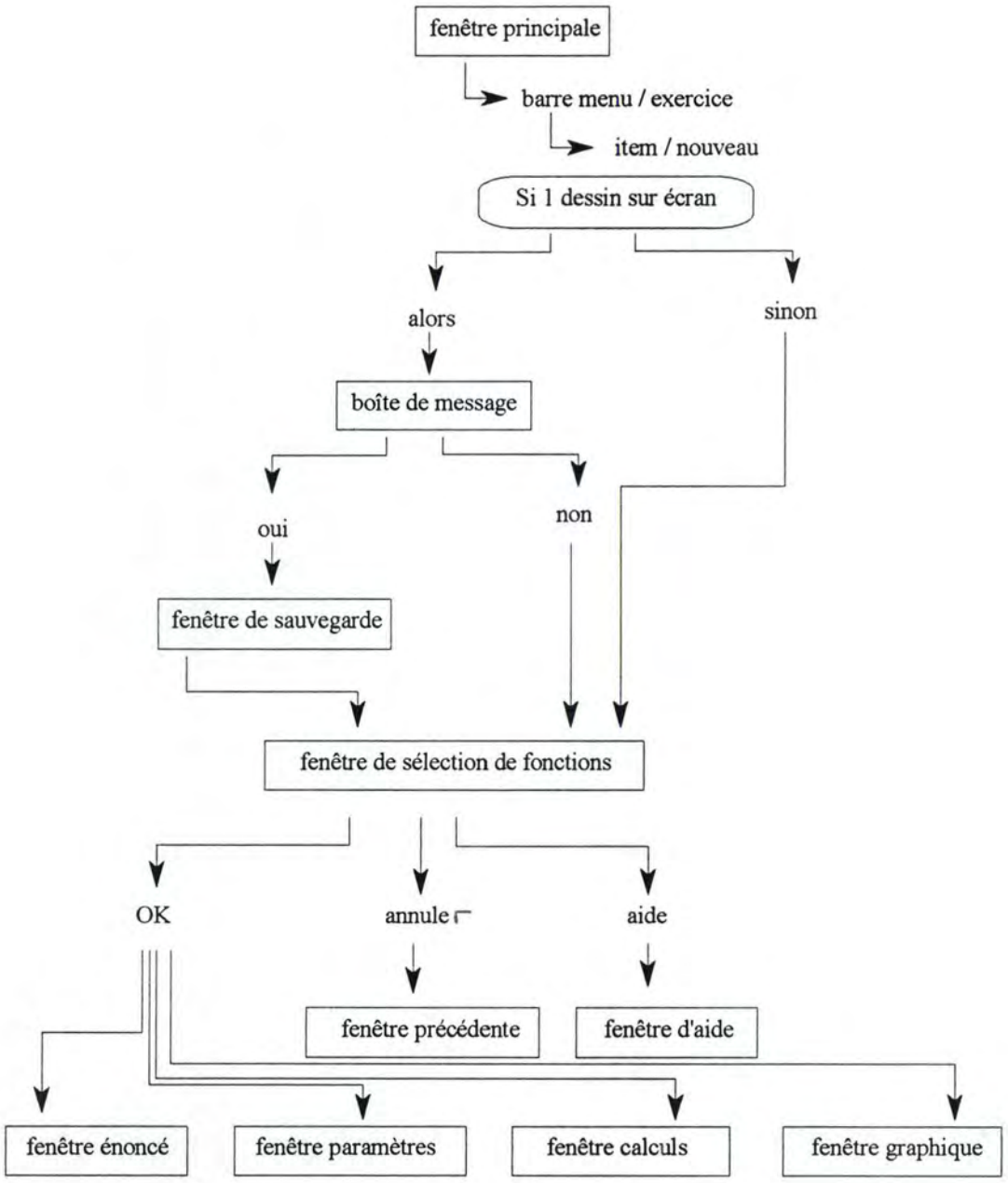


Figure 6.19 : L'enchaînement des fenêtres pour un nouvel exercice

6.4.2.4 Continuer un exercice existant

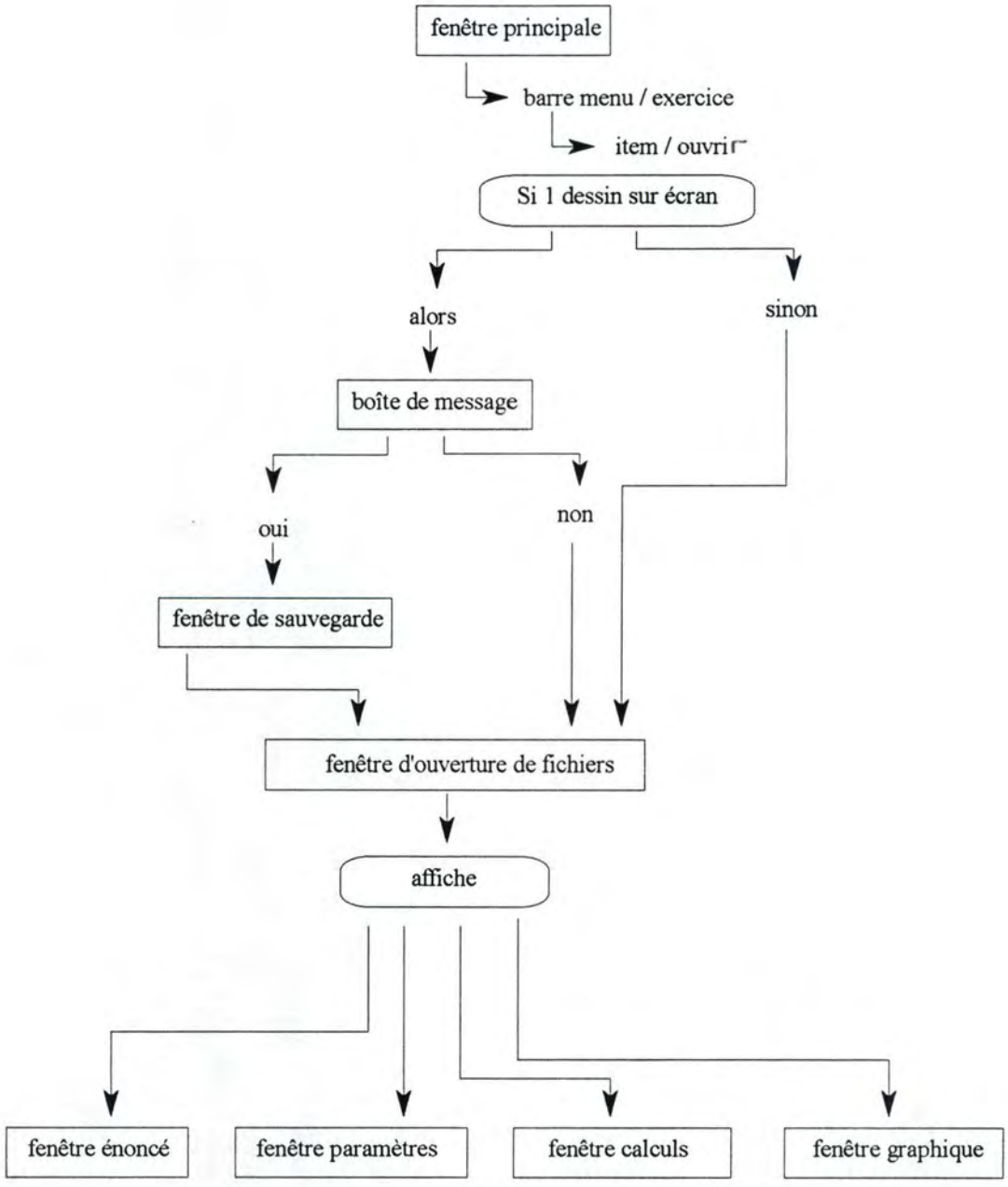


Figure 6.20 : Continuer un exercice existant

6.4.2.5 Quitter le programme

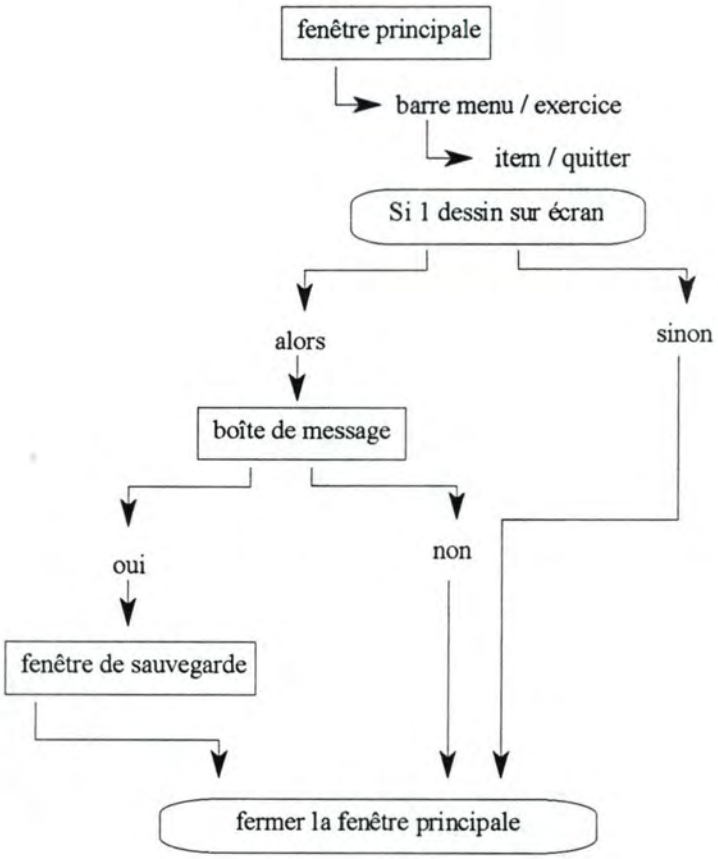


Figure 6.21 : Quitter le programme

6.4.2.6 Configuration des balayages

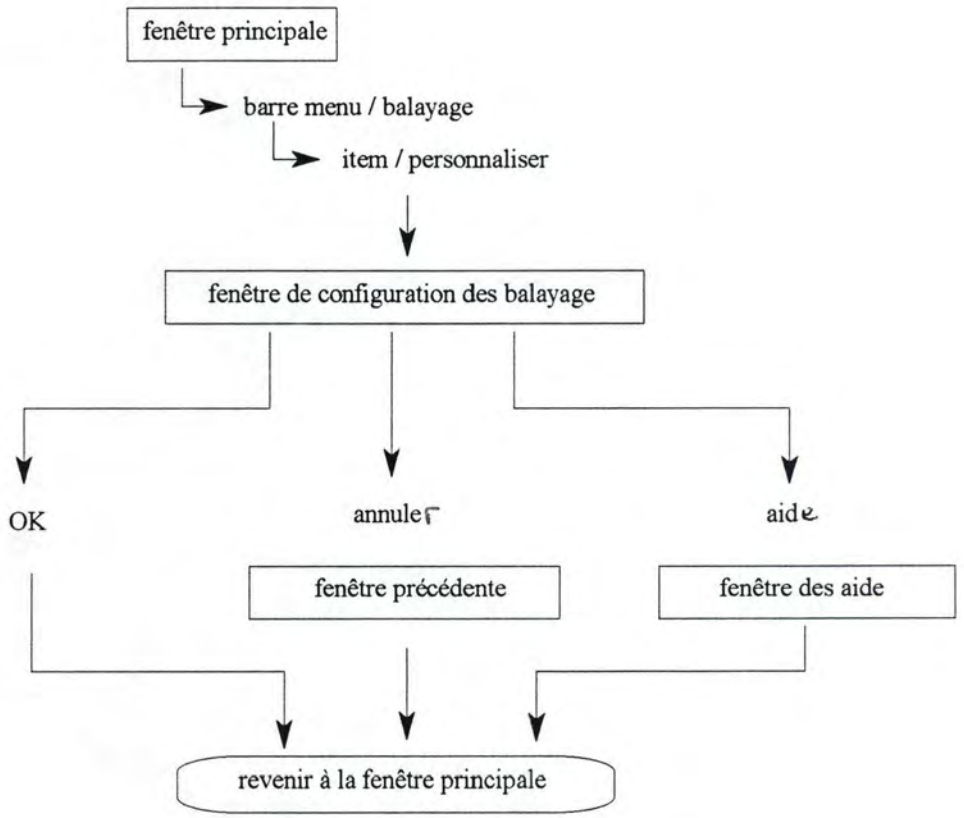


Figure 6.22 : Enchaînement des fenêtres pour la configuration des balayages

6.5 L'interface de l'enseignant

Voici quelques fenêtres de l'enseignant.

6.5.1 Fenêtre d'ouverture de fichiers

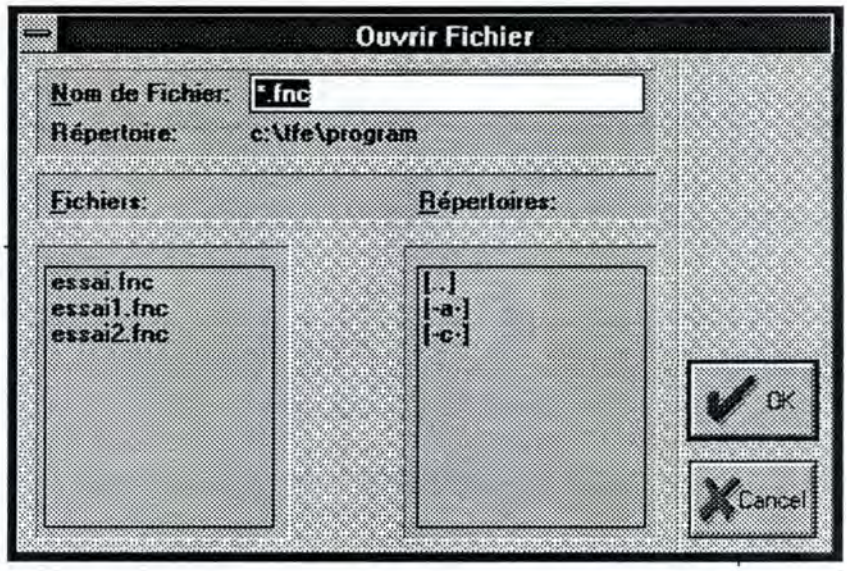


Figure 6.23 : Présentation de la fenêtre d'ouverture de fichiers

6.5.2 Fenêtre de sauvegarde de fichiers

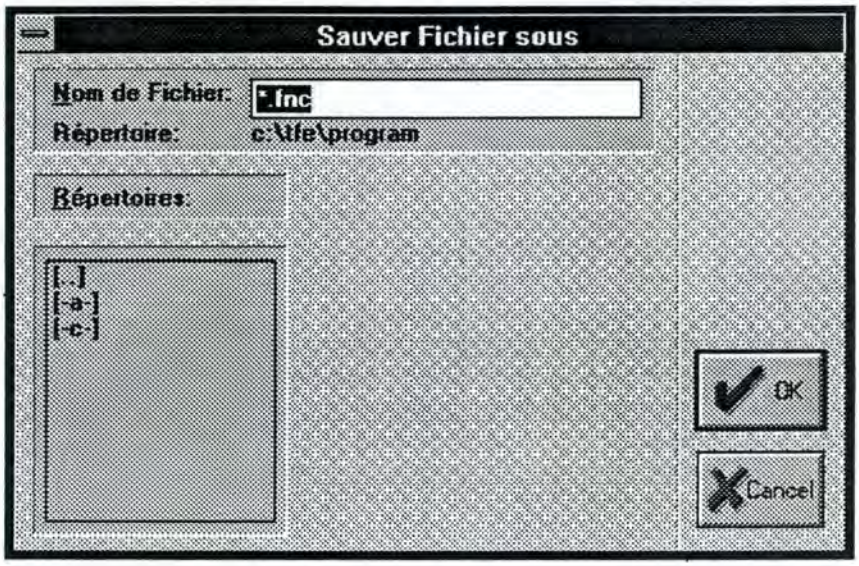


Figure 6.24 : Présentation de la fenêtre de sauvegarde de fichiers

6.5.3 Fenêtre de message

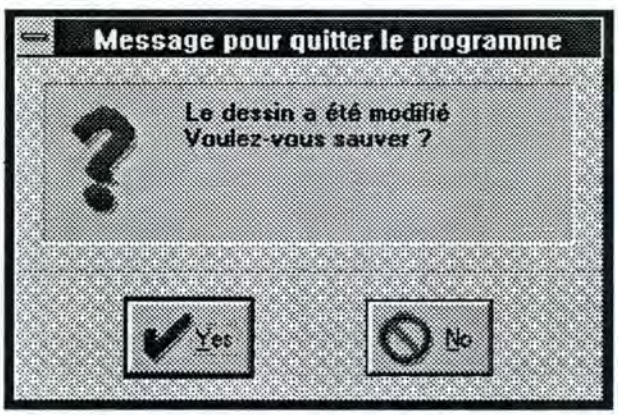


Figure 6.25 : Présentation de la fenêtre de message

7. Conclusion

7. Conclusion

7.1 L'objectif de départ

7.2 Performance du programme

7.3 Les extensions

7.4 L'aspect positif du travail

7.1 L'objectif de départ

Au niveau de l'implémentation de l'application, nous n'avons pas pu aller jusqu'au bout de notre objectif. Mais afin de pouvoir tester l'application dans son état actuel, nous nous sommes arrêtés aux points suivants :

7.1.1 Au niveau mathématique

Nous nous sommes limités aux formes d'équations les plus simples, à savoir :

$y = ax + b$	pour la fonction linéaire (la droite)
$y = ax^2 + bx + c$	pour la fonction du second degré (la parabole)
$y = k * \sin(ax + b)$	pour les fonctions trigonométriques

Quant aux propriétés mathématiques, nous nous sommes limités à celles définies dans la partie mathématique de l'étude des fonctionnalités (voir point 3.2.1).

Pour la droite :	son coefficient angulaire l'intersection avec les axes X et Y
Pour la parabole :	l'intersection avec les axes X et Y le sommet de la courbe l'axe de symétrie
Pour la courbe trigonométrique :	nous nous sommes limités à son dessin

7.1.2 Au niveau graphique

Cette partie, qui consiste à permettre à l'élève d'aller manipuler le dessin graphique afin de voir les répercussions sur les paramètres et les propriétés mathématiques, nous avons dû la laisser de côté.

De l'application, nous pouvons observer le tracé du graphe correspondant à son équation et sa modification lors des manipulations de ses paramètres.

7.1.3 Au niveau interface

Nous avons défini une interface la plus simple possible, dans la mesure nous pouvons montrer le lien dynamique qui relie une fonction algébrique et son dessin graphique.

Il faut également observer que l'interface que nous avons proposée ne permet pas encore à l'élève de travailler de manière autonome. En effet, dans l'application, l'enseignant joue encore un rôle important lors de la réalisation d'un exercice de l'élève : c'est lui qui lancera le logiciel et configurera les vitesses de balayage des écrans. L'élève joue encore un rôle passif dans la mesure où il ne peut qu'affecter des valeurs aux paramètres d'une équation algébrique et voir les répercussions sur son dessin graphique et ses propriétés mathématiques.

7.1.4 Au niveau des balayages

Une gestion de la hiérarchie des actions est encore à mettre au point pour que le balayage soit opérationnel de façon unifiée dans toute l'application.

Dans l'état actuel des choses, un menu doté d'un balayage ne fait que parcourir les items qui le composent sans pouvoir encore réagir aux messages de l'élève pour confirmer son choix (les messages que l'élève envoie au système correspondraient au clic du bouton droit de la souris, ce bouton jouant le rôle de contact de l'élève).

7.2 Performance du programme

Il est clair que dans le programme que nous proposons, beaucoup reste encore à faire pour que celui-ci soit entièrement opérationnel et être utilisé par l'élève de manière autonome.

Lors du test de l'application, un arrêt du programme pourrait survenir en raison de la non-gestion des horloges qui définissent le temps de balayage d'un élément d'un menu.

Une certaine lenteur lors de l'exécution de l'application a également été constatée. Ceci est dû au fait que la programmation orientée objet n'est pas encore très bien maîtrisée.

De même, lors de l'affichage dans la fenêtre des graphes et la fenêtre des calculs, une mauvaise gestion des handles peut entraîner la superposition de deux graphes ou celle de deux textes.

L'environnement de sauvegarde du travail sur disque et la configuration des balayages restent encore à mettre au point pour permettre à l'utilisateur de garder une trace de son exercice sur disque ou disquette.

7.3 Les extensions

Vu le caractère didactique du logiciel, nous pourrions également envisager son utilisation par un élève normal. C'est la raison pour laquelle nous pensons que plusieurs extensions du logiciel sont intéressantes à proposer.

7.3.1 Au niveau mathématique

En partant des formes d'équations générales :

$y = ax+b$
$y = ax^2+bx+c$
$y = k*\sin(ax+b))$

proposer d'autres formes de représentations, par exemple :

une droite passant par un point donné (x_0, y_0)
une droite passant par deux points donnés (x_1, y_1) et (x_2, y_2)
une droite ayant un certain angle avec l'axe des X, etc.

Pour plus de détail, voir l'annexe B.

En plus d'aller sélectionner l'énoncé d'un exercice dans un menu, une autre proposition serait de laisser à l'utilisateur d'aller directement éditer son équation dans la fenêtre d'édition des énoncés. Donc envisager de passer d'un type d'exercice de nature dirigée vers un type d'exercice de plus en plus interactif en incitant une plus grande participation de l'élève.

7.3.2 Au niveau graphique

Lors de la modification sur l'écran graphique, nous pourrions envisager de laisser à l'utilisateur la possibilité d'aller éditer du texte, par exemple à côté du dessin de la droite, aller mettre son équation.

7.3.3 Au niveau interaction

Le but étant d'offrir un outil didactique à une personne handicapée moteur, nous pourrions envisager de balayer l'ensemble des fenêtres de telle manière que l'élève soit complètement autonome lors de l'exécution de son exercice.

7.4 L'aspect positif du travail

De nos jours, le terme « Enseignement Assisté par Ordinateur » (E.A.O.) nous est devenu familier, ce qui n'était pas le cas il n'y a pas si longtemps que cela. Nous pouvons trouver sur le marché divers didacticiels, les bons et les moins bons. Ces didacticiels sont adressés au grand public comme les élèves, les enseignants, les médecins, les comptables, et bien d'autres. Mais si nous voulons trouver des didacticiels destinés à des personnes handicapées et en particulier à des personnes handicapées moteur, cela devient plus rare. Ceci est dû au fait qu'un didacticiel, pour pouvoir être utilisé par un utilisateur handicapé, doit être doté d'un outil supplémentaire, celui du balayage.

C'est la raison pour laquelle, la réalisation de ce mémoire à été très intéressante car elle nous a permis d'aborder différents problèmes liés à l'enseignement par ordinateur : celui lié à l'enseignement, celui lié à la programmation et celui lié à l'utilisation. C'est la rencontre de l'informaticien avec le non-informaticien dans un domaine commun qui est l'enseignement.

Personnellement, l'élaboration du projet m'a permis de tenir les trois rôles en même temps. Pour faire l'analyse des besoins, je devais me mettre à la place de l'enseignant et celle de l'utilisateur. Pour l'implémenter, je devais me mettre dans la peau du programmeur et pour la tester, je devenais l'utilisateur.

Pendant ce travail, j'ai connu des moments d'enthousiasme mais aussi des moments difficiles. L'enthousiasme venait au moment où j'ai su que mon travail serait utile à certaines personnes. C'était aussi au moment de ma visite au Centre d'Enseignement et de Traitements Différenciés (C.E.T.D.), là j'ai pu rencontrer des personnes formidables qui se sont mises à la disposition des enfants handicapés. J'ai également eu l'occasion de faire la connaissance d'un élève du centre Paul qui avait bien voulu me montrer son univers de travail. C'était à ce moment-là que je m'étais réellement rendue compte du rôle joué par l'informatique pour ces personnes.

Passé ce moment d'enthousiasme, il a fallu se mettre au travail. Et c'est le moment qui sera qualifié de difficile car avoir un sujet de mémoire, c'est une chose mais réfléchir aux moyens pour le réaliser s'en est une autre. C'est donc le moment de recherche et de lecture puisque la programmation orientée objet a été choisie pour implémenter notre application Or de l'approche orientée objet, je n'en ai qu'une connaissance théorique. Il a donc fallu que j'apprenne à maîtriser les concepts que l'approche met en avant et un langage de programmation orientée objet, le Borland Pascal. C'est une période fastidieuse, car pour bien comprendre ces concepts, une autre vision des choses était nécessaire, celui de la modélisation du monde en objets.

Une autre difficulté du projet est la conception du logiciel car les personnes à qui est destinée notre application sont des personnes particulières, il a donc fallu penser à leur environnement de travail, celui géré par le balayage.

Les difficultés rencontrées étaient de plusieurs niveaux : il fallait garder en tête l'aspect didactique du projet et essayer de se mettre à la place des utilisateurs, penser à la façon d'implémenter les interfaces, d'une part pour l'enseignant et d'autre part pour l'élève, le tout dans la vision de la programmation orientée objet.

Pour résumer, les trois niveaux de difficultés ont été :

Le niveau pédagogique (le rôle tenu par l'enseignant)
Le niveau handicap (l'utilisateur)
Le niveau de la programmation orientée objet (le programmeur)

Ce qui serait formidable, c'est que les objectifs soient atteints et donc notre application opérationnelle pour que les enseignants et les élèves puissent l'utiliser.

Personnellement, s'il fallait refaire un choix, mon choix resterait le même car je pense que l'informatique peut apporter tellement de choses, pour ne fût-ce faciliter la vie de ces personnes en leur rendant un peu plus autonome dans leur environnement de travail.

Bibliographie

« Conception assistée des systèmes d'information »
Méthode-Modèles-Outils
F. BODART, Y. PIGNEUR
2° Edition, MASSON, 1989

Cours de **« Méthodologie de développement de logiciels »**
Notes de cours révisées par Monsieur E. DUBOIS
Cours donné en 2ème Licence et Maîtrise en Informatique,
Année académique 1991-1992

« Une description orientée objet dans objets interactifs abstraits utilisés en Interface Homme-Machine »
Benoît SACRE, Isabelle PROVOT-SACRE, Jean VANDERDONCKT
7 octobre 1992
Rapport de recherche IHM/Ergo/10

« Corpus ergonomique minimal des applications de gestion »
Jean VANDERDONCKT, 29 octobre 1992
Rapport IHM/Règles/10

« Borland Pascal 7 », *par la pratique*
Scott D. PALMER
Sybex, 1992

« Object Windows », *Guide du programmeur*
Initiation-Utilisation d'ObjectWindows-Référence
Borland, 1991-1992

« Resource Workshop », *Guide de l'utilisateur*
Borland, 1991-1992

« Turbo Debugger », *Guide de l'utilisateur*
Version 3.2, Borland, 1992

« Programmer sous Windows 3.1 », Charles PETZOLD,
Adapté par Frédéric RUTKOWSKI, Deuxième tirage 1993
Microsoft Press

« Abrégés de médecine », *Neurologie*
Jean CAMBIER, Maurice MASSON et Henri DEHEN
6° édition, revue et corrigée, MASSON 1989

« L'Informatique à l'école », *Collection « Les études par l'exemple »*
PF. BREDECHE, H. COLIN
« Les applications dans les programmes de : mathématiques, sciences physiques et Biologie / Lycéens-étudiants, formation permanente des enseignants »
Editions ROUDIL

« Définitions techniques et formules mathématiques »
J. ALLARD, N. BERCKMANS, L. FRANCOIS
Institut Saint-Louis, cours scientifique spécial

Mémoire : **« Didacticiel de géométrie plane par simulation des outils classiques (équerre, rapporteur, etc.) »**
à l'usage des enfants handicapés moteurs : module auteur et élève
Nicolas MAES, Octobre 1994.

Table des matières

Table des figures

Annexe A : Infirmité Motrice Cérébrale

Infirmité Motrice Cérébrale

1. Les sources

Abrégés de médecine *« Neurologie »*
Jean Cambier, Maurice Masson et Henri Dehen
6° édition, revue et corrigée, MASSON 1989.

2. Définitions

Page 512 et suivantes *:*

Malformations du système nerveux et encéphalopatiques infantiles

« Encéphalopathies périnatales et post-natales
Problème de l'infirmité motrice cérébrale »

2.1 Nature des lésions

« Le risque d'altérations du système nerveux au cours du travail de l'accouchement est important. Des lésions peuvent se produire alors même que l'accouchement paraît s'être déroulé normalement; ceci est tout spécialement vrai pour des prématurés mais les lésions sont d'autant plus fréquentes que l'accouchement a été prolongé, la présentation inhabituelle, qu'il a fallu recourir au forceps, que l'enfant n'a pas crié spontanément. »

« Ces lésions peuvent être de nature traumatique, hémorragie méningée, hémorragie cérébrale, engagement temporel, mais on s'accorde à reconnaître la responsabilité majeure de l'anoxie, anoxie avant la naissance lors d'un travail qui se prolonge, après la naissance lorsque la respiration ne s'établit pas normalement. Tout effort visant à améliorer la qualité des conduites obstétricales et de la surveillance néo-natale entraîne une diminution de l'incidence des encéphalopathies enfantiles. Néanmoins à côté des lésions relevant du traumatisme de la naissance, il faut rappeler le caractère pathogène de certains désordres métaboliques post-nataux tels que l'ictère nucléaire, lié à l'incompatibilité Rh, dont on sait maintenant prévenir le développement, mais aussi la sensibilité du système nerveux, à certaines affections acquises durant la première enfance : méningites purulentes, thromboses veineuses, complications encéphaliques des maladies enfectieuses... »

« Une place à part revient au syndrome hémiconvulsions-hyperthermie dont l'étiologie reste souvent mystérieuse mais dont les conséquences apparaissent redoutables. »

2.2 Sémiologie clinique

« Sous le nom d'infirmité motrice cérébrale, on désigne les conséquences neurologiques de ces lésions frappant un système nerveux en plein développement. L'infirmité motrice revêt des aspects cliniques très divers qui ne prennent leur formule définitive que de façon retardée, à mesure des étapes de la maturation psychomotrice. Voici les schémas de quelques types fondamentaux[1] : »

2.2.1 L'infirmité motrice cérébrale

« L'infirmité motrice cérébrale, dans ses différents aspects peut être remarquablement dissociée de toute atteinte des fonctions cognitives et ne pas s'accompagner d'un déficit intellectuel notable. Il est loin d'en être toujours ainsi : le rôle des lésions de l'hémisphère dominant et aussi de celles de l'himisphère mineure dans le déterminisme des difficultés d'acquisition du langage ou du développement de la pensée spatiale est évident, notamment dans l'hémiplégie cérébrale infantile. D'un autre côté, bon nombre d'infirmes moteurs cérébraux ont un quotient intellectuel qui, même évalué par des méthodes appropriées, est nettement inférieur à la normale. Enfin l'épilepsie est fréquente. »

2.3 Traitement

« La conduite thérapeutique doit tenir compte de l'ensemble de ces facteurs. Elle englobe l'éducation de ces malades et nécessite souvent le recours à des internats spécialisés. Aux difficultés spécifiques d'ordre neurologique et intellectuel s'ajoutent des perturbations caractérielles résultant de la difficulté pour l'enfant d'assumer son infirmité dans un milieu fait d'individus normaux. »

[1] Je n'ai repris que le cas qui nous intéresse c'est-à-dire l'infirmité motrice cérébrale

Annexe B : Partie mathématique

Partie mathématique

Au niveau mathématique, nous allons faire, pour chaque famille de fonctions, leur étude et ce en tenant compte des besoins des étudiants et du contenu de leur cours de mathématique.

1. Le point

Les possibilités relatives à l'étude du point sont fort simples : il s'agira de pouvoir visualiser un point dans un repère cartésien tout en observant les valeurs de ses coordonnées.

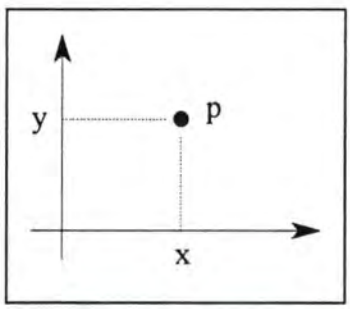


Figure B.1 : Un point et ses coordonnées cartésiennes

2. La droite

Pour l'étude de la droite, voyons tout d'abord les différentes façons de définir une droite.

2.1 Une droite passant par l'origine

La forme générale est $y = ax$ où a est le coefficient angulaire
$y = tg\alpha\ x$ où l'angle avec l'axe des X est α

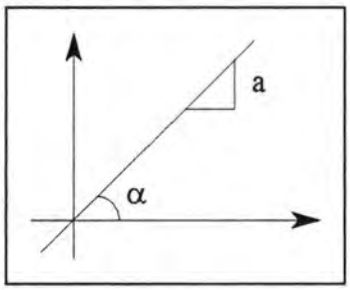


Figure B.2 : Une droite passant par l'origine

Parmi les droites passant par l'origine, nous avons les deux droites bissectrices par rapport au repère cartésien :

La 1° bissectrice a pour équation $y = x$
La 2° bissectrice a pour équation $y = -x$

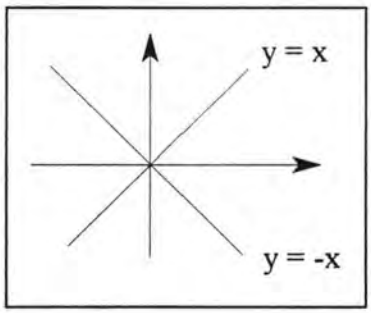


Figure B.3 : Les deux droites bissectrices par rapport au repère cartésien

2.2 *Une droite passant par un point*

La forme générale est $y - y_0 = a (x - x_0)$

$y - y_0 = tg\alpha (x - x_0)$

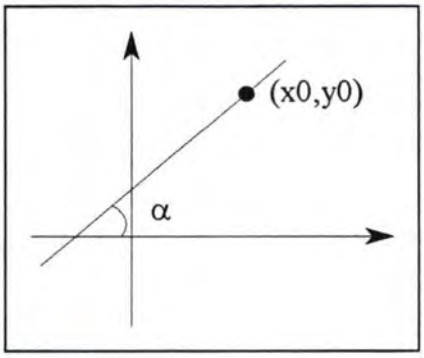


Figure B.4 : Une droite passant par un point

Parmi les droites passant par un point, nous avons les droites parallèles à l'axe des X. L'équation d'une droite parallèle par rapport à l'axe des X est y = a.

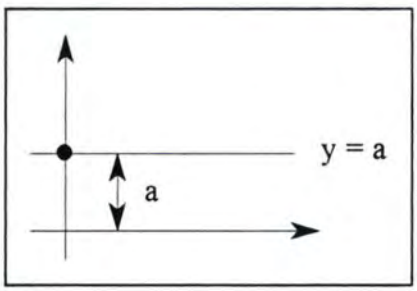


Figure B.5 : Une droite parallèle à l'axe des X

2.3 Fonction linéaire quelconque

La forme générale est y = ax + b

L'intersection avec les axes : si x = 0 alors y = b
si y = 0 alors x = - b / a

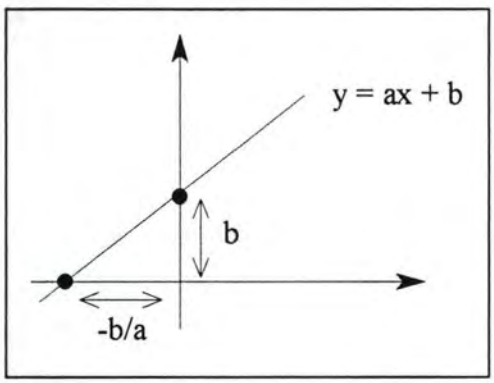


Figure B.6 : Une droite quelconque

Parmi les droites, nous avons les droites passant par deux points dont la forme générale est :

$$y - y_0 = \frac{y_1 - y_0}{x_1 - x_0}(x - x_0)$$

dont $\frac{y_1 - y_0}{x_1 - x_0}$ représente le coefficient angulaire de la droite.

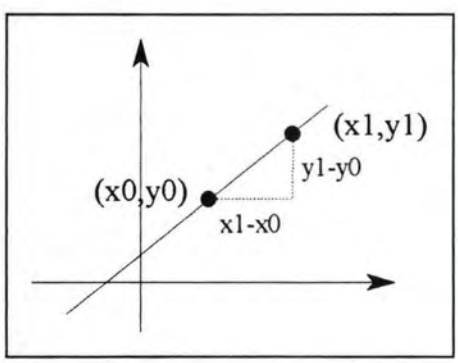


Figure B.7 : Une droite passant par deux points

3. La parabole

Pour l'étude de la parabole, nous envisageons les aspects suivants :

3.1 Forme de présentation

L'équation sous la forme $y = ax^2$

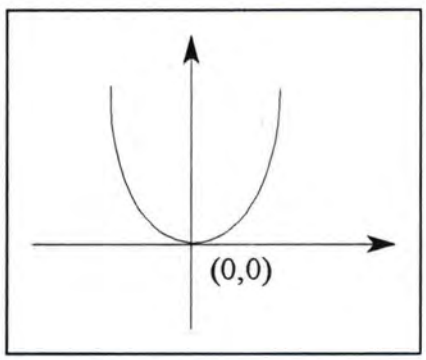


Figure B.8 : Une parabole sous la forme $y= ax^2$

L'équation générale sous les formes suivantes :

* $y = ax^2 + bx + c$

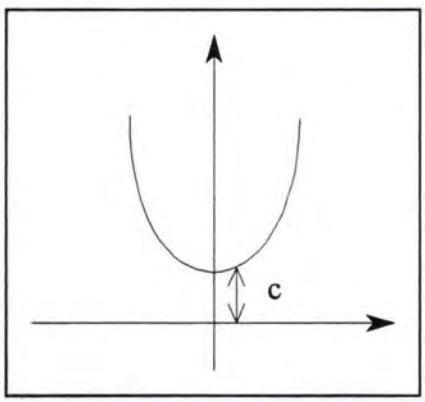


Figure B.9 : Une parabole sous la forme $y= ax^2+bx+c$ quand b est nul

3.2 Intersection avec les axes

* Pour la forme $y = ax^2$, la parabole passe par l'origine (0,0).

* Pour la forme $y = ax^2 + bx + c$, l'intersection avec :

l'axe des X :

si $y = 0$ alors $ax^2 + bx + c = 0$ $\quad (a \neq 0)$

si $\rho = b^2 - 4ac$: $\quad \rho > 0 : x = \dfrac{-b \pm \sqrt{\rho}}{2a}$

$$\rho = 0 : x = \frac{-b}{2a}$$

l'axe des Y : si $x = 0$ alors $y = c$

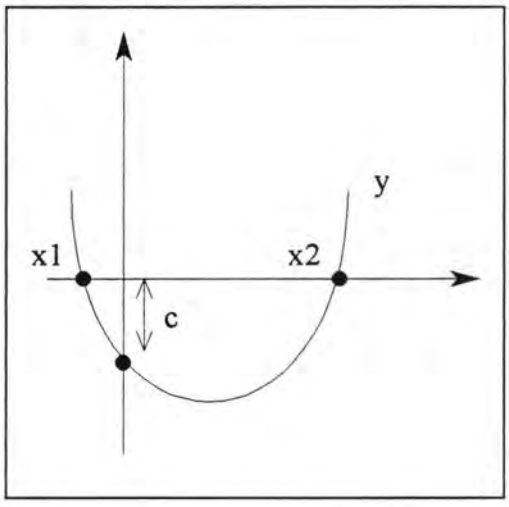


Figure B.10 : Les intersections d'une parabole avec les axes

3.3 Signe de y selon la valeur de a

En prenant les formes $y = ax^2 + bx + c$:

si $\rho > 0$: y a le signe de a à l'extérieur des racines
y = 0 pour les racines
y a le signe contraire de a à l'intérieur des racines

si $\rho = 0$: y a toujours le signe de a, sauf pour les racines où il s'annule; ce cas est valable aussi pour la forme $y = ax^2$

3.4 Extremums

En prenant la forme $y = ax^2 + bx + c$:

$$x_m = \frac{-b}{2a}$$

$$y_m = -\frac{b^2 - 4ac}{4a}$$

sommet = minimum si a > 0
maximum si a < 0

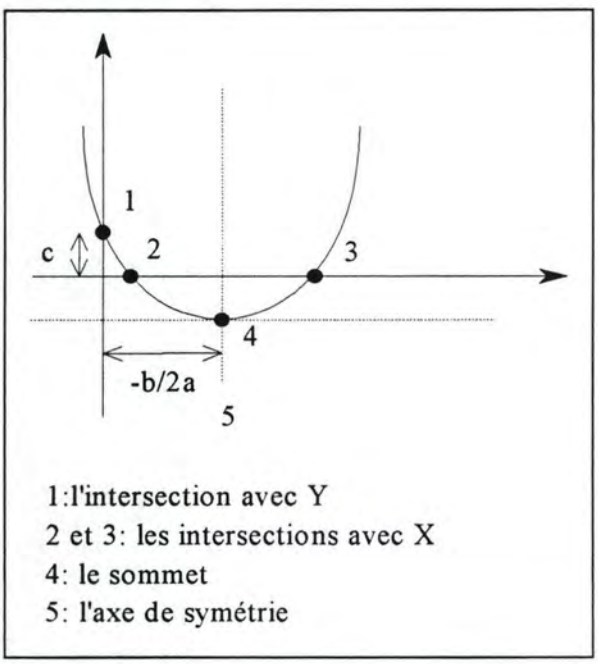


Figure B.11 : Les intersections, le sommet et l'axe de symétrie d'une parabole

3.5 Tangente en un point (x_0 , y_0) de la parabole

L'équation de la tangente est $y = \lambda (x - x_0) + y_0$

où $\lambda = ax_0 + b$ représente la dérivée première de $y = ax^2 + bx + c$

3.6 Axe de symétrie

$x = \frac{-b}{2a}$ si on prend la forme $y = ax^2 + bx + c$

4. Les fonctions trigonométriques

Pour l'étude des fonctions trigonométriques, nous envisageons les aspects suivants:

Rappelons les catégories des fonctions que nous allons étudier :

- k * sin(ax+b)
- k * cos(ax+b)
- k * tg(ax+b)

4.1 La période

$\frac{360}{|a|}$ *ou* $\frac{2\pi}{|a|}$ pour k * sin(ax+b) et k * cos(ax+b)

$\frac{180}{|a|}$ *ou* $\frac{\pi}{|a|}$ pour k * tg(ax+b)

4.2 La parité

Paire pour le cosinus si b = nπ, pour le sinus si b = (2n+1)π/2.
Impaire pour le sinus si b = nπ, pour le cosinus si b = (n2+1)π/2.

4.3 La symétrie

Par rapport aux verticales x = nπ-b pour le cosinus.
Par rapport aux verticales x = (2n+1)π/2-b pour le sinus.

4.4 Les zéros

Pour k * sin(ax+b) :

sin(ax+b) = 0

→ ax+b = t * 180

→ ax = t * 180 - b

→ x = (t * 180 - b) / a

Idem pour k * tg(ax+b)

Pour k * cos(ax+b) :

cos(ax+b) = 0

→ ax+b = 90 + t * 180

→ x = (90 + t * 180 - b) / a

4.5 Les extrema

Pour k * sin(ax+b) : x = (90 + t * 180 - b) / a
Pour k * cos(ax+b) : x = (t * 180 - b) / a
Pas d'extrema pour les tg.

Annexe C : Partie graphique

Partie graphique

1. Le point

Au niveau graphique, les seules opérations sur les points sont leurs déplacements sur le repère cartésien et l'évolution de leurs coordonnées en fonction de leurs déplacements.

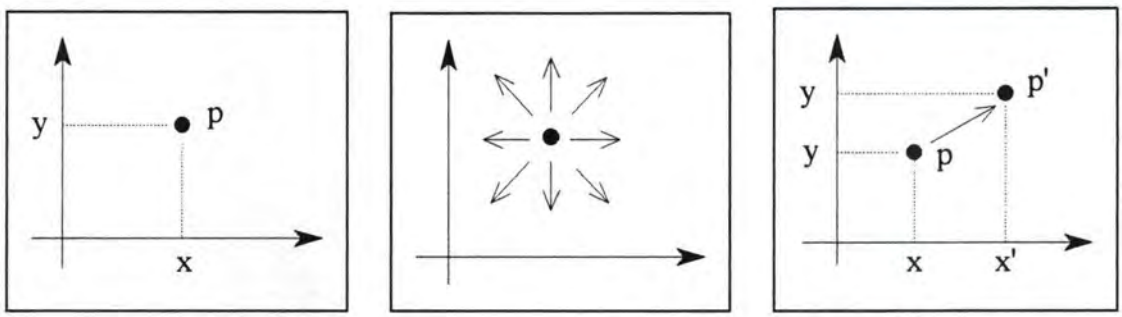


Figure C.1 : Le déplacement d'un point dans le repère cartésien

2. La droite

Les opérations sur les droites au niveau graphique sont les suivantes :

- Bouger un point quelconque de la droite en gardant un point fixe

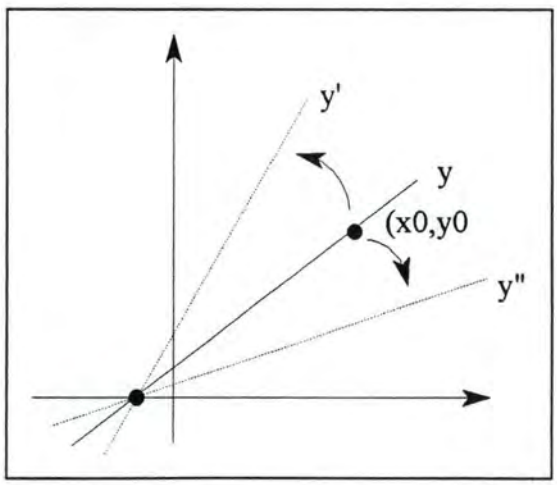


Figure C.2 : Le déplacement d'une droite passant par un point

- Bouger la droite entière par une translation

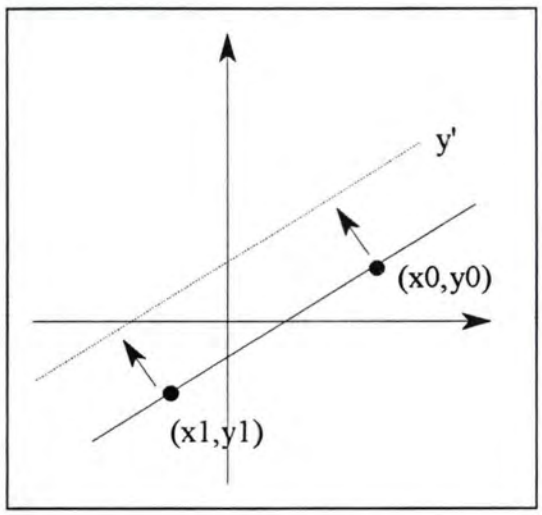


Figure C.3 : Le déplacement d'une droite passant par deux points

Une extension possible du logiciel serait de laisser à l'utilisateur la possibilité de nommer les droites ou les parties de droite, éventuellement des segments de droites.

3. La parabole

Les opérations sur les paraboles au niveau dessin sont les suivantes :

- Bouger uniquement le sommet vers le haut ou vers le bas en gardant fixe les points à l'intersection des axes.

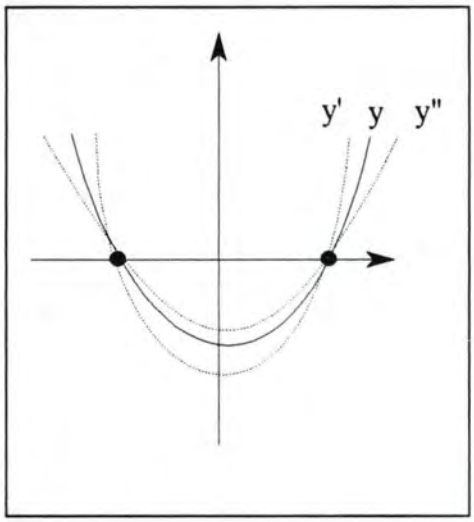


Figure C.4 : Le déplacement d'une parabole en bougeant le sommet

- Faire translater la parabole vers le haut ou vers le bas, vers la gauche ou vers la droite.

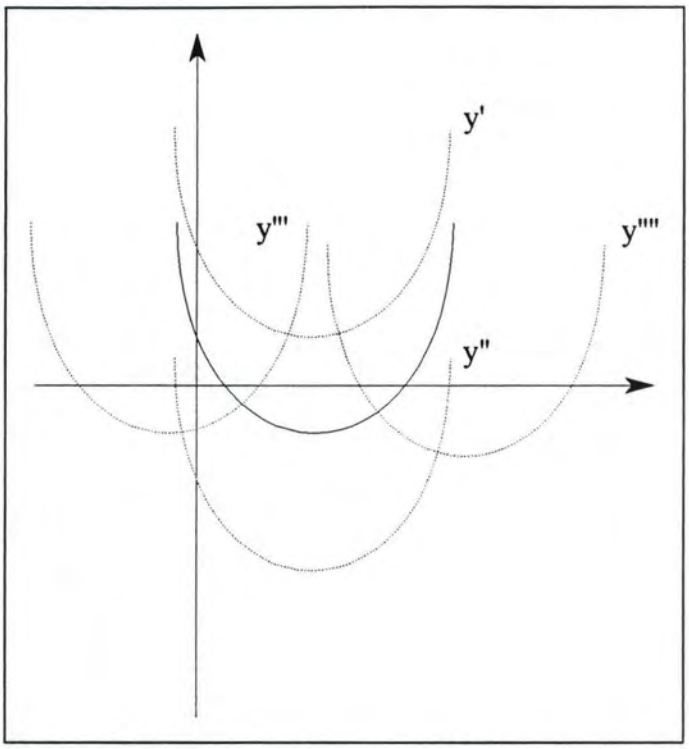


Figure C.5 : Le déplacement d'une parabole dans les 4 directions

- Faire afficher son axe de symétrie
- Faire afficher la tangente au sommet
- Faire afficher la tangente en un point de la courbe

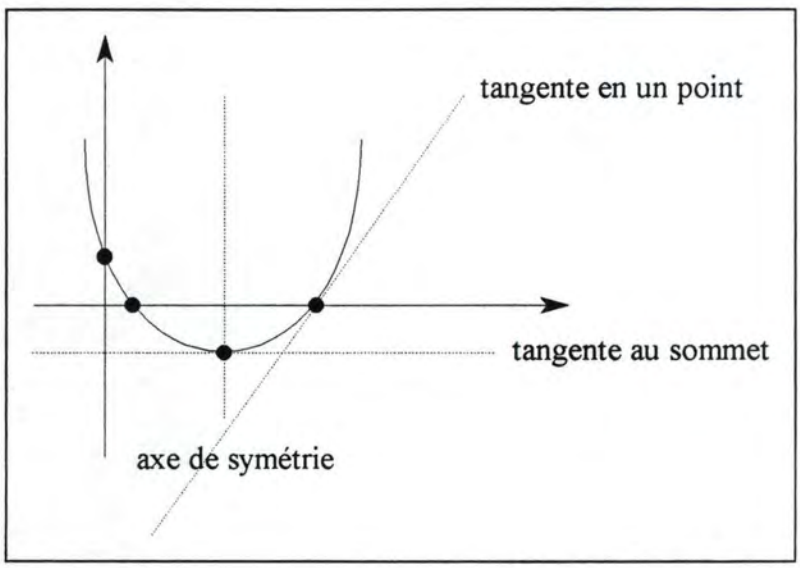


Figure C.6 : L'axe de symétrie et les tangentes d'une parabole quelconque

Une extension possible du logiciel serait de laisser à l'utilisateur la possibilité de nommer les courbes ou les parties de courbe.

4. Les fonctions trigonométriques

Quant aux opérations que les utilisateurs pourront effectuer sur les fonctions trigonométriques, nous avons envisagé les suivantes :

- Faire afficher les tangentes verticales pour la fonction tg

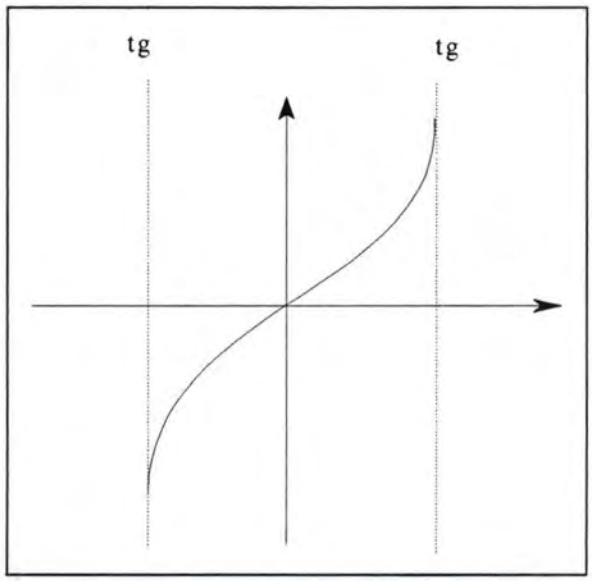


Figure C.7 : les tangentes verticales de la fonction tg

- Faire afficher la tangente d'un sommet donné

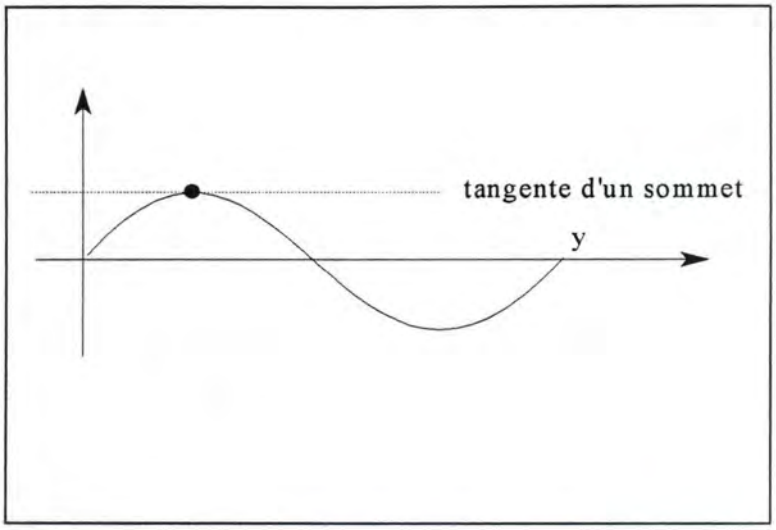


Figure C.8 : La tangente d'un sommet d'une fonction trigonométrique

- Déplacer la courbe par une translation par rapport à l'axe des Y

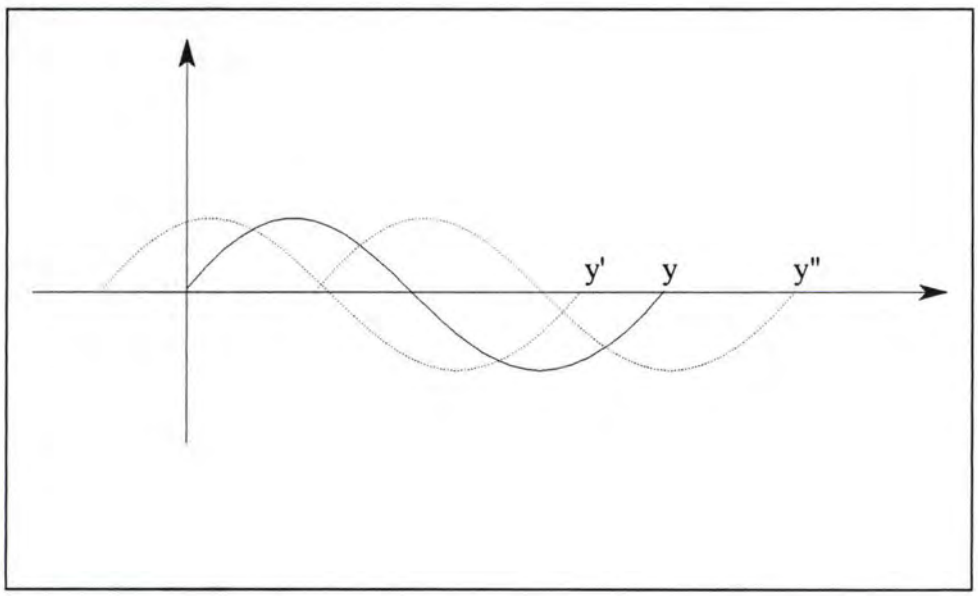


Figure C.9 : Le déplacement d'une fonction trigonométrique par rapport à Y

Ici aussi, il serait intéressant d'envisager la possibilité de nommer les courbes.

Listing du code source (sans commentaire)

```
{##############################################################################}
{#                    Programme Principal                                     #}
{##############################################################################}

PROGRAM ProgrammePrincipal; { le corps du programme principal }

USES
  WINTYPES, WINPROCS, OWINDOWS,
  FENPRINC, NOUV, OUVRIR, SAUVER, TERMINER,
  PARAM, GRAPHE, AIDE, ZOOMPLUS, ZOMMOINS;

TYPE
  PProgPrincApplic = ^TProgPrincApplic;
  TProgPrincApplic = OBJECT (TApplication)
    PROCEDURE InitMainWindow; VIRTUAL;
  END;

{==================== PROCEDURE de TProgPrincApplic ========================}

{------------------ PROCEDURE TProgPrincApplic.InitMainWindow -----------}
  PROCEDURE TProgPrincApplic.InitMainWindow;
    BEGIN
      MainWindow := New(PFenPrincApplic, Init(NIL,'Etude de fonctions'));
    END;

{******************************************************************************}
{***************************** Corps du programme principal *******************}
{******************************************************************************}

VAR
  ProgPrincApplic:TProgPrincApplic;

BEGIN
  ProgPrincApplic.Init('Programme Principal');
  ProgPrincApplic.Run;
  ProgPrincApplic.Done;
END.
```

```
{#########################################################################}
{#                    Unité FenPrinc                                     #}
{#########################################################################}

UNIT FenPrinc; { Unité de la fenêtre principale de l'application }

{$R menu.res}
{$R apropos.res}

{*************************************************************************}
{**************************** Interface **********************************}
{*************************************************************************}

INTERFACE

USES
  WINTYPES, WINPROCS, OWINDOWS, OBJECTS,
  WINDOS, STRINGS, OSTDDLGS, ODIALOGS, WINCRT,
  GESTFICH, FGRAPHE1, ENONEQ, PALOUTIL, FPALOUT, SOURIS,
  DEQUAT, PARAMET, FEXPLIC, CALPROP, VARGLOB, VARPOINT, BALAYAGE;

{$I menu.inc}
{$I apropos.inc}

TYPE
  PFenPrincApplic = ^TFenPrincApplic;
  TFenPrincApplic = OBJECT (TWindow)
    PFenetre : PWindow;
    PLeDessinFonction : PDessinFonctionCollect;
    PLaFenGraphe : PFenGraphe;
    PLaFenSouris : PFenSouris;
    PLaFenEnonEq : PFenEnonEq;
    PLaFenPaletteOutil : PFenPaletteOutil;
    PLaFenParametres : PFenParametres;
    PLaGestionFichier : PGestionFichier;
    PLaFenCommentaire : PFenCommentaire;
    PLaFenCalculsProp : PFenCalculsProp;
    PLesVariablesPointeurs : PVariablesPointeurs;
    Aide : Boolean;      { true to display help in WMCommand }
    AideNomFich : PChar;  { online help file name }
    CONSTRUCTOR Init(AParent:PWindowsObject;ATitle: PChar);
    DESTRUCTOR Done; VIRTUAL;
    FUNCTION CanClose : Boolean; VIRTUAL;
    PROCEDURE FonctionNonImplementee;
    PROCEDURE EntrerNomFich(NouvNomFich:PChar);
    PROCEDURE ChargerFich(NouvNomFich:PChar);
    PROCEDURE SauverFich(gNomFich:PChar);
    PROCEDURE Nouveau(VAR Msg: TMessage);
      VIRTUAL cm_First + cm_Nouveau;
    PROCEDURE Ouvrir(VAR Msg: TMessage);
      VIRTUAL cm_First + cm_Ouvrir;
    PROCEDURE Sauver(VAR Msg: TMessage);
      VIRTUAL cm_First + cm_Sauver;
    PROCEDURE Sauver_sous(VAR Msg: TMessage);
      VIRTUAL cm_First + cm_Sauver_sous;
    PROCEDURE Quitter(VAR Msg: TMessage);
      VIRTUAL cm_First + cm_Quitter;
    PROCEDURE Personnel(VAR Msg: TMessage);
      VIRTUAL cm_First + cm_Personnel;
```

```
  PROCEDURE SauverBal(VAR Msg: TMessage);
    VIRTUAL cm_First + cm_SauverBal;
  PROCEDURE Vue_Parametres(VAR Msg: TMessage);
    VIRTUAL cm_First + cm_Vue_Parametres;
  PROCEDURE Vue_Enonce(VAR Msg: TMessage);
    VIRTUAL cm_First + cm_Vue_Enonce;
  PROCEDURE Vue_Souris(VAR Msg: TMessage);
    VIRTUAL cm_First + cm_Vue_Souris;
  PROCEDURE Vue_Palette_Action(VAR Msg: TMessage);
    VIRTUAL cm_First + cm_Vue_Palette_Action;
  PROCEDURE Vue_Calculs_Prop(VAR Msg: TMessage);
    VIRTUAL cm_First + cm_Vue_Calculs_Prop;
  PROCEDURE CMAideIndex(VAR Msg: TMessage);
    VIRTUAL cm_First + cm_Index;
  PROCEDURE CMAideUsing(VAR Msg: TMessage);
    VIRTUAL cm_First + cm_Aide_Using;
  PROCEDURE Presentation(VAR Msg: TMessage);
    VIRTUAL cm_First + cm_Presentation;
  PROCEDURE Mise_en_route(VAR Msg: TMessage);
    VIRTUAL cm_First + cm_Mise_en_route;
  PROCEDURE Fonctionnalites(VAR Msg: TMessage);
    VIRTUAL cm_First + cm_Fonctionnalites;
  PROCEDURE Termes_math(VAR Msg: TMessage);
    VIRTUAL cm_First + cm_Termes_math;
  PROCEDURE CMAideAProposDe(VAR Msg: TMessage);
    VIRTUAL cm_First + cm_A_Propos_de;
  PROCEDURE WMDestroy(VAR Msg: TMessage);
    VIRTUAL wm_First + wm_Destroy;
  PROCEDURE WMLButtonDown (VAR Msg: TMessage);
    VIRTUAL wm_First + wm_LButtonDown;
  PROCEDURE WMRButtonDown (VAR Msg: TMessage);
    VIRTUAL wm_First + wm_RButtonDown;
  PROCEDURE WMNCButtonDblClk(VAR Msg: TMessage);
    VIRTUAL wm_First + wm_NCLButtonDblClk;
 END;

{*************************************************************************}
{***************************** Implementation ****************************}
{*************************************************************************}

IMPLEMENTATION

{==================== Procedures de TFenPrincApplic ======================}

{------------------ PROCEDURE TFenPrincApplic.FonctionNonImplementee ------------------}
 PROCEDURE TFenPrincApplic.FonctionNonImplementee;
  BEGIN
    MessageBox(HWindow,
      'Cette fonctionnalité n''a pas encore été implémentée',
      'Message d''information',
      mb_Ok OR mb_IconExclamation);
  END;
```

```
{---------- Constructeur TFenPrincApplic.Init ----------}
 CONSTRUCTOR TFenPrincApplic.Init(AParent:PWindowsObject;ATitle:PChar);
  CONST
   ExeNameMaxSize = 128;
  VAR
   FileNameLen : integer;
   FileName : ARRAY [0..ExeNameMaxSize + 1 ] OF Char;
   I : integer;
  BEGIN
   INHERITED Init(AParent,ATitle);
   attr.menu := LoadMenu(HInstance,'id_menu');
   WITH attr DO
    BEGIN
     x:=0;
     y:=0;
     w:=GetSystemMetrics(sm_cxscreen);
     h:=GetSystemMetrics(sm_cyscreen);
     style:=style OR ws_ClipChildren
              OR ws_Visible OR ws_MaximizeBox;
    END;

   new(PLeDessinFonction,init(@self));
   new(PLaFenSouris,init(@self,'Souris'));
   new(PLaFenGraphe,init(@self,'',PLaFenSouris,PLeDessinFonction));
   new(PLaGestionFichier,init(@self,''));
   new(PLaFenEnonEq,init(@self,'Enoncé de l''exercice'));
   new(PLaFenPaletteOutil,init(@self,'Palette Action',Pal_PosDefPalette));
   new(PLaFenCalculsProp,init(@self,'Les propriétés mathématiques'));
   new(PLaFenParametres,init(@self,'Les Paramètres'));
   new(PLaFenCommentaire,init(@self,''));
   new(PLesVariablesPointeurs,init(@self,'',PFenetre,PLaGestionFichier,
     PLaFenCommentaire,PLaFenGraphe,PLaFenCalculsProp));

   {- construct AideNomFich from Module Name -}
   FileNameLen := GetModuleFileName(HInstance, FileName, ExeNameMaxSize);
   I := FileNameLen - 1;
   WHILE (I <> 0) AND ((FileName[I] <> '\') AND (FileName[I] <> ':')) DO
    Dec(I);
   Inc(I);
   IF I + 13 <= ExeNameMaxSize
    THEN StrCopy(@FileName[I], 'fonction.HLp')
    ELSE StrCopy(@FileName[I], 'Aide');
   AideNomFich := StrNew(FileName);
   Aide := false;
   EstNouvFich := true;
   EntrerNomFich('sans nom');
   SetWindowText(HWindow,Titre);
   EstModifie := False;
  END;
```

```
{---------- FONCTION TFenPrincApplic.CanClose ----------}
 FUNCTION TFenPrincApplic.CanClose : Boolean;
  VAR
   Msg : TMessage;
  BEGIN
   IF EstModifie THEN
    BEGIN
     reponse := MessageBox(HWindow,
           'Le dessin a été modifié Voulez-vous sauver ?',
           'Message pour quitter le programme',
            mb_YesNo OR mb_IconQuestion);
     IF reponse = id_Yes
      THEN
       BEGIN
        Sauver_Sous(msg);
        CanClose := true;
       END;
    END;
   CanClose := True;
   EstModifie := False;
  END;

{---------- PROCEDURE TFenPrincApplic.EntrerNomFich ----------}
 PROCEDURE TFenPrincApplic.EntrerNomFich(NouvNomFich:PChar);
  VAR
   Titre : ARRAY [0..fsPathName + 19] OF Char;
  BEGIN
   StrCopy(NomFich,NouvNomFich);
   StrCopy(StrECopy(StrECopy(Titre,'Etude de fonctions: ['),
        NomFich), ']');
   SetWindowText(HWindow,Titre);
  END;

{---------- PROCEDURE TFenPrincApplic.ChargerFich ----------}
 PROCEDURE TFenPrincApplic.ChargerFich (NouvNomFich:PChar);
  VAR
   aFichier:TDosStream;
  BEGIN
   aFichier.Init(NouvNomFich,stOpen);
   CASE aFichier.status OF
    stOK:
     BEGIN
      EstNouvFich:=false;
      InvalidateRect(HWindow,NIL,true);
      EntrerNomFich(NouvNomFich);
     END;
    stInitError:
     MessageBox(HWindow,'Nom de fichier invalide',
                'Stream error',mb_IconExclamation OR mb_OK);
    ELSE
     {DisplayStreamError(aFichier);}
   END;
   aFichier.done;
  END;
```

```
{---------- PROCEDURE TFenPrincApplic.SauverFich ----------}
 PROCEDURE TFenPrincApplic.SauverFich(gNomFich:PChar);
  VAR
   aFichier:TDosStream;
  BEGIN
   aFichier.Init(gNomFich,stCreate);
   aFichier.Done;
   EstNouvFich:=false;
   EntrerNomFich(gNomFich);
  END;

{---------- PROCEDURE TFenPrincApplic.Nouveau ----------}
 PROCEDURE TFenPrincApplic.Nouveau (VAR Msg:TMessage);
  BEGIN
   IF CanClose THEN
    BEGIN
     InvalidateRect(HWindow,NIL,true);
     EstNouvFich := true;
     EntrerNomFich('sans nom');
    END;
   Application^.ExecDialog(New(PDiaFamFonc,
    Init(@Self, PChar(id_dial_equations))));
  END;

{---------- PROCEDURE TFenPrincApplic.Ouvrir ----------}
 PROCEDURE TFenPrincApplic.Ouvrir (VAR Msg:TMessage);
  VAR
   NouvNomFich : ARRAY [0..fsPathNAme] OF Char;
  BEGIN
   IF CanClose THEN
    IF Application^.ExecDialog(New(PFileDialog,
     Init(@self,MakeIntResource(sd_FileOpen),
     StrCopy(NouvNomFich,'*.fnc')))) = id_Ok THEN
     ChargerFich(NouvNomFich);
  END;

{---------- PROCEDURE TFenPrincApplic.Sauver ----------}
 PROCEDURE TFenPrincApplic.Sauver (VAR Msg:TMessage);
  BEGIN
   IF EstNouvFich THEN Sauver_sous(Msg)
          ELSE
           BEGIN
            SauverFich(NomFich);
           END;
  END;
```

```
{---------- PROCEDURE TFenPrincApplic.Sauver_Sous ----------}
 PROCEDURE TFenPrincApplic.Sauver_Sous (VAR Msg:TMessage);
  VAR
   paFichDialogue:PFileDialog;
   NouvNomFich : ARRAY [0..fsPathName] OF Char;
  BEGIN
   IF EstNouvFich THEN StrCopy(NouvNomFich,'*.fnc')
          ELSE StrCopy(NouvNomFich,NomFich);
   IF Application^.ExecDialog(New(PFileDialog,
      Init(@Self,MakeIntResource(sd_FileSave),NouvNomFich))) = id_Ok
     THEN
      BEGIN
       SauverFich(NouvNomFich);
      END;
  END;

{---------- PROCEDURE TFenPrincApplic.Terminer ----------}
 PROCEDURE TFenPrincApplic.Quitter(VAR Msg: TMessage);
  BEGIN
   CloseWindow;
  END;

{---------- PROCEDURE TFenPrincApplic.Personnel ----------}
 PROCEDURE TFenPrincApplic.Personnel (VAR Msg:TMessage);
  BEGIN
   Application^.ExecDialog(New(PDiabalayage,
    Init(@Self, PChar(id_dial_fen_balayage))));
  END;

{---------- PROCEDURE TFenPrincApplic.SauverBal ----------}
 PROCEDURE TFenPrincApplic.SauverBal (VAR Msg:TMessage);
  BEGIN
   IF EstNouvFich
    THEN
     BEGIN
      Sauver_sous(Msg);
     END
    ELSE
     BEGIN
      SauverFich(NomFich);
     END;
  END;

{---------- PROCEDURE TFenPrincApplic.Vue_Parametres ----------}
 PROCEDURE TFenPrincApplic.Vue_Parametres (VAR Msg: TMessage);
  BEGIN
   IF (mf_Checked AND GetMenuState(attr.Menu,cm_Vue_Parametres,mf_ByCommand)) <> 0
    THEN
     BEGIN
      CheckMenuItem(attr.Menu,cm_Vue_Parametres,mf_ByCommand OR mf_Unchecked);
      PLaFenParametres^.Show(sw_Hide);
     END
    ELSE
     BEGIN
      CheckMenuItem(attr.Menu,cm_Vue_Parametres,mf_ByCommand OR mf_Checked);
      PLaFenParametres^.Show(sw_Show);
     END;
  END;
```

```
{------------------ PROCEDURE TFenPrincApplic.Vue_Enonce ----------------}
 PROCEDURE TFenPrincApplic.Vue_Enonce (VAR Msg: TMessage);
  BEGIN
   IF (mf_Checked AND GetMenuState(attr.Menu,cm_Vue_Enonce,mf_ByCommand)) <> 0
    THEN
     BEGIN
      CheckMenuItem(attr.Menu,cm_Vue_Enonce,mf_ByCommand OR mf_Unchecked);
      PLaFenEnonEq^.Show(sw_Hide);
     END
    ELSE
     BEGIN
      CheckMenuItem(attr.Menu,cm_Vue_Enonce,mf_ByCommand OR mf_Checked);
      PLaFenEnonEq^.Show(sw_Show);
     END;
  END;

{------------------ PROCEDURE TFenPrincApplic.Vue_Souris ----------------}
 PROCEDURE TFenPrincApplic.Vue_Souris(VAR Msg: TMessage);
  BEGIN
   IF (mf_Checked AND GetMenuState(attr.Menu,cm_Vue_Souris,mf_ByCommand)) <> 0
    THEN
     BEGIN
      CheckMenuItem(attr.Menu,cm_Vue_Souris,mf_ByCommand OR mf_Unchecked);
      PLaFenSouris^.Show(sw_Hide);
     END
    ELSE
     BEGIN
      CheckMenuItem(attr.Menu,cm_Vue_Souris,mf_ByCommand OR mf_Checked);
      PLaFenSouris^.Show(sw_Show);
     END;
  END;

{------------------ PROCEDURE TFenPrincApplic.Vue_Menu_Principal --------}
 PROCEDURE TFenPrincApplic.Vue_Palette_Action(VAR Msg: TMessage);
  BEGIN
   IF (mf_Checked AND GetMenuState(attr.Menu,cm_Vue_Palette_Action,mf_ByCommand)) <> 0
    THEN
     BEGIN
      CheckMenuItem(attr.Menu,cm_Vue_Palette_Action,mf_ByCommand OR mf_Unchecked);
      PLaFenPaletteOutil^.Show(sw_Hide);
     END
    ELSE
     BEGIN
      CheckMenuItem(attr.Menu,cm_Vue_Palette_Action,mf_ByCommand OR mf_Checked);
      PLaFenPaletteOutil^.Show(sw_Show);
     END;
  END;
```

```
{---------- PROCEDURE TFenPrincApplic.Vue_Calculs_Prop ----------}
 PROCEDURE TFenPrincApplic.Vue_Calculs_Prop (VAR Msg: TMessage);
  BEGIN
   IF (mf_Checked AND GetMenuState(attr.Menu,cm_Vue_Calculs_Prop,mf_ByCommand)) <> 0
    THEN
     BEGIN
      CheckMenuItem(attr.Menu,cm_Vue_Calculs_Prop,mf_ByCommand OR mf_Unchecked);
      PLaFenCalculsProp^.Show(sw_Hide);
     END
    ELSE
     BEGIN
      CheckMenuItem(attr.Menu,cm_Vue_Calculs_Prop,mf_ByCommand OR mf_Checked);
      PLaFenCalculsProp^.Show(sw_Show);
     END;
  END;

{---------- PROCEDURE TFenPrincApplic.CMAideIndex ----------}
 PROCEDURE TFenPrincApplic.CMAideIndex (VAR Msg:TMessage);
  BEGIN
   WinHelp(HWindow, AideNomFich, Help_Index, 0);
   MessageBox(HWindow,
         'Cette fonctionnalité n''a pas encore été implémentéé',
         'Message d''information',mb_Ok OR mb_IconExclamation);
  END;

{---------- PROCEDURE TFenPrincApplic.CMAideUsing ----------}
 PROCEDURE TFenPrincApplic.CMAideUsing (VAR Msg: TMessage);
  BEGIN
   WinHelp(HWindow, 'WINHELP.HLP', Help_Index, 0);
  END;

{---------- PROCEDURE TFenPrincApplic.Presentation ----------}
 PROCEDURE TFenPrincApplic.Presentation (VAR Msg:TMessage);
  BEGIN
   FonctionNonImplementee;
  END;

{---------- PROCEDURE TFenPrincApplic.Mise_en_route ----------}
 PROCEDURE TFenPrincApplic.Mise_en_route (VAR Msg:TMessage);
  BEGIN
   FonctionNonImplementee;
  END;

{---------- PROCEDURE TFenPrincApplic.Fonctionnalites----------}
 PROCEDURE TFenPrincApplic.Fonctionnalites (VAR Msg:TMessage);
  BEGIN
   FonctionNonImplementee;
  END;

{---------- PROCEDURE TFenPrincApplic.Termes_math ----------}
 PROCEDURE TFenPrincApplic.Termes_math (VAR Msg:TMessage);
  BEGIN
   FonctionNonImplementee;
  END;
```

```
{------------------ PROCEDURE TFenPrincApplic.CMAProposDe ---------------}
 PROCEDURE TFenPrincApplic.CMAideAProposDe (VAR Msg:TMessage);
   VAR
     AideDialogue : TDialog;
   BEGIN
     AideDialogue.Init(@self, PChar(id_apropos));
     AideDialogue.Execute;
     AideDialogue.Done;
   END;

{------------------ PROCEDURE TFenPrincApplic.Done ----------------------}
 DESTRUCTOR TFenPrincApplic.Done;
   BEGIN
     StrDispose(AideNomFich);
     INHERITED Done;
   END;

{------------------ PROCEDURE TFenPrincApplic.WMDestroy -----------------}
{ Tell help system to close its window if open }
 PROCEDURE TFenPrincApplic.WMDestroy(VAR Msg: TMessage);
   BEGIN
     WinHelp(HWindow, AideNomFich, help_Quit, 0);
     TWindow.WMDestroy(Msg);
   END;

{------------------ TFenPrincApplic.WMNCButtonDblClk --------------------}
 CONST
   FreeMemory: ARRAY [1..3] OF LonginT = (0,0,0);
 PROCEDURE TFenPrincApplic.WMNCButtonDblClk(VAR Msg: TMessage);
   VAR
     s: ARRAY [0..41] OF Char;
   BEGIN
     FreeMemory[1] := MemAvail;
     FreeMemory[2] := FreeMemory[1]-FreeMemory[3];
     FreeMemory[3] := FreeMemory[1];
     WVSPrintF(s,'total: %ld, difference: %ld',FreeMemory);
     MessageBox(hWindow,s,'Free memory left',mb_IconInformation OR mb_OK);
   END;

{------------------ PROCEDURE TFenPrincApplic.WMLButtonDown -------------}
 PROCEDURE TFenPrincApplic.WMLButtonDown (VAR Msg: TMessage);
   BEGIN
   END;

{------------------ PROCEDURE TFenPrincApplic.WMRButtonDown -------------}
 PROCEDURE TFenPrincApplic.WMRButtonDown (VAR Msg: TMessage);
   var s : string;
   BEGIN
     SetCapture(FenAyantCapture);
   END;

{*************************************************************************}
{*************************** Initialisation ******************************}
{*************************************************************************}

BEGIN
END.
```

```
{#########################################################################}
{#                    Unité Paramet                                      #}
{#########################################################################}

UNIT Paramet; { Unité de la fenêtre d'affichage des paramètres }
        { des équations qui sont affichées dans la fenêtre }
        { des énoncés }

{*************************************************************************}
{***************************** Interface *********************************}
{*************************************************************************}

INTERFACE

USES WINTYPES, WINPROCS, OBJECTS, OWINDOWS, ODIALOGS, STRINGS, BWCC,
   VARPOINT, VARGLOB;

{$I idparam.inc}


TYPE
 PFenParametres = ^TFenParametres;
 TFenParametres = OBJECT (TWindow)
   Stat_1 : PStatic;
   Stat_2 : PStatic;
   Stat_3 : PStatic;
   CONSTRUCTOR Init(AParent:PWindowsObject;ATitle:PChar);
   PROCEDURE SetupWindow; VIRTUAL;
   DESTRUCTOR Done ; VIRTUAL;
   PROCEDURE ScrollBar_1_Msg(VAR Msg: TMessage);
     VIRTUAL id_First + id_Scrollbar_1;
   PROCEDURE ScrollBar_1a_Msg(VAR Msg: TMessage);
     VIRTUAL id_First + id_Scrollbar_1a;
   PROCEDURE ScrollBar_2_Msg(VAR Msg: TMessage);
     VIRTUAL id_First + id_Scrollbar_2;
   PROCEDURE ScrollBar_3_Msg(VAR Msg: TMessage);
     VIRTUAL id_First + id_Scrollbar_3;
   PROCEDURE WMDestroy (VAR Msg: TMessage);
     VIRTUAL wm_First + wm_Destroy;
   PROCEDURE BalayageBarDefilANeg (Hdle:HWnd;cpt:integer);
   PROCEDURE BalayageBarDefilAPos (Hdle:HWnd;cpt:integer);
   PROCEDURE BalayageBarDefilB (Hdle:HWnd;cpt:integer);
   PROCEDURE BalayageBarDefilC (Hdle:HWnd;cpt:integer);
   PROCEDURE WMTimer (VAR Msg: TMessage);
     VIRTUAL wm_First + wm_Timer;
 END;

{*************************************************************************}
{************************** Implementation *******************************}
{*************************************************************************}

IMPLEMENTATION

{================== PROCEDURES de TFenParametres ========================}

{----------------- CONSTRUCTEUR TFenParametres.Init --------------------}
 CONSTRUCTOR TFenParametres.Init(AParent:PWindowsObject;Atitle:PChar);
   VAR
     wtemp : integer;
     htemp : integer;
```

```
BEGIN
 TWindow.Init(AParent,ATitle);
 WITH attr DO
  BEGIN
  x :=5;
  y :=2;
  wtemp := getsystemmetrics(sm_CXscreen);
  htemp := getsystemmetrics(sm_CYscreen);
  IF wtemp = 800
   THEN
    w :=getsystemmetrics(sm_CXscreen) - LargEcranParam800
   ELSE
    BEGIN
     IF wtemp = 640
      THEN
       w :=getsystemmetrics(sm_CXscreen) - LargEcranParam640;
    END;
  IF htemp = 600
   THEN
    h := HautEcranParam600
   ELSE
    BEGIN
     IF htemp = 480
      THEN
       H := HautEcranParam480;
    END;
  style := style OR ws_Child OR ws_Visible
        OR ws_ClipSiblings OR ws_Caption OR ws_Border
        OR ws_SysMenu OR ws_OverLapped;
  END;
 new(Stat_1,Init(@self,id_Statictext_1,'param a:',2,5,80,25,12));
 new(Stat_2,Init(@self,id_Statictext_2,'param b:',2,35,80,25,12));
 new(Stat_3,Init(@self,id_Statictext_3,'param c(k):',2,65,80,25,12));
 new(Ed_1,Init(PFenParametres(@self),id_edittext_1,'',75,5,30,25,20,false));
 new(Ed_2,Init(PFenParametres(@self),id_edittext_2,'',75,35,30,25,4,false));
 new(Ed_3,Init(PFenParametres(@self),id_edittext_3,'',75,65,30,25,4,false));
 IF wtemp = 800
  THEN
   BEGIN
    new(Scroll_1,Init(PFenParametres(@Self),id_Scrollbar_1,110,7,LongScrollBar1800,20,true));

new(Scroll_1a,Init(PFenParametres(@Self),id_Scrollbar_1a,DepartScrollBar1800,7,LongScrollBar1800,20,tru
e));
    new(Scroll_2,Init(PFenParametres(@self),id_Scrollbar_2,110,37,LongScrollBar2800,20,true));
    new(Scroll_3,Init(PFenParametres(@self),id_Scrollbar_3,110,67,LongScrollBar3800,20,true));
   END
  ELSE
   BEGIN
    IF wtemp = 640
     THEN
      BEGIN
       new(Scroll_1,Init(PFenParametres(@Self),id_Scrollbar_1,110,7,LongScrollBar1640,20,true));

new(Scroll_1a,Init(PFenParametres(@Self),id_Scrollbar_1a,DepartScrollBar1640,7,LongScrollBar1640,20,tru
e));
       new(Scroll_2,Init(PFenParametres(@self),id_Scrollbar_2,110,37,LongScrollBar2640,20,true));
       new(Scroll_3,Init(PFenParametres(@self),id_Scrollbar_3,110,67,LongScrollBar3640,20,true));
      END;
   END;
END;
```

```
{---------- PROCEDURE TFenParametres.SetupWindow ----------}
 PROCEDURE TFenParametres.SetupWindow;
  VAR
   aa : ARRAY [0..10] OF Char;
   aa1: ARRAY [0..10] OF Char;
   bb : ARRAY [0..10] OF Char;
   cc : ARRAY [0..10] OF Char;
  BEGIN
   INHERITED SetupWindow;
   Scroll_1^.SetPosition(parametre1);
   Scroll_1a^.SetPosition(parametre1);
   Scroll_2^.SetPosition(parametre2);
   Scroll_3^.SetPosition(parametre3);
   Scroll_1^.SetRange(BarDefIntervalInf,-1);
   Scroll_1a^.SetRange(1,BarDefIntervalSup);
   Scroll_2^.SetRange(BarDefIntervalInf,BarDefIntervalSup);
   Scroll_3^.SetRange(BarDefIntervalInf,BarDefIntervalSup);
   Scroll_1^.LineMagnitude := 1;
   Scroll_1a^.LineMagnitude := 1;
   Scroll_2^.LineMagnitude := 1;
   Scroll_3^.LineMagnitude := 1;
   str(parametre1,aa);
   Ed_1^.SetText(aa);
   str(parametre1,aa1);
   Ed_1^.SetText(aa1);
   str(parametre2,bb);
   Ed_2^.SetText(bb);
   str(parametre3,cc);
   Ed_3^.SetText(cc);
   SetTimer(HWindow,TimerBarDef_ID,Timer_Intervalle,NIL);
  END;

{---------- PROCEDURE TFenParametres.ScrollBar_1_Msg ----------}
 PROCEDURE TFenParametres.ScrollBar_1_Msg(VAR Msg:TMessage);
  VAR
   CString1: ARRAY [0..10] OF char;
   PaintDC : HDC;
   PaintInfo:TPaintStruct;
  BEGIN
   KillTimer(HWindow,TimerBarDef_ID);
   ActiverParamAPos := False;
   Str(Scroll_1^.GetPosition,CString1);
   Ed_1^.SetText(CString1);
   parametre1 := Scroll_1^.GetPosition;
   InvalidateRect(ptFenGraphe^.HWindow,nil,true);
   ptFenGraphe^.Paint(PaintDC,PaintInfo);
   InvalidateRect(ptFenCalculsProp^.HWindow,nil,true);
   ptFenCalculsProp^.Paint(PaintDC,PaintInfo);
  END;

{---------- PROCEDURE TFenParametres.ScrollBar_1a_Msg ----------}
 PROCEDURE TFenParametres.ScrollBar_1a_Msg(VAR Msg:TMessage);
  VAR
   CString1a: ARRAY [0..10] OF Char;
   PaintDC : HDC;
   PaintInfo:TPaintStruct;
  BEGIN
   KillTimer(HWindow,TimerBarDef_ID);
   ActiverParamANeg := False;
```

```
    Str(Scroll_1a^.GetPosition,CString1a);
    Ed_1^.SetText(CString1a);
    parametre1 := Scroll_1a^.GetPosition;
    InvalidateRect(ptFenGraphe^.HWindow,nil,true);
    ptFenGraphe^.Paint(PaintDC,PaintInfo);
    InvalidateRect(ptFenCalculsProp^.HWindow,nil,true);
    ptFenCalculsProp^.Paint(PaintDC,PaintInfo);
   END;

{------------------ PROCEDURE TFenParametres.ScrollBar_2_Msg ------------}
 PROCEDURE TFenParametres.ScrollBar_2_Msg(VAR Msg:TMessage);
   VAR
    CString2: ARRAY [0..10] OF char;
    PaintDC : HDC;
    PaintInfo:TPaintStruct;
   BEGIN
    KillTimer(HWindow,TimerBarDef_ID);
    ActiverParamBPos := False;
    ActiverParamBNeg := False;
    Str(Scroll_2^.GetPosition,CString2);
    Ed_2^.SetText(CString2);
    parametre2 := Scroll_2^.GetPosition;
    InvalidateRect(ptFenGraphe^.HWindow,nil,true);
    ptFenGraphe^.Paint(PaintDC,PaintInfo);
    InvalidateRect(ptFenCalculsProp^.HWindow,nil,true);
    ptFenCalculsProp^.Paint(PaintDC,PaintInfo);
    SetTimer(HWindow,TimerBarDef_ID,Timer_Intervalle,NIL);
   END;

{------------------ PROCEDURE TFenParametres.ScrollBar_3_Msg ------------}
 PROCEDURE TFenParametres.ScrollBar_3_Msg(VAR Msg:TMessage);
   VAR
    CString3: ARRAY [0..10] OF char;
    PaintDC : HDC;
    PaintInfo:TPaintStruct;
   BEGIN
    KillTimer(HWindow,TimerBarDef_ID);
    ActiverParamCPos := False;
    ActiverParamCNeg := False;
    Str(Scroll_3^.GetPosition,CString3);
    Ed_3^.SetText(CString3);
    parametre3 := Scroll_3^.GetPosition;
    InvalidateRect(ptFenGraphe^.HWindow,nil,true);
    ptFenGraphe^.Paint(PaintDC,PaintInfo);
    InvalidateRect(ptFenCalculsProp^.HWindow,nil,true);
    ptFenCalculsProp^.Paint(PaintDC,PaintInfo);
   END;

{------------------ PROCEDURE TFenParametres.WMDestroy ------------------}
 PROCEDURE TFenParametres.WMDestroy (VAR Msg: TMessage);
   BEGIN
    KillTimer(HWindow,TimerBarDef_ID);
    TWindow.WMDestroy(Msg);
   END;

{------------------ PROCEDURE TFenParametres.BalayageBarDefilANeg -------}
 PROCEDURE TFenParametres.BalayageBarDefilANeg (Hdle:HWnd;cpt:integer);
   VAR
    cString : ARRAY [0..10] OF Char;
   BEGIN
```

```
    Hdle := GetDlgItem(HWindow,id_ScrollBar_1);
    cpt := parametre1;
    Scroll_1^.SetPosition(cpt);
    Str(cpt,cString);
    Ed_1^.SetText(cString);
   END;

{------------------ PROCEDURE TFenParametres.BalayageBarDefilAPos -------}
 PROCEDURE TFenParametres.BalayageBarDefilAPos (Hdle:HWnd;cpt:integer);
   VAR
    cString : ARRAY [0..10] OF Char;
   BEGIN
    Hdle := GetDlgItem(HWindow,id_ScrollBar_1a);
    cpt := parametre1;
    Scroll_1a^.SetPosition(cpt);
    Str(cpt,cString);
    Ed_1^.SetText(cString);
   END;

{------------------ PROCEDURE TFenParametres.BalayageBarDefil -----------}
 PROCEDURE TFenParametres.BalayageBarDefilB (Hdle:HWnd;cpt:integer);
   VAR
    cString : ARRAY [0..10] OF Char;
   BEGIN
    Hdle := GetDlgItem(HWindow,id_ScrollBar_2);
    cpt := parametre2;
    Scroll_2^.SetPosition(cpt);
    Str(cpt,cString);
    Ed_2^.SetText(cString);
   END;

{------------------ PROCEDURE TFenParametres.BalayageBarDefilC ----------}
 PROCEDURE TFenParametres.BalayageBarDefilC (Hdle:HWnd;cpt:integer);
   VAR
    cString : ARRAY [0..10] OF Char;
   BEGIN
    Hdle := GetDlgItem(HWindow,id_ScrollBar_3);
    cpt := parametre3;
    Scroll_3^.SetPosition(cpt);
    Str(cpt,cString);
    Ed_3^.SetText(cString);
   END;

{------------------ PROCEDURE TFenParametres.WMTimer --------------------}
 PROCEDURE TFenParametres.WMTimer (VAR Msg: TMessage);
   BEGIN
    IF ActiverParamAPos = true
     THEN
      BEGIN
       { IF parametre1 <> 1
         THEN parametre1 := 1;}
       IF parametre1 < BarDefIntervalSup
        THEN
         BEGIN
          parametre1 := parametre1 + 1;
          BalayageBarDefilAPos(id_ScrollBar_1a,parametre1);
         END
        ELSE
         BEGIN
          parametre1 := 1;
```

```
        BalayageBarDefilAPos(id_ScrollBar_1a,parametre1);
       END;
   END;
IF ActiverParamANeg = true
 THEN
  BEGIN
   {IF parametre1 <> -1
    THEN parametre1 := -1;}
   IF parametre1 > BarDefIntervalInf
    THEN
     BEGIN
      parametre1 := parametre1 - 1;
      BalayageBarDefilANeg(id_ScrollBar_1,parametre1);
     END
    ELSE
     BEGIN
      parametre1 := -1;
      BalayageBarDefilANeg(id_ScrollBar_1,parametre1);
     END;
   END;
IF ActiverParamBPos = true
 THEN
  BEGIN
   IF parametre2 < BarDefIntervalSup
    THEN
     BEGIN
      parametre2 := parametre2 + 1;
      BalayageBarDefilB(id_ScrollBar_2,parametre2);
     END
    ELSE
     BEGIN
      parametre2 := 0;
      BalayageBarDefilB(id_ScrollBar_2,parametre2);
     END;
   END;
IF ActiverParamBNeg = true
 THEN
  BEGIN
   IF parametre2 > BarDefIntervalInf
    THEN
     BEGIN
      parametre2 := parametre2 - 1;
      BalayageBarDefilB(id_ScrollBar_2,parametre2);
     END
    ELSE
     BEGIN
      parametre2 := 0;
      BalayageBarDefilB(id_ScrollBar_2,parametre2);
     END;
   END;
IF ActiverParamCPos = true
 THEN
  BEGIN
   IF parametre3 < BarDefIntervalSup
    THEN
     BEGIN
      parametre3 := parametre3 + 1;
      BalayageBarDefilC(id_ScrollBar_3,parametre3);
     END
    ELSE
```

```
          BEGIN
            parametre3 := 0;
            BalayageBarDefilC(id_ScrollBar_3,parametre3);
          END;
      END;
    IF ActiverParamCNeg = true
     THEN
      BEGIN
       IF parametre3 > BarDefIntervalInf
        THEN
          BEGIN
            parametre3 := parametre3 - 1;
            BalayageBarDefilC(id_ScrollBar_3,parametre3);
          END
        ELSE
          BEGIN
            parametre3 := 0;
            BalayageBarDefilC(id_ScrollBar_3,parametre3);
          END;
      END;
  END;

{------------------ DESTRUCTEUR TFenParametres.Done ---------------------}
 DESTRUCTOR TFenParametres.Done;
  BEGIN
   INHERITED Done;
  END;

{*************************************************************************}
{***************************** Initialisation ****************************}
{*************************************************************************}

BEGIN
END.
```

```
{##############################################################################}
{#                    Unité EnonEq                                            #}
{##############################################################################}

UNIT EnonEq; { Unité de la fenêtre d'affichage des énoncés des équations }

{******************************************************************************}
{***************************** Interface **************************************}
{******************************************************************************}

INTERFACE

USES
 WINTYPES, WINPROCS, OWINDOWS, STRINGS, WINCRT,
 WINDOS, ODIALOGS, OBJECTS, OSTDDLGS,
 VARGLOB;

CONST
 id_EditBoite = 107;

TYPE
 PFenEnonEq = ^TFenEnonEq;
 TFenEnonEq = OBJECT(TWindow)
  CONSTRUCTOR Init(AParent:PWindowsObject;ATitle:PChar);
  PROCEDURE SetupWindow; VIRTUAL;
  DESTRUCTOR Done; VIRTUAL;
 END;

{******************************************************************************}
{************************** Implementation ************************************}
{******************************************************************************}

IMPLEMENTATION

{================== PROCEDURES de TFenEnonEq ================================}

{----------------- CONSTRUCTEUR TFenEnonEQ.Init ----------------------}
 CONSTRUCTOR TFenEnonEq.Init(AParent:PWindowsObject;ATitle:PChar);
  VAR
   wtemp : integer;
   ytemp : integer;
   EditTemp : integer;
  BEGIN
   INHERITED init(AParent,atitle);
   WITH attr DO
    BEGIN
     x := 5;
     ytemp := GetSystemMetrics(sm_CYscreen);
     IF ytemp = 600
      THEN
       y := GetSystemMetrics(sm_CYscreen) - CoordYEcranEnonce600
      ELSE
       BEGIN
        IF ytemp = 480
         THEN
          y := GetSystemMetrics(sm_CYscreen) - CoordYEcranEnonce480;
       END;
     wtemp := GetSystemMetrics(sm_CXscreen);
     IF wtemp = 800
      THEN
```

```
        w := GetSystemMetrics(sm_CXscreen) - LargEcranEnonce800
       ELSE
        BEGIN
         IF wtemp = 640
          THEN
           w := GetSystemMetrics(sm_CXscreen) - LargEcranEnonce640;
        END;
      h := 55;
      style := ws_Child OR ws_ClipSiblings OR ws_Visible
           OR ws_SysMenu OR ws_Caption OR ws_Border
           OR ws_OverLapped;
    END;
   EditTemp := GetSystemMetrics(sm_CXscreen);
   IF EditTemp = 800
    THEN
     New(EditBoite, Init(PFenEnonEq(@self),id_EditBoite,
       '',20,3,320,28,45,false))
    ELSE
     BEGIN
      IF EditTemp = 640
       THEN
        New(EditBoite, Init(PFenEnonEq(@self),id_EditBoite,
          '',20,3,230,28,45,false));
     END;
  END;

{------------------ PROCEDURE TFenEnonEq.EcrireEq ----------------------}
 PROCEDURE TFenEnonEq.SetupWindow;
  BEGIN
   INHERITED SetupWindow;
   UpdateWindow(HWindow);
  END;

{------------------ PROCEDURE TFenEnonEq.Done --------------------------}
 DESTRUCTOR TFenEnonEq.Done;
  BEGIN
   dispose(EditBoite,done);
   INHERITED Done;
  END;

{*************************************************************************}
{************************** Initialisation ***************************}
{*************************************************************************}

BEGIN
END.
```

```
{#########################################################################}
{#                    Unité DFamFonc                                     #}
{#########################################################################}

UNIT DFamFonc;   { Unité de la boîte de dialogue du menu }
            { des familles de fonctions}

{$R dfamfonc.res} { Ressource de la boîte de dialogue }
            { des familles de fontions}

{*************************************************************************}
{************************** Interface ************************************}
{*************************************************************************}

INTERFACE

USES
  WINTYPES, WINPROCS, OWINDOWS, OBJECTS, WINDOS,
  WINCRT, STRINGS, OSTDDLGS, ODIALOGS,
  VARGLOB;

{$I dfamfonc.inc}

TYPE
  PDiaFamFonc = ^TDiaFamFonc;
  TDiaFamFonc = OBJECT (TDialog)
    ValArt1 : integer;
    ValArt2 : integer;
    ValArt3 : integer;
    HControl1 : HWnd;
    HControl2 : HWnd;
    HControl3 : HWnd;
    index1 : integer;
    index2 : integer;
    index3 : integer;
    s5 : PChar;
    s6 : ARRAY [0..64] OF Char;
    s7 : ARRAY [0..64] OF Char;
    PROCEDURE SetupWindow; VIRTUAL;
    PROCEDURE Ok (VAR Msg: TMessage);
      VIRTUAL id_First + id_Ok;
    PROCEDURE Cancel (VAR Msg: TMessage);
      VIRTUAL id_First + id_Cancel;
    PROCEDURE Help (VAR Msg: TMessage);
      VIRTUAL id_First + id_Help;
    PROCEDURE InitialiserParametres;
    PROCEDURE IDDroite (VAR Msg : TMessage);
      VIRTUAL id_First + id_ff_droite;
    PROCEDURE IDParabole (VAR Msg : TMessage);
      VIRTUAL id_First + id_ff_parabole;
    PROCEDURE IDTrigono (VAR Msg : TMessage);
      VIRTUAL id_First + id_ff_trigono;
    PROCEDURE WMDestroy (VAR Msg: TMessage);
      VIRTUAL wm_First + wm_Destroy;
    PROCEDURE Balayage (HCtrl:HWnd;ValeurArt:integer);
    PROCEDURE Balayage1 (HCtrl:HWnd);
    PROCEDURE WMTimer (VAR Msg: TMessage);
      VIRTUAL wm_First + wm_Timer;
  END;
```

```
{*************************************************************************}
{************************** Implementation ***************************}
{*************************************************************************}

IMPLEMENTATION

USES
 VARPOINT;

{==================== PROCEDURE SetListBox ===================================}
{ Ajoute la chaîne dont l'adresse est donnée par P à la boîte combinée
 identifiée par CtrlID }

 PROCEDURE SetListBox ( HDlg:HWnd; CtrlID:Word; P:PChar);
  BEGIN
   SendDlgItemMessage(HDlg,CtrlID,lb_AddString,0,LongInt(P));
  END;

{==================== PROCEDURE GetListBox ===================================}
 PROCEDURE GetListBox ( HDlg:HWnd; CtrlID:Word; VAR P:PChar; VAR Len:Word);
  VAR
   Index : integer;
   HControl : HWnd;
  BEGIN
   P := NIL;
   HControl := GetDlgItem(HDlg,CtrlID);
   Index := SendMessage(HControl,lb_GetCurSel,0,0);
   IF Index <> cb_Err THEN
    BEGIN
     Len := SendMessage(HControl,lb_GetTextLen,Index,0);
     IF Len > 0 THEN
      BEGIN
       GetMem(P,Len+1);
       IF P <> NIL THEN
        SendMessage(HControl,lb_GetText,Index,LongInt(P))
      END
    END;
   ReleaseDC(HDlg,HControl);
  END;

{================== PROCEDURES de TDiaFamFonc ================================}

{----------------- PROCEDURE TDiaFamFonc.SetupWindow ----------------}
 PROCEDURE TDiaFamFonc.SetupWindow;
  VAR
   i : integer;
  BEGIN
   TDialog.SetupWindow;
   SetTimer(HWindow, Timer_ID, Timer_Intervalle, NIL);
   FOR i := 0 TO NbEqDroite-1 DO
    SetListBox(HWindow,id_ff_droite,EqDroite[i]);
   FOR i := 0 TO NbEqParabole-1 DO
    SetListBox(HWindow,id_ff_parabole,EqParabole[i]);
   FOR i := 0 TO NbEqTrigo-1 DO
    SetListBox(HWindow,id_ff_trigono,EqTrigo[i]);
   ValArt1 := 0;
   ValArt2 := 0;
   ValArt3 := 0;
   HControl1 := GetDlgItem(HWindow,id_ff_droite);
   HControl2 := GetDlgItem(HWindow,id_ff_parabole);
```

```
    HControl3 := GetDlgItem(HWindow,id_ff_trigono);
    index1 := 0;
    index2 := 0;
    index3 := 0;
  END;

{------------------ PROCEDURE TDiaFamFonc.Ok ---------------------------}
 PROCEDURE TDiaFamFonc.Ok (VAR Msg: TMessage);
  VAR
    s1, s2, s3 : PChar;  {pointeurs sur sélections de la boîte combo}
    l1, l2, l3 : Word;   {longueurs en octets des tableaux s1 à s3}
    s4 : ARRAY[0..64] OF Char;
  BEGIN
    KillTimer(HWindow,Timer_ID);
    s4[0] := Chr(0);
    GetListBox(HWindow,id_ff_droite,s1,l1);
    GetListBox(HWindow,id_ff_parabole,s2,l2);
    GetListBox(HWindow,id_ff_trigono,s3,l3);
    IF s1 <> Nil THEN strCat(s4,s1);
    {StrCat(S4, ', ');}
    IF s2 <> NIL THEN strCat(s4,s2);
    {StrCat(s4, ', ');}
    IF s3 <> NIL THEN strCat(s4,s3);
    IF s1 <> NIL THEN FreeMem(s1,l1+1);
    IF s2 <> NIL THEN FreeMem(s2,l2+1);
    IF s3 <> NIL THEN FreeMem(s3,l3+1);
   { EditBoite^.SetText(s4);}
    TDialog.Ok(Msg);
  END;

{------------------ PROCEDURE TDiaFamFonc.Cancel ----------------------}
 PROCEDURE TDiaFamFonc.Cancel (VAR Msg: TMessage);
  BEGIN
    KillTimer(HWindow,Timer_ID);
    TDialog.Cancel(Msg);
  END;

{------------------ PROCEDURE TFenParametres.Help ----------------------}
 PROCEDURE TDiaFamFonc.Help (VAR Msg: TMessage);
  BEGIN
    MessageBox(HWindow,
          'Cette fonctionnalité n''a pas encore été implémentée',
          'Message d''information',mb_OK OR mb_IconExclamation);
  END;

{------------------ PROCEDURE TFenParametres.InitialiserParametres ------}
 PROCEDURE TDiaFamFonc.InitialiserParametres;
  VAR
   aa : ARRAY [0..10] OF Char;
   aa1: ARRAY [0..10] OF Char;
   bb : ARRAY [0..10] OF Char;
   cc : ARRAY [0..10] OF Char;
  BEGIN
   IF DroiteBool
     THEN
       BEGIN
         parametre1 := 1;
         parametre2 := 1;
         parametre3 := 0;
       END;
```

```
    IF ParaboleBool OR TrigonoBool
     THEN
       BEGIN
         parametre1 := 1;
         parametre2 := 1;
         parametre3 := 1;
       END;
    Scroll_1^.SetPosition(parametre1);
    Scroll_1a^.SetPosition(parametre1);
    Scroll_2^.SetPosition(parametre2);
    Scroll_3^.SetPosition(parametre3);
    str(parametre1,aa);
    Ed_1^.SetText(aa);
    str(parametre1,aa1);
    Ed_1^.SetText(aa1);
    str(parametre2,bb);
    Ed_2^.SetText(bb);
    str(parametre3,cc);
    Ed_3^.SetText(cc);
  END;

{------------------ PROCEDURE TDiaFamFonc.IDDroite ----------------------}
 PROCEDURE TDiaFamFonc.IDDroite (VAR Msg : TMessage);
  VAR
    s1 : PChar;
    l1 : Word;
    SelectedText : ARRAY [0..30] OF Char;
    aa : ARRAY [0..10] OF Char;
    bb : ARRAY [0..10] OF Char;
    cc : ARRAY [0..10] OF Char;
  BEGIN
    GetListBox(HWindow,id_ff_droite,s1,l1);
    EcrireTexte := s1;
    EditBoite^.SetText(EcrireTexte);
    InvalidateRect(ptFenGraphe^.HWindow,NIL,true);
    InitialiserParametres;
    DroiteBool := True;
    IF ParaboleBool THEN ParaboleBool := false;
    IF TrigonoBool THEN TrigonoBool := false;
  END;

{------------------ PROCEDURE TDiaFamFonc.IDParab ----------------------}
 PROCEDURE TDiaFamFonc.IDParabole (VAR Msg : TMessage);
  VAR
    s2 : PChar;
    l2 : Word;
    SelectedText : ARRAY [0..30] OF Char;
    aa : ARRAY [0..10] OF Char;
    bb : ARRAY [0..10] OF Char;
    cc : ARRAY [0..10] OF Char;
  BEGIN
    GetListBox(HWindow,id_ff_parabole,s2,l2);
    EcrireTexte := s2;
    EditBoite^.SetText(EcrireTexte);
    InvalidateRect(ptFenGraphe^.HWindow,NIL,true);
    InitialiserParametres;
    ParaboleBool := True;
    IF DroiteBool THEN DroiteBool := false;
    IF TrigonoBool THEN TrigonoBool := false;
  END;
```

```
{---------- PROCEDURE TDiaFamFonc.IDTrigo ----------}
 PROCEDURE TDiaFamFonc.IDTrigono (VAR Msg : TMessage);
  VAR
   s3 : PChar;
   l3 : Word;
   SelectedText : ARRAY [0..30] OF Char;
   aa : ARRAY [0..10] OF Char;
   bb : ARRAY [0..10] OF Char;
   cc : ARRAY [0..10] OF Char;
  BEGIN
   GetListBox(HWindow,id_ff_trigono,s3,l3);
   EcrireTexte := s3;
   EditBoite^.SetText(EcrireTexte);
   InvalidateRect(ptFenGraphe^.HWindow,NIL,true);
   InitialiserParametres;
   TrigonoBool := True;
   IF DroiteBool THEN DroiteBool := false;
   IF ParaboleBool THEN ParaboleBool := false;
  END;

{---------- PROCEDURE TDiaFamFonc.WMDestroy ----------}
 PROCEDURE TDiaFamFonc.WMDestroy (VAR Msg: TMessage);
  BEGIN
   KillTimer(HWindow, Timer_ID);
   ReleaseDC(HWindow,HControl1);
   ReleaseDC(HWindow,HControl2);
   ReleaseDC(HWindow,HControl3);
   TDialog.WMDestroy(Msg);
  END;

{---------- PROCEDURE TDiaFamFonc.Balayage ----------}
 PROCEDURE TDiaFamFonc.Balayage(HCtrl:HWnd;ValeurArt:integer);
  BEGIN
   SendMessage(HCtrl,lb_SetCurSel,ValeurArt,0);
   SendMessage(HCtrl,wm_SetFocus,ValeurArt,0);
  END;

{---------- PROCEDURE TDiaFamFonc.Balayage1 ----------}
 PROCEDURE TDiaFamFonc.Balayage1(HCtrl:HWnd);
  BEGIN
   SendMessage(HCtrl,wm_Killfocus,0,0);
  END;

{---------- PROCEDURE TDiaFamFonc.WMTimer ----------}
 PROCEDURE TDiaFamFonc.WMTimer (VAR Msg: TMessage);
  BEGIN
   index1 := SendMessage(HControl1,lb_GetCount,0,0);
   IF ValArt1 <= Index1
    THEN
     BEGIN
      Balayage(HControl1,ValArt1);
      ValArt1 := ValArt1 + 1;
     END
    ELSE
     BEGIN
      balayage1(HControl1);
      index2 := SendMessage(HControl2,lb_GetCount,0,0);
      IF ValArt2 <= index2
       THEN
```

```
          BEGIN
            Balayage(HControl2,ValArt2);
            ValArt2 := ValArt2 + 1;
          END
        ELSE
          BEGIN
            Balayage1(HControl2);
            index3 := SendMessage(HControl3,lb_GetCount,0,0);
            IF ValArt3 <= index3
              THEN
                BEGIN
                  Balayage(HControl3,ValArt3);
                  ValArt3 := ValArt3 + 1;
                END
              ELSE
                BEGIN
                  ValArt1 := 0;
                  Balayage(HControl1,ValArt1);
                  ValArt2 := 0;
                  Balayage(HControl2,ValArt2);
                  ValArt3 := 0;
                  Balayage(HControl3,ValArt3);
                END;
          END;
      END;
  END;

{*****************************************************************************}
{*************************** Initialisation ********************************}
{*****************************************************************************}

BEGIN
END.
```

```
{##############################################################################}
{#                    Unité Souris                                            #}
{##############################################################################}

UNIT Souris; { Unité de la fenêtre des coordonnées de la souris }

{******************************************************************************}
{*************************** Interface ****************************************}
{******************************************************************************}

INTERFACE

USES
 WINTYPES, WINPROCS, OWINDOWS, STRINGS, WINDOS,
 OBJECTS, OSTDDLGS, VARGLOB;

TYPE
 PFenSouris = ^TFenSouris;
 TFenSouris = OBJECT (TWindow)
  CONSTRUCTOR Init(AParent: PWindowsObject; ATitle: PChar);
  DESTRUCTOR Done; VIRTUAL;
 END;

{******************************************************************************}
{*************************** Implementation ***********************************}
{******************************************************************************}

IMPLEMENTATION

{==================== PROCEDURES de PFenSouris ================================}

{----------------- Constructeur PFenSouris.Init ----------------------}
 CONSTRUCTOR TFenSouris.Init(AParent: PWindowsObject; ATitle: PChar);
  VAR
   xtemp : integer;
   ytemp : integer;
  BEGIN
   INHERITED Init(AParent, ATitle);
   WITH attr DO
    BEGIN
     xtemp := GetSystemMetrics(sm_CXscreen);
     IF xtemp = 800
      THEN
       x := GetSystemMetrics(sm_CXscreen) - CoordXEcranSouris800
      ELSE
       BEGIN
        IF xtemp = 640
         THEN
          x := GetSystemMetrics(sm_CXscreen) - CoordXEcranSouris640;
       END;
     ytemp := GetSystemMetrics(sm_CYscreen);
     IF ytemp = 600
      THEN
       y := GetSystemMetrics(sm_CYscreen) - CoordYEcranEnonce600
      ELSE
       BEGIN
        IF ytemp = 480
         THEN
          y := GetSystemMetrics(sm_CYscreen) - CoordYEcranEnonce480;
       END;
```

```
      w := 100;
      h := 55;
      style := ws_Child OR ws_ClipChildren OR ws_Visible
            OR ws_Border OR ws_ClipSiblings OR ws_Caption
            OR ws_SysMenu OR ws_OverLapped;
    END;
  END;

{------------------ DESTRUCTEUR TFenSouris.Done -------------------------}
 DESTRUCTOR TFenSouris.Done;
  BEGIN
   INHERITED Done;
  END;

{*****************************************************************************}
{***************************** Initialisation ********************************}
{*****************************************************************************}

BEGIN
END.
```

```
{##############################################################################}
{#                    Unité FGraphe1                                          #}
{##############################################################################}

UNIT FGraphe1;

{$R curcroix.res}

{******************************************************************************}
{***************************** Interface **************************************}
{******************************************************************************}

INTERFACE

USES
 WINTYPES, WINPROCS, OWINDOWS, STRINGS, WINDOS,
 ODIALOGS, OBJECTS, OSTDDLGS,
 SOURIS, VARGLOB;

TYPE
 PFenGraphe = ^TFenGraphe;
 PDessinFonctionCollect = ^TDessinFonctionCollect;

 TFenGraphe = OBJECT (TWindow)
  LButtonDown : Boolean;
  coordX1, coordY1, coordX2, coordY2 : Integer;
  ptDessinFonction : PDessinFonctionCollect;
  ptFenSouris : PFenSouris;
  CONSTRUCTOR Init(AParent:PWindowsObject;ATitle:PChar;
            ptgFenSouris:PFenSouris;ptgDessinFonction:PDessinFonctionCollect);
  PROCEDURE SetupWindow; VIRTUAL;
  DESTRUCTOR Donc; VIRTUAL;
  FUNCTION GetClassName : PChar; VIRTUAL;
  PROCEDURE GetWindowClass(VAR AWndClass:TWndClass); VIRTUAL;
  PROCEDURE Paint(paintDC:HDC; VAR PaintInfo:TPaintStruct); VIRTUAL;
  PROCEDURE ShowMouseLocation(x:integer;y:integer);
  PROCEDURE WMLButtonDown(VAR Msg: TMessage);
   VIRTUAL wm_First + wm_LButtonDown;
  PROCEDURE WMLButtonUp(VAR Msg: TMessage);
   VIRTUAL wm_First + wm_LButtonUp;
  PROCEDURE WMMouseMove(VAR Msg: TMessage);
   VIRTUAL wm_First + wm_MouseMove;
 END;

 TDessinFonctionCollect = OBJECT (TCollection)
  ptOwnerWindow : PFenGraphe;
  CONSTRUCTOR Init(ptgOwnerWindow:PFenGraphe);
  DESTRUCTOR Done; VIRTUAL;
  PROCEDURE Reset;
  PROCEDURE CouleurAxesInit(aDC:HDC);
  PROCEDURE CouleurAxesDone(aDC:HDC);
  PROCEDURE CouleurCourbesInit(aDC:HDC);
  PROCEDURE CouleurCourbesDone(aDC:HDC);
  PROCEDURE CouleurProprietesInit(aDC:HDC);
  PROCEDURE CouleurProprietesDone(aDC:HDC);
  PROCEDURE DessinFonctionDroite(aDC:HDC);
  PROCEDURE DessinFonctionParabole(aDC:HDC);
  PROCEDURE DessinFonctionTrigono(aDC:HDC);
  PROCEDURE Display (paWindow:PFenGraphe);
  PROCEDURE Paint(aDC:HDC);
```

```
  CONSTRUCTOR Load (VAR s: TStream);
  PROCEDURE Store (VAR s: TStream); VIRTUAL;
 END;

CONST
 RDessinFonctionCollect : TStreamRec =
            (objType : 1000;
             vMTLink : Ofs(typeOf(TDessinFonctionCollect)^);
             Load : @TDessinFonctionCollect.Load;
             Store : @TDessinFonctionCollect.Store);

{*************************************************************************}
{************************** Implementation ******************************}
{*************************************************************************}

IMPLEMENTATION

{=================== PROCEDURES de TFenGraphe ============================}

{----------------- CONSTRUCTOR TFenGraphe.Init ----------------------}
 CONSTRUCTOR TFenGraphe.Init(AParent:PWindowsObject;ATitle:PChar;
       ptgFenSouris:PFenSouris;ptgDessinFonction:PDessinFonctionCollect);
  VAR
   ytemp : integer;
   htemp : integer;
  BEGIN
   INHERITED init(AParent,atitle);
   WITH attr DO
   BEGIN
    x :=5;
    ytemp := getsystemmetrics(sm_CYscreen);
    IF ytemp = 600
     THEN
      y :=getsystemmetrics(sm_CYscreen)- CoordYEcranGraphe600
     ELSE
      BEGIN
       IF ytemp = 480
        THEN
         y :=getsystemmetrics(sm_CYscreen)- CoordYEcranGraphe480;
      END;
    w :=getsystemmetrics(sm_CXscreen)-143;
    htemp := getsystemmetrics(sm_CYscreen);
    IF htemp = 600
     THEN
      h :=HautEcranGraphe600
     ELSE
      BEGIN
       IF htemp = 480
        THEN
         h :=HautEcranGraphe480;
      END;
    style := ws_Child OR ws_Visible OR ws_ClipChildren
         OR ws_Border OR ws_Clipsiblings OR ws_OverLapped
  END;
  ptFenSouris := ptgFenSouris;
  ptDessinFonction := ptgDessinFonction;
  LButtonDown := False;
 END;
```

```
{---------- PROCEDURE TFenGraphe.SetupWindow ----------}
 PROCEDURE TFenGraphe.SetupWindow;
  BEGIN
   INHERITED SetupWindow;
   UpdateWindow(HWindow);
  END;

{---------- Fonction TFenGraphe.GetClassName ----------}
 FUNCTION TFenGraphe.GetClassName : PChar;
  BEGIN
   GetClassName := 'TCurseur';
  END;

{---------- PROCEDURE TFenGraphe.GetWindowClass ----------}
 PROCEDURE TFenGraphe.GetWindowClass(VAR AWndClass: TWndClass);
  BEGIN
   INHERITED GetWindowclass(AWndClass);
   AWndClass.HCursor := LoadCursor(HInstance,'cursor_1');
  END;

{---------- PROCEDURE TFenGraphe.ShowMouseLocation ----------}
 PROCEDURE TFenGraphe.ShowMouseLocation(x:integer;y:integer);
  VAR
   SX, SY: STRING[5];
   S: STRING[25];
  BEGIN
   Str(X:3, SX);
   Str(Y:3, SY);
   S := 'X:'+SX+' ;  '+'Y:'+SY+' ';
   SourisDC := GetDC(ptFenSouris^.HWindow);
   SetTextColor(SourisDC,RGB(0,0,225));
   TextOut(SourisDC, 3, 10, @S[1], Length(S));
   ReleaseDC(ptFenSouris^.HWindow,SourisDC);
  END;

{---------- PROCEDURE TFenGraphe.Paint ----------}
{Dessine à l'écran le contenu de la fenêtre}
 PROCEDURE TFenGraphe.Paint(paintDC:HDC;VAR PaintInfo:TPaintStruct);
  BEGIN
   ptDessinFonction^.Paint(PaintDC);
   {peut-être mettre dans PaintDC le handle de la fenêtre des graphes ? }
  END;

{---------- PROCEDURE TFenSouris.WMLButtonDown ----------}
 PROCEDURE TFenGraphe.WMLButtonDown(VAR Msg: TMessage);
  BEGIN
   EstModifie := true;
   IF NOT LButtonDown THEN WITH Msg DO
    BEGIN
     ShowMouseLocation(LOWORD(LParam),HIWORD(LParam));
     coordX1 := LParamLo;
     coordY1 := LParamHi;
     coordX2 := coordX1;
     coordY2 := coordY1;
     LButtonDown := True;
     SetCapture(HWindow);
    END;
  END;
```

```
{------------------ PROCEDURE TFenSouris.WMLButtonUp --------------------}
 PROCEDURE TFenGraphe.WMLButtonUp(VAR Msg: TMessage);
  BEGIN
   IF LButtonDown THEN WITH Msg DO
    BEGIN
     LButtonDown := False;
     ReleaseCapture;
    END;
  END;

{------------------ PROCEDURE TFenGraphe.WMMouseMove --------------------}
 PROCEDURE TFenGraphe.WMMouseMove(VAR Msg: TMessage);
  BEGIN
   IF LButtonDown THEN WITH Msg DO
    BEGIN
     ShowMouseLocation(LOWORD(LParam), HIWORD(LParam));
     WITH Msg DO
      BEGIN
       coordX2 := LParamLo;
       coordY2 := LParamHi;
      END;
    END;
  END;

{------------------DESTRUCTEUR TFenGraphe.Done -------------------------}
 DESTRUCTOR TFenGraphe.Done;
  BEGIN
   INHERITED Done;
  END;

{=================== PROCEDURES de TDessinFonctionCollect ================}

{------------------ CONSTRUCTEUR TDessinFonctionCollect.Init ------------}
 CONSTRUCTOR TDessinFonctionCollect.Init(ptgOWnerWindow:PFenGraphe);
  BEGIN
   INHERITED Init(20,20);
   ptOwnerWindow := ptgOwnerWindow;
  END;

{------------------ DESTRUCTEUR TDessinFonctionCollect.Done -------------}
 DESTRUCTOR TDessinFonctionCollect.Done;
  BEGIN
   INHERITED Done;
  END;

{------------------ PROCEDURE TDessinFonctionCollect.Reset --------------}
 PROCEDURE TDessinFonctionCollect.Reset;
  BEGIN
   FreeAll;
  END;

{------------------ PROCEDURE TDessinFonctionCollect.CouleurAxesInit ----}
 PROCEDURE TDessinFonctionCollect.CouleurAxesInit(aDC:HDC);
  BEGIN
   pen1:=createpen(ps_solid,1,RGB(0,0,0));
   pen2:=selectObject(aDC,pen1);
  END;
```

```
{------------ PROCEDURE TDessinFonctionCollect.CouleurAxesDone ---------}
 PROCEDURE TDessinFonctionCollect.CouleurAxesDone(aDC:HDC);
  BEGIN
   deleteobject(selectobject(aDC,pen2));
  END;

{------------ PROCEDURE TDessinFonctionCollect.CouleurCourbesInit ------}
 PROCEDURE TDessinFonctionCollect.CouleurCourbesInit(aDC:HDC);
  BEGIN
   pen1:=createpen(ps_solid,1,RGB(0,0,225));
   pen2:=selectObject(aDC,pen1);
  END;

{------------ PROCEDURE TDessinFonctionCollect.CouleurCourbesDone ------}
 PROCEDURE TDessinFonctionCollect.CouleurCourbesDone(aDC:HDC);
  BEGIN
   deleteobject(selectobject(aDC,pen2));
  END;

{------------ PROCEDURE TDessinFonctionCollect.CouleurProprietesInit ---}
 PROCEDURE TDessinFonctionCollect.CouleurProprietesInit(aDC:HDC);
  BEGIN
   pen1:=CreatePen(ps_solid,1,RGB(225,0,0));
   pen2:=selectObject(aDC,pen1);
   brush1 := CreateSolidBrush(RGB(225,0,0));
   brush2 := SelectObject(aDC,brush1);
  END;

{------------ PROCEDURE TDessinFonctionCollect.CouleurProprietesDone ---}
 PROCEDURE TDessinFonctionCollect.CouleurProprietesDone(aDC:HDC);
  BEGIN
   deleteobject(selectobject(aDC,pen2));
   deleteObject(SelectObject(aDC,brush2));
  END;

{------------ PROCEDURE TDessinFonctionCollect.DessinFonctionDroite ----}
 PROCEDURE TDessinFonctionCollect.DessinFonctionDroite(aDC:HDC);
  VAR
   a, b          : real;
   coeff         : real;
   bix, biy      : real;
   bsx, bsy      : real;
   ValEchX, ValEchY  : real;
   echx, echy    : real;
   i             : real;
   j             : integer;
   xp, yp        : longint;
   htemp         : integer;
  BEGIN
   { les axes } { en connaissant le domaine de définition }
   biy := (-10)+compteur3;
   bsy := (+10)+compteur4;
   bix := (-10)+compteur1;
   bsx := (+10)+compteur2;
   a := parametre1;
   b := parametre2;
   coeff := (-b)/a;
   ValEchX := GetSystemMetrics(sm_CXscreen)-143;
   htemp := GetSystemMetrics(sm_CYscreen);
   IF htemp = 600
```

```
  THEN
    ValEchY := HautEcranGraphe600
  ELSE
    BEGIN
     IF htemp = 480
       THEN
         ValEchY := HautEcranGraphe480;
    END;
echx := ValEchX/abs(bsx-bix);
echy := ValEchY/abs(bsy-biy);
CouleurAxesInit(aDC);
IF (0>biy) AND (0<bsy)
  THEN
    BEGIN
      MoveTo(aDC,0,round(bsy*echy));
      LineTo(aDC,round(ValEchX),round(bsy*echy));
      i := bix-1;
      REPEAT
        i := i+1;
        j := j+1;
        MoveTo(aDC,round(((i-bix)*echx)),round(bsy*echy));
        LineTo(aDC,round(((i-bix)*echx)),round(bsy*echy)+5);
      UNTIL i >= bsx;
    END
  ELSE
    BEGIN
      MoveTo(aDC,0,round(ValEchY)-10);
      LineTo(aDC,round(ValEchX),round(ValEchY)-10);
      i := bix-1;
      REPEAT
        i := i+1;
        MoveTo(aDC,round(((i-bix)*echx)),round(ValEchY)-10);
        LineTo(aDC,round(((i-bix)*echx)),round(ValEchY)-10+5);
      UNTIL i > biy;
    END;
IF (0>bix) AND (0<bsx)
  THEN
    BEGIN
      MoveTo(aDC,round(-bix*echx),0);
      LineTo(aDC,round(-bix*echx),round(ValEchY));
      i := biy-1;
      REPEAT
        i := i+1;
        MoveTo(aDC,round(-bix*echx),round((i+bsy)*echy));
        LineTo(aDC,round(-bix*echx)+5,round((i+bsy)*echy));
      UNTIL i >= bsy;
    END
  ELSE
    BEGIN
      MoveTo(aDC,10,0);
      LineTo(aDC,10,round(ValEchY));
    END;
CouleurAxesDone(aDC);
CouleurCourbesInit(aDC);
i := bix;
```

```
    REPEAT
      xp := round(((i-bix)*echx));
      yp := round((bsy-(a*i+b))*echy);
      IF ( yp < ValEchY) AND (yp > 0) THEN
       IF ( xp < ValEchX) AND (xp > 0) THEN
         BEGIN
           {Setpixel(aDC, xp, yp,RGB(0,0,225));}
           MoveTo(aDC,xp,yp);
           LineTo(aDC,xp,yp+1);
         END;
       i := i + 0.01;
    UNTIL i > bsx;
    CouleurCourbesDone(aDC);
    CouleurProprietesInit(aDC);
    Ellipse(aDC,round((coeff-bix)*echx)-2,round((bsy-0)*echy)-2,
          round((coeff-bix)*echx)+3,round((bsy-0)*echy)+3);
    Ellipse(aDC,round((0-bix)*echx)-2,round((bsy-b)*echy)-2,
          round((0-bix)*echx)+3,round((bsy-b)*echy)+3);
    CouleurProprietesDone(aDC);
  END;

{------------ PROCEDURE TDessinFonctionCollect.DessinFonctionParabole --}
 PROCEDURE TDessinFonctionCollect.DessinFonctionParabole(aDC:HDC);
  VAR
    a, b, c         : real;
    xmax, ymax      : real;
    ro,r1,r2        : real;
    px, py          : integer; { coordonnées nécessaires pour tracer le rectangle }
    x1, y1          : integer;
    ex, ey          : integer;
    m               : real;
    bix, biy        : real;
    bsx, bsy        : real;
    echx, echy      : real;
    i               : real;
    temp            : real;
    ValEchX, ValEchY : real;
    xp, yp          : longint;
    htemp           : integer;
  BEGIN
    ValEchX := GetSystemMetrics(sm_CXscreen)-143;
    htemp := GetSystemMetrics(sm_CYscreen);
    IF htemp = 600
     THEN
       ValEchY := HautEcranGraphe600
     ELSE
       BEGIN
         IF htemp = 480
          THEN
            ValEchY := HautEcranGraphe480;
       END;
    { les axes }
    biy := -10+compteur3;
    bsy := +10+compteur4;   { servent également dans les zooms }
    bix := -10+compteur1;
    bsx := +10+compteur2;
    a := parametre1;
    b := parametre2;
    c := parametre3;
    ro := b*b-4*a*c;
```

```
IF ro >= 0 THEN
 BEGIN
  r1 := (-b+sqrt(ro))/2/a;
  r2 := (-b-sqrt(ro))/2/a;
 END;
xmax := -b/2/a;
ymax := a*sqr(xmax)+b*xmax+c;
echx := ValEchX/abs(bsx-bix);
echy := ValEchY/abs(bsy-biy);
CouleurAxesInit(aDC);
IF (0>biy) AND (0<bsy)
 THEN
  BEGIN
   MoveTo(aDC,0,round(bsy*echy));
   LineTo(aDC,round(ValEchX),round(bsy*echy));
   i := bix-1;
   REPEAT
    i := i+1;
    MoveTo(aDC,round(((i-bix)*echx)),round(bsy*echy));
    LineTo(aDC,round(((i-bix)*echx)),round(bsy*echy)+5);
   UNTIL i >= bsx;
  END
 ELSE
  BEGIN
   MoveTo(aDC,0,round(ValEchY)-10);
   LineTo(aDC,round(ValEchX),round(ValEchY)-10);
   i := bix-1;
   REPEAT
    i := i +1;
    MoveTo(aDC,round(((i-bix)*echx)),round(ValEchY)-10);
    LineTo(aDC,round(((i-bix)*echx)),round(ValEchY)-10+5);
   UNTIL i > bsx;
  END;
IF (0>bix) AND (0<bsx)
 THEN
  BEGIN
   MoveTo(aDC,round(-bix*echx),0);
   LineTo(aDC,round(-bix*echx),round(ValEchY));
   i := biy-1;
   REPEAT
    i := i+1;
    MoveTo(aDC,round(-bix*echx),round((i+bsy)*echy));
    LineTo(aDC,round(-bix*echx)+5,round((i+bsy)*echy));
   UNTIL i >= bsy;
  END
 ELSE
  BEGIN
   MoveTo(aDC,10,0);
   LineTo(aDC,10,round(ValEchY));
  END;
CouleurAxesDone(aDC);
CouleurCourbesInit(aDC);
i := bix;
REPEAT
 xp := round(((i-bix)*echx));
 yp := round((bsy-(a*sqr(i)+b*i+c))*echy);
 IF ( yp < ValEchY) AND (yp > 0) THEN
  IF ( xp < ValEchX) AND (xp > 0) THEN
   BEGIN
    {Setpixel(aDC, xp, yp, RGB(0,0,225));}
```

```
        MoveTo(aDC,xp,yp);
        LineTo(aDC,xp,yp+1);
        {Setpixel(aDC,round(((2*xmax-i)-bix)*echx),yp,RGB(0,0,225));}
        MoveTo(aDC,round(((2*xmax-i)-bix)*echx),yp);
        LineTo(aDC,round(((2*xmax-i)-bix)*echx),yp+1);
       END;
    i := i + 0.001;
   UNTIL i > bsx;
   CouleurCourbesDone(aDC);
   CouleurProprietesInit(aDC);
   MoveTo(aDC,round((xmax-bix)*echx),round((bsy-ymax)*echy));
   LineTo(aDC,round((xmax-bix)*echx),round((bsy-ymax)*echy)+(round(ValEchY-((bsy-ymax)*echy))));
   MoveTo(aDC,round((xmax-bix)*echx),round((bsy-ymax)*echy));
   LineTo(aDC,round((xmax-bix)*echx),round((bsy-ymax)*echy)-(round(ValEchY-((bsy-ymax)*echy))));
   Ellipse(aDC,round((xmax-bix)*echx)-2,round((bsy-ymax)*echy)-2,
        round((xmax-bix)*echx)+3,round((bsy-ymax)*echy)+3);
   IF ro > 0
    THEN
     BEGIN
      Ellipse(aDC,round((r1-bix)*echx)-2,round((bsy-0)*echy)-2,
           round((r1-bix)*echx)+3,round((bsy-0)*echy)+3);
      Ellipse(aDC,round((r2-bix)*echx)-2,round((bsy-0)*echy)-2,
           round((r2-bix)*echx)+3,round((bsy-0)*echy)+3);
     END;
   IF ro = 0
    THEN
     BEGIN
      Ellipse(aDC,round((xmax-bix)*echx)-2,round((bsy-0)*echy)-2,
           round((xmax-bix)*echx)+3,round((bsy-0)*echy)+3);
     END;
   Ellipse(aDC,round((0-bix)*echx)-2,round((bsy-c)*echy)-2,
        round((0-bix)*echx)+3,round((bsy-c)*echy)+3);
   CouleurProprietesDone(aDC);
  END;

{------------ PROCEDURE TDessinFonctionCollect.DessinFonctionTrigono ---}
 PROCEDURE TDessinFonctionCollect.DessinFonctionTrigono(aDC:HDC);
  VAR
   a, b, k        : real;
   xmax           : real;
   ampli          : real;
   periode        : real;
   interX         : real;
   bix, biy       : real;
   bsx, bsy       : real;
   ValEchX, ValEchY : real;
   echx, echy     : real;
   i              : real;{compteur}
   xp, yp         : longint;{les coordonnées des points de la courbe à tracer}
   htemp          : integer;
  BEGIN
   {les valeurs des paramètres}
   a := parametre1;
   b := parametre2;
   k := parametre3;
   xmax := (pi/2-b)/a;
   ampli := round(k*sin(a*xmax+b));
   periode := 2*pi/abs(a);
```

```
{la zone délimitée pour tracer les axes}
{détermination du domaine de définition de la fonction}
biy := (-(abs(ampli)+5))+compteur3;{en augmentant la constante,}
bsy := (abs(ampli)+5)+compteur4;   {on applatit la courbe}
bix := -(periode)+compteur1;
bsx := +(periode)+compteur2;
{constantes contenant les dimensions de la fenêtre de dessin}
ValEchX := GetSystemMetrics(sm_CXscreen)-143;
htemp := GetSystemMetrics(sm_CYscreen);
IF htemp = 600
 THEN
  ValEchY := HautEcranGraphe600
 ELSE
  BEGIN
   IF htemp = 480
    THEN
     ValEchY := HautEcranGraphe480;
  END;
echx := ValEchX/abs(bsx-bix);
echy := ValEchY/abs(bsy-biy);
CouleurAxesInit(aDC);
IF (0>biy) AND (0<bsy)
 THEN
  BEGIN
   MoveTo(aDC,0,round(bsy*echy));   {au centre du cadre}
   LineTo(aDC,round(ValEchX),round(bsy*echy));
   i := bix-1;
   REPEAT
    i := i+1;
    {dessin des graduations et des motifs de graduation}
    MoveTo(aDC,round(((i-bix)*echx)),round(bsy*echy));
    LineTo(aDC,round(((i-bix)*echx)),round(bsy*echy)+5);
    {textOut(aDC,round(((i-bix)*echx)),round(bsy*echy)+8,s,1);}
   UNTIL i >= bsx;
  END
 ELSE
  BEGIN
   MoveTo(aDC,0,round(ValEchY)-10);     {dessin de l'axe des X}
   LineTo(aDC,round(ValEchX),round(ValEchY)-10);{en bas du cadre}
   i := bix-1;
   REPEAT
    i := i +1;
    {dessin des graduation et des motifs de graduation}
    MoveTo(aDC,round(((i-bix)*echx)),round(ValEchY)-10);
    LineTo(aDC,round(((i-bix)*echx)),round(ValEchY)-10+5);
   UNTIL i > bsx;
  END;
IF (0>bix) AND (0<bsx)
 THEN
  BEGIN
   MoveTo(aDC,round(-bix*echx),0);      {dessin de l'axe des Y}
   LineTo(aDC,round(-bix*echx),round(ValEchY));{au bord du cadre}
   i := biy-1;
   REPEAT
    i := i+1;
    MoveTo(aDC,round(-bix*echx),round((i+bsy)*echy));
    LineTo(aDC,round(-bix*echx)+5,round((i+bsy)*echy));
   UNTIL i >= bsy;
  END
 ELSE
```

```
      BEGIN
        MoveTo(aDC,10,0);              {dessin de l'axe des Y}
        LineTo(aDC,10,round(ValEchY));  {au centre du cadre}
      END;
    CouleurAxesDone(aDC);
    CouleurCourbesInit(aDC);
    i := bix; {le point de départ pour tracer la courbe}
    REPEAT
     xp := round(((i-bix)*echx));          {tracé de la fonction}
     yp := round((bsy-(k*(sin(a*i+b))))*echy);
     IF ( yp < ValEchY) AND (yp > 0) THEN
      IF ( xp < ValEchX) AND (xp > 0) THEN
       BEGIN
        {Setpixel(aDC, xp, yp, RGB(0,0,225));}
        MoveTo(aDC,xp,yp);
        LineTo(aDC,xp,yp+1);
       END;
     i := i + 0.001;
    UNTIL i > bsx;
    CouleurCourbesDone(aDC);
    CouleurProprietesInit(aDC);
    MoveTo(aDC,round((xmax-bix)*echx),round((bsy-ampli)*echy));
    LineTo(aDC,round((xmax-bix)*echx),round((bsy-ampli)*echy)+(round(ValEchY-((bsy-ampli)*echy))));
    MoveTo(aDC,round((xmax-bix)*echx),round((bsy-ampli)*echy));
    LineTo(aDC,round((xmax-bix)*echx),round((bsy-ampli)*echy)-(round(ValEchY-((bsy-ampli)*echy))));
    Ellipse(aDC,round((xmax-bix)*echx)-2,round((bsy-ampli)*echy)-2,
            round((xmax-bix)*echx)+3,round((bsy-ampli)*echy)+3);
    Ellipse(aDC,round((((pi-b)/a)-bix)*echx)-2,round((bsy-0)*echy)-2,
            round((((pi-b)/a)-bix)*echx)+3,round((bsy-0)*echy)+3);
    Ellipse(aDC,round((-2*((pi-b)/a)-bix)*echx)-2,round((bsy-0)*echy)-2,
            round((-2*((pi-b)/a)-bix)*echx)+3,round((bsy-0)*echy)+3);
    CouleurProprietesDone(aDC);
   END;

{------------------ PROCEDURE TDessinFonctionCollect.Paint ----------}
 PROCEDURE TDessinFonctionCollect.Paint(aDC:HDC);
  BEGIN
   CASE DroiteBool OF
    true :
     BEGIN
      {aDC := GetDC(ptOwnerWindow^.HWindow);}
      DessinFonctionDroite(aDC);
      {ReleaseDC(ptOwnerWindow^.HWindow,aDC);}
     END
   END;
   CASE ParaboleBool OF
    true :
     BEGIN
      {DC := GetDC(ptOwnerWindow^.HWindow);}
      DessinFonctionParabole(aDC);
      {ReleaseDC(ptOwnerWindow^.HWindow,aDC);}
     END
   END;
   CASE TrigonoBool OF
    true:
     BEGIN
      {aDC := GetDC(ptOwnerWindow^.HWindow);}
      DessinFonctionTrigono(aDC);
      {ReleaseDC(ptOwnerWindow^.HWindow,aDC);}
     END
```

```
    END;
    EstModifie := true;
  END;

{------------------ PROCEDURE TDessinFonctionCollect.Display ------------}
 PROCEDURE TDessinFonctionCollect.Display(paWindow:PFenGraphe);
  VAR
   aDC:HDC;
  BEGIN
   IF Count <> 0
    THEN
     BEGIN
      aDC := GetDC(paWindow^.HWindow);
      paint(aDC);
      ReleaseDC(paWindow^.HWindow,aDC);
     END;
  END;

{------------------ CONSTRUCTEUR TDessinFonctionCollect.Load ------------}
 CONSTRUCTOR TDessinFonctionCollect.Load (VAR s: TStream);
  BEGIN
   INHERITED Load(s);
  END;

{------------------ PROCEDURE TDessinFonctionCollect.Store --------------}
 PROCEDURE TDessinFonctionCollect.Store (VAR s: TStream);
  BEGIN
   INHERITED Store(s);
  END;

{*************************************************************************}
{***************************** Initialisation ****************************}
{*************************************************************************}

BEGIN
 RegisterType (RCollection);
 RegisterType (RDessinFonctionCollect);
END.
```

```
{#########################################################################}
{#                    Unité CalProp                                      #}
{#########################################################################}

UNIT CalProp; { Unité de la fenêtre des calculs de propriétes math. }

{*************************************************************************}
{***************************** Interface *********************************}
{*************************************************************************}

INTERFACE

USES
 WINTYPES, WINPROCS, OWINDOWS, STRINGS, WINDOS, OBJECTS, OSTDDLGS,
 VARGLOB;

TYPE
 PFenCalculsProp = ^TFenCalculsProp;
 TFenCalculsProp = OBJECT (TWindow)
  CalculsPropDC : HDC;
  CONSTRUCTOR Init(AParent: PWindowsObject; ATitle: PChar);
  PROCEDURE SetupWindow; VIRTUAL;
  PROCEDURE PropDroite;
  PROCEDURE PropParabole;
  PROCEDURE PropTrigonometrie;
  PROCEDURE Paint(PaintDC:HDC;VAR PaintInfo:TPaintStruct); VIRTUAL;
  DESTRUCTOR Done; VIRTUAL;
 END;

{*************************************************************************}
{*************************** Implementation ******************************}
{*************************************************************************}

IMPLEMENTATION

{==================== PROCEDURES de PFenCalculsProp ======================}

{----------------- Constructeur PFenCalculsProp.Init ------------------}
 CONSTRUCTOR TFenCalculsProp.Init(AParent: PWindowsObject; ATitle: PChar);
  VAR
   xtemp : integer;
   wtemp : integer;
  BEGIN
   INHERITED Init(AParent, ATitle);
   WITH attr DO
    BEGIN
     xtemp := GetSystemMetrics(sm_CXscreen);
     IF xtemp = 800
      THEN
       x := GetSystemMetrics(sm_CXscreen) - CoordXEcranCalcul800
      ELSE
       BEGIN
        IF xtemp = 640
         THEN
          x := GetSystemMetrics(sm_CXscreen) - CoordXEcranCalcul640;
       END;
     y := 2;
     wtemp := GetSystemMetrics(sm_CXscreen);
     IF wtemp = 800
      THEN
```

```
        w := LargEcranCalcul800
      ELSE
        BEGIN
          IF wtemp = 640
            THEN
              w := LargEcranCalcul640;
        END;
      h := 171;
      style := style OR ws_Child OR ws_Visible
            OR ws_ClipSiblings OR ws_SysMenu OR ws_OverLapped
            OR ws_Border OR ws_Caption;
    END;
  END;

{------------------ PROCEDURE TFenCalculsProp.SetupWindow ----------------}
 PROCEDURE TFenCalculsProp.SetupWindow;
  BEGIN
   INHERITED SetupWindow;
   UpdateWindow(HWindow);
   CalculsPropDC := GetDC(HWindow);
  END;

{------------------ PROCEDURE TFenCalculsProp.PropDroite ----------------}
 PROCEDURE TFenCalculsProp.PropDroite;
  VAR
   zero : integer;
   coefang : real;
   angX : real;
   interX : real;
   interY : real;
   Szero, Scoefang, SangX, SinterX, SinterY : string;
   Pcoefang, PangX, PinterX, PinterY : string;
   Tcoefang, TangX, TinterX, TinterY : string;
  BEGIN
   zero := 0;
   Str(zero:3, Szero);
   coefang := (-parametre2)/parametre1;
   Str(coefang:3, Scoefang);
   Tcoefang := Scoefang;
   angX := 0;
   Str(angX:3, SangX);
   TangX := SangX;
   interX := (-parametre2)/parametre1;
   Str(interX:3, SinterX);
   TinterX := '('+SinterX+';'+Szero+')';
   interY := parametre2;
   Str(interY:3, SinterY);
   TinterY := '('+Szero+';'+SinterY+')';
   Pcoefang := 'Coeff. angulaire: ';
   PangX := 'Angle avec X: ';
   PinterX := 'InterX: ';
   PinterY := 'InterY: ';
   {écriture des intitulés des propriétés}
   SetTextColor(CalculsPropDC,RGB(0,0,225));
   TextOut(CalculsPropDC,10,10,@Pcoefang[1],Length(Pcoefang));
   TextOut(CalculsPropDC,10,40,@PangX[1],Length(PangX));
   TextOut(CalculsPropDC,10,70,@PinterX[1],Length(PinterX));
   TextOut(CalculsPropDC,10,100,@PinterY[1],Length(PinterY));
   {écriture des valeurs des propriétés}
   SetTextColor(CalculsPropDC,RGB(48,159,48));
```

```
    TextOut(CalculsPropDC,130,10,@Tcoefang[1],Length(Tcoefang));
    TextOut(CalculsPropDC,110,40,@TangX[1],Length(TangX));
    TextOut(CalculsPropDC,60,70,@TinterX[1],Length(TinterX));
    TextOut(CalculsPropDC,60,100,@TinterY[1],Length(TinterY));
   END;

{------------------ PROCEDURE TFenCalculsProp.PropParabole ---------------}
 PROCEDURE TFenCalculsProp.PropParabole;
  VAR
   r0 : real; { le radiant }
   zero : integer;
   interX0, interX1, interX2 : real;
   interY : real;
   sommetX, sommetY : real;
   tg1pt : real;
   axesym : real;
   Szero, SinterX0, SinterX1, SinterX2 : string;
   SinterY, SsommetX, SsommetY, Stg1pt, Saxesym : string;
   PinterX, PinterY, Psommet, Ptg1pt, Paxesym : string;
   Tzero, TinterX0, TinterX1, TinterX2, TinterY, Tsommet : string;
   Ttg1pt, Taxesym : string;
  BEGIN
   zero := 0;
   Str(zero:0, Szero);
   r0 := (parametre2*parametre2)-(4*parametre1*parametre3);
   IF r0 > 0
    THEN
     BEGIN
      interX1 := (-(parametre2)+sqrt(r0))/(2*parametre1);
      interX2 := (-(parametre2)-sqrt(r0))/(2*parametre1);
     END;
   IF r0 = 0
    THEN
     BEGIN
      interX0 := -(parametre2)/(2*parametre1);
     END;
   Str(interX0:3, SinterX0);
   Str(interX1:3, SinterX1);
   Str(interX2:3, SinterX2);
   IF r0 > 0
    THEN
     BEGIN
      TinterX1 := '('+SinterX1+';'+Szero+')';
      TinterX2 := '('+SinterX2+';'+Szero+')';
     END;
   IF r0 = 0
    THEN
     BEGIN
      TinterX0 := '('+SinterX0+';'+Szero+')';
     END;
   IF r0 < 0
    THEN
     BEGIN
      TinterX0 := 'n"existe pas';
     END;
   interY := parametre3;
   Str(interY:3, SinterY);
   TinterY := '('+Szero+';'+SinterY+')';
   sommetX := -(parametre2)/(2*parametre1);
   Str(sommetX:3, SsommetX);
```

```
    sommetY := -((parametre2*parametre2)-(4*parametre1*parametre3))/(4*parametre1);
    Str(sommetY:3, SsommetY);
    Tsommet := '('+SsommetX+';'+SsommetY+')';
    tg1pt := 0;
    Str(tg1pt:3, Stg1pt);
    Ttg1pt := 'y = $(x-x0)+y0';
    axesym := -(parametre2)/(2*parametre1);
    Str(axesym:3, Saxesym);
    Taxesym := 'x := '+Saxesym;
    PinterX := 'InterX: ';
    PinterY := 'InterY: ';
    Psommet := 'Sommet: ';
    Ptg1pt := 'Tg1pt: ';
    Paxesym := 'Symétrie: ';
    {écriture des intitulés des propriétés}
    SetTextColor(CalculsPropDC,RGB(0,0,225));
    TextOut(CalculsPropDC,10,7,@PinterX[1],Length(PinterX));
    TextOut(CalculsPropDC,10,37,@PinterY[1],Length(PinterY));
    TextOut(CalculsPropDC,10,67,@Psommet[1],Length(Psommet));
    TextOut(CalculsPropDC,10,97,@Ptg1pt[1],Length(Ptg1pt));
    TextOut(CalculsPropDC,10,127,@Paxesym[1],Length(Paxesym));
    {écriture des valeurs des propriétés}
    SetTextColor(CalculsPropDC,RGB(48,159,48));
    IF r0 > 0
     THEN
      BEGIN
       TextOut(CalculsPropDC,70,7,@TinterX1[1],Length(TinterX1));
       TextOut(CalculsPropDC,150,7,@TinterX2[1],Length(TinterX2));
      END;
    IF r0 < 0
     THEN
      BEGIN
       TextOut(CalculsPropDC,70,7,@TinterX0[1],Length(TinterX0));
      END;
    IF r0 = 0
     THEN
      BEGIN
       TextOut(CalculsPropDC,70,7,@TinterX0[1],Length(TinterX0));
      END;
    TextOut(CalculsPropDC,70,37,@TinterY[1],Length(TinterY));
    TextOut(CalculsPropDC,70,67,@Tsommet[1],Length(Tsommet));
    TextOut(CalculsPropDC,70,97,@Ttg1pt[1],Length(Ttg1pt));
    TextOut(CalculsPropDC,90,127,@Taxesym[1],Length(Taxesym));
   END;

{------------------ PROCEDURE TFenCalculsProp.PropTrigonométrie ---------}
 PROCEDURE TFenCalculsProp.PropTrigonometrie;
  VAR
   periode : real;
   Pperiode, Pparite, Pzeros, Psommet, Paxesym : String;
   Speriode : string;
   Tperiode, Tparite, Tzeros, Tsommet, Taxesym : string;
  BEGIN
   periode := (2*pi)/abs(parametre1);
   Str(periode:3, Speriode);
   Tperiode := Speriode;
   Tparite := '';
   Tzeros := '';
   Tsommet := '';
   Taxesym := '';
```

```
    Pperiode := 'Période: ';
    Pparite := 'Parité: ';
    Pzeros := 'Zéros: ';
    Psommet := 'Extrema: ';
    Paxesym := 'AxeSym: ';
    SetTextColor(CalculsPropDC,RGB(0,0,225));
    TextOut(CalculsPropDC,10,7,@Pperiode[1],Length(Pperiode));
    TextOut(CalculsPropDC,10,37,@Pparite[1],Length(Pparite));
    TextOut(CalculsPropDC,10,67,@Pzeros[1],Length(Pzeros));
    TextOut(CalculsPropDC,10,97,@Psommet[1],Length(Psommet));
    textOut(CalculsPropDC,10,127,@Paxesym[1],Length(Paxesym));
    SetTextColor(CalculsPropDC,RGB(48,159,48));
    TextOut(CalculsPropDC,80,7,@Tperiode[1],Length(Tperiode));
    TextOut(CalculsPropDC,80,37,@Tparite[1],Length(Tparite));
    TextOut(CalculsPropDC,80,67,@Tzeros[1],Length(Tzeros));
    TextOut(CalculsPropDC,80,97,@Tsommet[1],Length(Tsommet));
    textOut(CalculsPropDC,80,127,@Taxesym[1],Length(Taxesym));
  END;

{------------------ PROCEDURE TFenCalculsProp.Paint ---------------------}
 PROCEDURE TFenCalculsProp.Paint(PaintDC:HDC;VAR PaintInfo:TPaintStruct);
  BEGIN
    CASE DroiteBool OF
     true :
       BEGIN
         PropDroite;
       END
    END;
    CASE ParaboleBool OF
     true :
       BEGIN
         PropParabole;
       END
    END;
    CASE TrigonoBool OF
     true:
       BEGIN
         PropTrigonometrie;
       END
    END;
  END;

{------------------ DESTRUCTEUR TFenCalculsProp.Done --------------------}
 DESTRUCTOR TFenCalculsProp.Done;
  BEGIN
    ReleaseDC(HWindow,CalculsPropDC);
    INHERITED Done;
  END;

{*************************************************************************}
{*************************** Initialisation ******************************}
{*************************************************************************}

BEGIN
END.
```

```
{#########################################################################}
{#                    Unité FPalOut                                      #}
{#########################################################################}

UNIT FPalOut;

{*************************************************************************}
{***************************** Interface *********************************}
{*************************************************************************}

INTERFACE

USES WINTYPES, WINPROCS, OWINDOWS, OBJECTS,
   PALOUTIL, VARGLOB;

TYPE
 PFenPaletteOutil = ^TFenPaletteOutil;
 TFenPaletteOutil = OBJECT (TPaletteOutil)
  CONSTRUCTOR Init(AParent:PWindow;ATitle:PChar;PalettePos:TPoint);
 END;

 PFenOutilPalette = ^TFenOutilPalette;
 TFenOutilPalette = OBJECT (TOutilPalette)
  vbool : boolean;
  PROCEDURE SetUpWindow; VIRTUAL;
  PROCEDURE WMDestroy (VAR Msg: TMessage);
   VIRTUAL wm_First + wm_Destroy;
  PROCEDURE BalayageBPalette;
  PROCEDURE WMTimer (VAR Msg: TMessage);
   VIRTUAl wm_First + wm_Timer;
  PROCEDURE WMLButtonDown (VAR Msg: TMessage);
   VIRTUAL wm_First + wm_LButtonDown;
 END;

{*************************************************************************}
{***************************** Implementation ****************************}
{*************************************************************************}

IMPLEMENTATION

{==================== PROCEDURE de TFenPaletteOutil ======================}

{----------------- Constructeur TFenPaletteOutil.Init -----------------}
 CONSTRUCTOR TFenPaletteOutil.Init(AParent:PWindow;ATitle:PChar;
                    PalettePos:TPoint);
  BEGIN
   INHERITED InitRempli(AParent,ATitle,PalettePos,ElemPaletteOutil);
  END;

{==================== PROCEDURE de TFenOutilPalette ======================}

{----------------- PROCEDURE TFenOutilPalette.SetUpWindow -------------}
 PROCEDURE TFenOutilPalette.SetUpwindow;
  BEGIN
   SetTimer(HWindow,Timer_ID,Timer_Intervalle,NIL);
  END;
```

```
{---------- PROCEDURE TFenOutilPalette.WMDestroy ----------}
 PROCEDURE TFenOutilPalette.WMDestroy (VAR Msg: TMessage);
  BEGIN
   KillTimer(HWindow,Timer_ID);
   INHERITED WMDestroy(Msg);
  END;

{---------- PROCEDURE TFenOutilPalette.BalayageBPalette ---------}
 PROCEDURE TFenOutilPalette.BalayageBPalette;
  VAR
   i : integer;
  BEGIN
   i := 1;
   WHILE i <= 20000 DO
    BEGIN
     select;
     i := i + 1
    END;
  END;

{---------- PROCEDURE TFenOutilPalette.WMTimer ----------}
 PROCEDURE TFenOutilPalette.WMTimer (VAR Msg: TMessage);
  BEGIN
   balayageBPalette;
   KillTimer(HWindow,Timer_ID);
  END;

{---------- PROCEDURE TFenOutilPalette.WMLButtonDown ----------}
 PROCEDURE TFenOutilPalette.WMLButtondown (VAR Msg: TMessage);
  BEGIN
   vbool := true;
   select;
  END;

{*****************************************************************************}
{************************** Initialisation ***********************************}
{*****************************************************************************}

BEGIN
 ElemPaletteOutil.Init(10,5);
END.
```

```
{#########################################################################}
{#                    Unité PalOutil                                     #}
{#########################################################################}

UNIT PalOutil;

{*************************************************************************}
{***************************** Interface *********************************}
{*************************************************************************}

INTERFACE

USES WINTYPES, WINPROCS, OBJECTS, OWINDOWS, ODIALOGS, BWCC,
   VARGLOB ;

TYPE
 TRectTaille = RECORD
             w:integer;
             h:integer;
            END;

CONST
 pal_Bouton3232    : TRectTaille = (w:32;h:32);
 pal_PosDefPalette  : TPoint = (x: -1;y: -1);
 pal_PosPaletteSuiv : TPoint = (x:-2;y: -2);

TYPE
 POutilPalette = ^TOutilPalette;
 PPaletteOutil = ^TPaletteOutil;
 PInitOutilPalette = ^TInitOutilPalette;

 TPaletteOutil = OBJECT (TWindow)
  CONSTRUCTOR InitVide(AParent:PWindowsObject;ATitle:PChar;PalettePos:TPoint);
  CONSTRUCTOR InitRempli(AParent:PWindowsObject;Atitle:PChar;
       PosPalette:tpoint;InitOutilCollection:TCollection);
  PROCEDURE GetWindowClass(VAR AWndClass:TWndClass); VIRTUAL;
  PROCEDURE Paint(aDC:HDC;VAR PaintInfo:TPaintStruct); VIRTUAL;
  FUNCTION pOutilSelectionne:POutilPalette;
  FUNCTION OutilSelectionneID:integer;
  FUNCTION OutilSelectionneIndex:integer;
  PROCEDURE SelectionnerOutil(pUnOutil:POutilPalette);
  PROCEDURE SelectionnerOutilID(IDOutil:integer);
  PROCEDURE SelectionnerOutilIndex(IndexOutil:integer);
  PRIVATE
   halfwidthfreef:boolean;
   pCurOutilSelectionne:POutilPalette;
   PROCEDURE aggrandir(hauteur:integer);
   PROCEDURE WMCommand(VAR Msg:TMessage);
    VIRTUAL WM_First + WM_Command;
 END;

 TInitOutilPaletteProc = PROCEDURE (AParent:PPaletteOutil);

 TInitOutilPalette = OBJECT(TObject)
  CONSTRUCTOR Init(gInitOutilPaletteProc:TInitOutilPaletteProc);
  PRIVATE
   procF:TInitOutilPaletteProc;
  END;

 TOutilPalette = OBJECT(TButton)
```

```
  CONSTRUCTOR Init(AParent: PPaletteOutil; ButtonID:Integer;
                   TailleBouton:TRectTaille; BParDefaut:Boolean);
  PROCEDURE Select;
 END;

{*************************************************************************}
{************************** Implementation ***************************}
{*************************************************************************}

IMPLEMENTATION

 CONST
  palettewidth         = 90;
  secondhalfwidthbuttonx = 72;
  firsthalfwidthbuttonx  = 20;
  fullwidthbuttonx       = 4;

 VAR
  lasthorizpos, lastvertpos : integer;

{------------------ PROCEDURE BoutonSelectionne ------------------------}
 PROCEDURE BoutonSelectionne
  (dlgHandle:HWnd;aDC:HDC;BoutonID:integer;selectSw:Boolean);
  VAR
   BoutonHandle:HWnd;
   BoutonRect,DlgRect,ClientRect:TRect;
   x0, y0, c1, c2, c3:integer;
   aPen,HoldPen:HPen;
  BEGIN
   c1 := 225;
   c2 := 225;
   c3 := 225;
   GetWindowRect(dlgHandle,DlgRect);
   GetClientRect(dlgHandle,ClientRect);
   BoutonHandle:= GetDlgItem (dlgHandle,BoutonID);
   x0:= DlgRect.left;
   y0:= DlgRect.bottom - ClientRect.bottom;
   GetWindowRect(BoutonHandle,boutonRect);
   IF selectSw
    THEN
     BEGIN
      c1 := 0;
      c2 := 0;
      c3 := 225;
     END
    ELSE
     BEGIN
      c1 := 255;
      c2 := 225;
      c3 := 225;
     END;
   aPen:= CreatePen(ps_solid,2,RGB(c1,c2,c3));
   WITH BoutonRect DO
    BEGIN
     left:=left-x0-3;
     right:=right-x0+2;
     top:=top-y0;
     bottom:=bottom-y0+3;
     holdPen:= selectobject(aDC,aPen);
     SetBKMode(aDC,Transparent);
```

```
      Rectangle(aDC,left,top,right,bottom);
    END;
   selectobject(aDC,holdPen);
   deleteobject(aPen);
  END;

{------------------ PROCEDURE SelectionnerBouton ------------------------}
 PROCEDURE SelectionnerBouton
  (dlghandle:HWnd;BoutonID:Integer;selectSw:boolean);
  VAR
   aDC:HDC;
  BEGIN
   aDC:=GetDC(dlghandle);
   BoutonSelectionne(dlghandle,aDC,BoutonID,SelectSw);
   ReleaseDC(dlghandle,aDC);
  END;

{=================== PROCEDURES de TPaletteOutil ============================}

{------------------ PROCEDURE TPaletteOutil.InitVide --------------------}
 CONSTRUCTOR TPaletteOutil.InitVide(AParent:PWindowsObject;ATitle:PChar;
                        PalettePos:TPoint);
  VAR
   ytemp : integer;
  BEGIN
   TWindow.Init(AParent,ATitle);
   WITH attr DO
    BEGIN
     style:= ws_Child  {OR ws_ClipChildren} OR ws_Border
          OR ws_Overlapped OR ws_ClipSiblings OR ws_Caption
          OR ws_Visible OR ws_SysMenu;
     CASE palettepos.x OF
      -1:  x:= GetSystemMetrics(sm_CXscreen)-136;
      -2:  x:= lasthorizpos;
      ELSE x:= palettepos.x;
     END;
     CASE palettepos.y OF
      -1:
       BEGIN
        ytemp := GetSystemMetrics(sm_CYscreen);
        IF ytemp = 600
         THEN
          y:= GetSystemMetrics(sm_CYscreen) - CoordYEcranGraphe600
         ELSE
          BEGIN
           IF ytemp = 480
            THEN
             y:= GetSystemMetrics(sm_CYscreen) - CoordYEcranGraphe480;
          END;
       END;
      -2:  y:= lastvertpos + 3;
      ELSE y:= palettepos.y;
     END;
     lasthorizpos := x;
     {x := GetSystemMetrics(sm_CXscreen)-160;}
     {y := GetSystemMetrics(sm_CYscreen)-380;}
     w := 124;
     h := 29;
    END;
   pCurOutilSelectionne := NIL;
```

```
    halfwidthfreef := false;
  END;

{------------------ PROCEDURE TPaletteOutil.InitRempli ------------------}
 CONSTRUCTOR TPaletteOutil.InitRempli(AParent:PWindowsObject;ATitle:PChar;
             PosPalette:TPoint;InitOutilCollection:TCollection);
  PROCEDURE InitUnOutil(PInitProcObj:PInitOutilPalette);
    BEGIN
      PInitProcObj^.procF(@self);
    END;
  BEGIN
    InitVide(AParent,ATitle,PosPalette);
    InitOutilCollection.ForEach(@InitUnOutil);
  END;

{------------------ PROCEDURE TPaletteOutil.GetWindowClass --------------}
 PROCEDURE TPaletteOutil.GetWindowClass(VAR AWndClass:TWndClass);
  BEGIN
    TWindow.GetWindowClass(AWndClass);
  END;

{------------------ PROCEDURE TPaletteOutil.Aggrandir -------------------}
 PROCEDURE TPaletteOutil.Aggrandir(hauteur:integer);
  BEGIN
    attr.h :=attr.h + hauteur;
  END;

{------------------ PROCEDURE TPaletteOutil.Paint -----------------------}
 PROCEDURE TPaletteOutil.Paint(aDC:HDC;VAR PaintInfo:TPaintStruct);
  BEGIN
    INHERITED Paint(aDC,PaintInfo);
    IF pCurOutilSelectionne <> NIL THEN
      BoutonSelectionne(HWindow,aDC,pCurOutilSelectionne^.GetID,true);
  END;

{------------------ Fonction TPaletteOutil.POutilSelectionne ------------}
 FUNCTION TPaletteOutil.POutilSelectionne:POutilPalette;
  BEGIN
    pOutilSelectionne:=pCurOutilSelectionne;
  END;

{------------------ Fonction TPaletteOutil.OutilSelectionneID -----------}
 FUNCTION TPaletteOutil.OutilSelectionneID:integer;
  BEGIN
    IF pCurOutilSelectionne <> NIL
      THEN OutilSelectionneID:=pCurOutilSelectionne^.GetID
      ELSE OutilSelectionneId:=0;
  END;

{------------------ Fonction TPaletteOutil.OutilSelectionneIndex --------}
 FUNCTION TPaletteOutil.OutilSelectionneIndex:integer;
  BEGIN
    IF pCurOutilSelectionne <> NIL
      THEN OutilSelectionneIndex:=IndexOf (pCurOutilSelectionne)
      ELSE OutilSelectionneIndex:=0;
   END;

{------------------ PROCEDURE TPaletteOutil.SelectionnerOutil -----------}
 PROCEDURE TPaletteOutil.SelectionnerOutil(pUnOutil:POutilPalette);
  BEGIN
```

```
    pCurOutilSelectionne:=pUnOutil;
    InvalidateRect(HWindow,NIL,true);
  END;

{---------- PROCEDURE TPaletteOutil.SelectionnerOutilID ---------}
 PROCEDURE TPaletteOutil.SelectionnerOutilID(IDOutil:integer);
  VAR pUnOutil:POutilPalette;
  BEGIN
   IF IDOutil <> 0
     THEN pUnOutil := POutilPalette(childWithID(IDOutil))
     ELSE pUnOutil := NIL;
   SelectionnerOutil(pUnOutil);
  END;

{---------- PROCEDURE TPaletteOutil.SelectionnerOutilIndex ------}
 PROCEDURE TPaletteOutil.SelectionnerOutilIndex(IndexOutil:integer);
  BEGIN
   SelectionnerOutil(PoutilPalette(at(IndexOutil)));
  END;

{---------- PROCEDURE TPaletteOutil.WMCommand ----------}
 PROCEDURE TPaletteOutil.WMCommand (VAR Msg:TMessage);
  BEGIN
   SelectionnerOutilID(Msg.WParam);
  END;

{========== PROCEDURE de TInitOutilPalette ==========}

{---------- CONSTRUCTEUR TInitOutilPalette.Init ----------}
 CONSTRUCTOR TInitOutilPalette.Init
        (gInitOutilPaletteProc:TInitOutilPaletteProc);
  BEGIN
   INHERITED Init;
   procF:=gInitOutilPaletteProc;
  END;

{========== PROCEDURES de TOutilPalette ==========}

{---------- CONSTRUCTEUR TOutilPalette.Init ----------}
 CONSTRUCTOR TOutilPalette.Init(AParent:PPaletteOutil;
                    ButtonID:integer;
                    TailleBouton:TRectTaille;
                    BParDefaut:Boolean);
  VAR
   x,y:integer;
  BEGIN
   IF aparent^.halfwidthfreef AND (taillebouton.w = 32)
    THEN
     BEGIN
      x := secondhalfwidthbuttonx;
      aparent^.halfwidthfreef := false;
     END
    ELSE
     BEGIN
      AParent^.Aggrandir(TailleBouton.h + 10);
      IF taillebouton.w = 32
       THEN
        BEGIN
         x := firsthalfwidthbuttonx;
         aparent^.halfwidthfreef := true;
```

```
          END
        ELSE x := fullwidthbuttonx;
      END;
    y:=aparent^.attr.h - taillebouton.h - 30;
    WITH TailleBouton DO
     INHERITED Init(AParent,ButtonID,'',x,y,w,h,BParDefaut);
    attr.style:=attr.style OR ws_Child OR ws_Visible {OR ws_Border};
    IF BParDefaut THEN
     BEGIN
      select;
     END;
    lastvertpos := aparent^.attr.y + aparent^.attr.h;
  END;

{------------------ PROCEDURE TOutilPalette.Select ----------------------}
 PROCEDURE TOutilPalette.Select;
  BEGIN
   PPaletteOutil(Parent)^.SelectionnerOutilID(GetID);
  END;

{*************************************************************************}
{***************************** Initialisation ****************************}
{*************************************************************************}

BEGIN
END.
```

```
{##############################################################################}
{#                    Unité FExplic                                           #}
{##############################################################################}

UNIT FExplic; { Unité de la fenêtre des explications des actions }

{******************************************************************************}
{***************************** Interface **************************************}
{******************************************************************************}

INTERFACE

USES
 WINTYPES, WINPROCS, OWINDOWS, OBJECTS, WINDOS;

TYPE
 PFenExplications = ^TFenExplications;
 TFenExplications = OBJECT (TWindow)
  CONSTRUCTOR Init(AParent: PWindowsObject; ATitle: PChar);
  DESTRUCTOR Done; VIRTUAL;
 END;

{******************************************************************************}
{***************************** Implementation *********************************}
{******************************************************************************}

IMPLEMENTATION

{=================== PROCEDURES de TFenExplications =======================}

{------------------ Constructeur TFenExplications.Init ------------------}
 CONSTRUCTOR TFenExplications.Init(AParent: PWindowsObject; ATitle: PChar);
  BEGIN
   INHERITED Init(AParent, ATitle);
   WITH attr DO
    BEGIN
     x := 0;
     y := GetSystemMetrics(sm_CYscreen)-65;
     w := GetSystemMetrics(sm_CXscreen);
     h := 60;
     style := style OR ws_Child OR ws_Visible OR ws_Border
           OR ws_ClipSiblings;
    END;
  END;

{------------------ DESTRUCTEUR TFenExplications.Done -------------------}
 DESTRUCTOR TFenExplications.Done;
  BEGIN
   INHERITED Done;
  END;

{******************************************************************************}
{***************************** Initialisation *********************************}
{******************************************************************************}

BEGIN
END.
```

```
{#########################################################################}
{#                    Unité Balayage                                     #}
{#########################################################################}

UNIT Balayage; {Unité de la boîte de dialogue permettant de choisir }
          { la vitesse de balayage des fenêtre de l'applicatoion }

{$R dbalay.res}

{*************************************************************************}
{************************** Interface ************************************}
{*************************************************************************}

INTERFACE

USES WINTYPES, WINPROCS, OBJECTS, OWINDOWS, ODIALOGS, STRINGS, BWCC;

{$I dbalay.inc}

CONST
 valeur_balayage_defaut = 1000;

TYPE
 PDiaBalayage = ^TDiaBalayage;
 TDiaBalayage = OBJECT (TDialog)
   Edit_pal_act  : PEdit;
   Edit_boite_dial : PEdit;
   Edit_boite_mes  : PEdit;
   BarDef_pal_act : PScrollBar;
   BarDef_boite_dial : PScrollBar;
   BarDef_boite_mes : PScrollBar;
   CONSTRUCTOR Init(AParent:PWindowsObject;ATitle:PChar);
   PROCEDURE SetupWindow; VIRTUAL;
   DESTRUCTOR Done ; VIRTUAL;
   PROCEDURE IDHelp (VAR Msg: TMessage);
     VIRTUAL id_First + id_Help;
   PROCEDURE ScrollBar_pal_act(VAR Msg: TMessage);
     VIRTUAL id_First + id_bardef_pal_act;
   PROCEDURE ScrollBar_boite_dial(VAR Msg: TMessage);
     VIRTUAL id_First + id_bardef_boite_dial;
   PROCEDURE ScrollBar_boite_mes(VAR Msg: TMessage);
     VIRTUAL id_First + id_bardef_boite_mes;
 END;

{*************************************************************************}
{************************** Implementation *******************************}
{*************************************************************************}

IMPLEMENTATION

{==================== PROCEDURES de TDiaBalayage ========================}

{----------------- CONSTRUCTEUR TDiaBalayage.Init ---------------------}
 CONSTRUCTOR TDiaBalayage.Init(AParent:PWindowsObject;Atitle:PChar);
   BEGIN
     TDialog.Init(AParent,ATitle);
     new(Edit_pal_act,InitResource(PDiaBalayage(@self),id_edit_pal_act,20));
     new(Edit_boite_dial,InitResource(PDiaBalayage(@self),id_edit_boite_dial,20));
     new(Edit_boite_mes,InitResource(PDiaBalayage(@self),id_edit_boite_mes,20));
     new(BarDef_pal_act,InitResource(PDiaBalayage(@Self),id_bardef_pal_act));
```

```
    new(BarDef_boite_dial,InitResource(PDiaBalayage(@self),id_bardef_boite_dial));
    new(BarDef_boite_mes,InitResource(PDiaBalayage(@self),id_bardef_boite_mes));
  END;

{------------------ PROCEDURE TDiaBalayage.SetupWindow ------------------}
 PROCEDURE TDiaBalayage.SetupWindow;
  VAR
   aa : ARRAY [0..10] OF Char;
   aa1: ARRAY [0..10] OF Char;
   bb : ARRAY [0..10] OF Char;
   cc : ARRAY [0..10] OF Char;
  BEGIN
   INHERITED SetupWindow;
   bardef_pal_act^.SetPosition(valeur_balayage_defaut);
   bardef_boite_dial^.SetPosition(valeur_balayage_defaut);
   bardef_boite_mes^.SetPosition(valeur_balayage_defaut);
   bardef_pal_act^.SetRange(1000,10000);
   bardef_boite_dial^.SetRange(1000,10000);
   bardef_boite_mes^.SetRange(1000,10000);
   bardef_pal_act^.LineMagnitude := 1000;
   bardef_boite_dial^.LineMagnitude := 1000;
   bardef_boite_mes^.LineMagnitude := 1000;
   str(valeur_balayage_defaut,aa);
   Edit_pal_act^.SetText(aa);
   str(valeur_balayage_defaut,bb);
   Edit_boite_dial^.SetText(bb);
   str(valeur_balayage_defaut,cc);
   Edit_boite_mes^.SetText(cc);
  END;

{------------------ PROCEDURE TDiaBalayage.ScrollBar_pal_act ------------}
 PROCEDURE TDiaBalayage.ScrollBar_pal_act (VAR Msg:TMessage);
  VAR
   CString1: ARRAY [0..10] OF char;
  BEGIN
   Str(bardef_pal_act^.GetPosition,CString1);
   Edit_pal_act^.SetText(CString1);
  END;

{------------------ PROCEDURE TDiaBalayage.ScrollBar_boite_dial ---------}
 PROCEDURE TDiaBalayage.ScrollBar_boite_dial(VAR Msg:TMessage);
  VAR
   CString2: ARRAY [0..10] OF Char;
  BEGIN
   Str(bardef_boite_dial^.GetPosition,CString2);
   Edit_boite_dial^.SetText(CString2);
 END;

{------------------ PROCEDURE TDiaBalayage.ScrollBar_boite_mes ----------}
 PROCEDURE TDiaBalayage.ScrollBar_boite_mes(VAR Msg:TMessage);
  VAR
   CString3: ARRAY [0..10] OF char;
  BEGIN
   Str(bardef_boite_mes^.GetPosition,CString3);
   Edit_boite_mes^.SetText(CString3);
  END;

{------------------ PROCEDURE TDiaBalayage.IDHelp ----------------------}
 PROCEDURE TDiaBalayage.IDHelp (VAR Msg: TMessage);
  BEGIN
```

```
    MessageBox(HWindow,
            'Cette fonctionnalité n''a pas encore été implémentée',
            'Message d''information',mb_Ok OR mb_IconExclamation);
  END;

{------------------ DESTRUCTEUR TDiaBalayage.Done ----------------------}
 DESTRUCTOR TDiaBalayage.Done;
  BEGIN
   INHERITED Done;
  END;

{*************************************************************************}
{***************************** Initialisation ****************************}
{*************************************************************************}

BEGIN
END.
```

```
{##############################################################################}
{#                    Unité Nouv                                              #}
{##############################################################################}

UNIT Nouv; { Unité de l'outil Nouvel Exercice de la palette d'actions }

{$R nouv.res}

{******************************************************************************}
{***************************** Interface **************************************}
{******************************************************************************}

INTERFACE

USES WINTYPES, WINPROCS, OBJECTS, OWINDOWS, ODIALOGS,
   PALOUTIL, FPALOUT, DEQUAT, VARPOINT, VARGLOB;

{$I nouv.inc}

TYPE
 POutilNouveau = ^TOutilNouveau;
 TOutilNouveau = OBJECT (TFenOutilPalette)
  PROCEDURE AfficherExplication;
  PROCEDURE WMLButtonDown (VAR Msg: TMessage);
    VIRTUAL wm_First + wm_LButtonDown;
  PROCEDURE WMLButtonUp ( VAR Msg: TMessage);
    VIRTUAL wm_First + wm_LButtonUp;
 END;

{******************************************************************************}
{***************************** Implementation *********************************}
{******************************************************************************}

IMPLEMENTATION

 VAR
  PLeOutilNouveau : POutilNouveau;

{----------------- PROCEDURE AJoutOutilNouveau ------------------------}
 PROCEDURE AJoutOutilNouveau (AParent: PPaletteOutil);
  BEGIN
   New(PLeOutilNouveau,
      init(AParent,id_outil_Nouveau+id_pal_Nouveau,pal_Bouton3232,false));
  END;

{==================== PROCEDURES de TOutilNouveau ========================}

{----------------- PROCEDURE TOutilNouveau.AfficherExplication ---------}
 PROCEDURE TOutilNouveau.AfficherExplication;
  VAR
   Commentaire : String;
  BEGIN
   Commentaire := 'Permet de commencer un nouvel exercice';
   CommtDC := GetDC(ptFenCommentaire^.HWindow);
   SetTextColor(CommtDC,RGB(225,0,0));
   TextOut(CommtDC,2,2,@Commentaire[1],Length(Commentaire));
   ReleaseDC(ptFenCommentaire^.HWindow,CommtDC);
  END;

{----------------- PROCEDURE TOutilNouveau.WMLButtonDown ----------------}
```

```
 PROCEDURE TOutilNouveau.WMLButtonDown (VAR Msg: TMessage);
  BEGIN
   Select;
   AfficherExplication;
   IF EstModifie THEN
    BEGIN
     Application^.ExecDialog(New(PDiaMessageQuitter,
      Init(@Self, PChar(id_dial_mes_quitter))));
    END;
   Application^.ExecDialog(New(PDiaFamFonc,
    Init(@Self, PChar(id_dial_equations))));
  END;

{----------------- PROCEDURE ToutilNouveau.WMLButtonUp ----------------}
 PROCEDURE TOutilNouveau.WMLButtonUp ( VAR Msg: TMessage);
  BEGIN
   InvalidateRect(ptFenCommentaire^.HWindow,NIL,true);
   UpdateWindow(ptFenCommentaire^.HWindow);
  END;

{*************************************************************************}
{************************** Initialisation ******************************}
{*************************************************************************}

BEGIN
 ElemPaletteOutil.insert(New(PInitOutilPalette,init(AJoutOutilNouveau)));
END.
```

```
{#########################################################################}
{#                    Unité Ouvrir                                       #}
{#########################################################################}

UNIT Ouvrir; { Unité de l'outil Ouvrir un exercice existant }
        { de la palette d'actions }

{$R ouvrir.res}
{$R douvfich.res}

{*************************************************************************}
{**************************** Interface **********************************}
{*************************************************************************}

INTERFACE

USES WINTYPES, WINPROCS, OBJECTS, OWINDOWS,
   WINDOS, OSTDDLGS, ODIALOGS, STRINGS,
   PALOUTIL, FPALOUT, VARPOINT, VARGLOB;

{$I ouvrir.inc}
{$I douvfich.inc}

TYPE
 POutilOuvrir = ^TOutilOuvrir;
 TOutilOuvrir = OBJECT (TFenOutilPalette)
  PROCEDURE AfficherExplication;
  PROCEDURE WMLButtonDown ( VAR Msg: TMessage);
   VIRTUAL wm_First + wm_LButtonDown;
  PROCEDURE WMLButtonUp ( VAR Msg: TMessage);
   VIRTUAL wm_First + wm_LButtonUp;
 END;

 PDiaOuvrirFichier = ^TDiaOuvrirFichier;
 TDiaOuvrirFichier = OBJECT (TDialog)
  PROCEDURE IDHelp (VAR Msg: TMessage);
   VIRTUAL id_First + id_Help;
 END;

{*************************************************************************}
{************************** Implementation ******************************}
{*************************************************************************}

IMPLEMENTATION

 VAR
  PLeOutilOuvrir : POutilOuvrir;

{----------------- PROCEDURE AJoutOutilOuvrir -------------------------}
 PROCEDURE AJoutOutilOuvrir (AParent: PPaletteOutil);
  BEGIN
   New(PLeOutilOuvrir,
     init(AParent,id_outil_Ouvrir+id_pal_Ouvrir,pal_Bouton3232,false));
  END;

{================== PROCEDURES de TOutilOuvrir ------------------------}

{----------------- PROCEDURE TOutilOuvrir.AfficherExplication ----------}
 PROCEDURE TOutilOuvrir.AfficherExplication;
  VAR
```

```
    Commentaire : String;
  BEGIN
    Commentaire := 'Ouvre un fichier d''exercice existant';
    CommtDC := GetDC(ptFenCommentaire^.HWindow);
    SetTextColor(CommtDC,RGB(225,0,0));
    TextOut(CommtDC,2,2,@Commentaire[1],Length(Commentaire));
    ReleaseDC(ptFenCommentaire^.HWindow,CommtDC);
  END;

{------------------ PROCEDURE TOutilOuvrir.WMLButtonDown ----------------}
 PROCEDURE TOutilOuvrir.WMLButtonDown (VAR Msg: TMessage);
  BEGIN
    Select;
    AfficherExplication;
    IF EstModifie THEN
     BEGIN
       Application^.ExecDialog(New(PDiaMessageQuitter,
        Init(@Self, PChar(id_dial_mes_quitter))));
     END;
    Application^.ExecDialog(New(PDiaOuvrirFichier,
     Init(@Self, PChar(id_dial_ouvrir_fichier))));
  END;

{------------------ PROCEDURE TOutilOuvrir.WMLButtonUp ------------------}
 PROCEDURE TOutilOuvrir.WMLButtonUp ( VAR Msg: TMessage);
  BEGIN
    InvalidateRect(ptFenCommentaire^.HWindow,NIL,true);
    UpdateWindow(ptFenCommentaire^.HWindow);
  END;

{==================== PROCEDURES de TDiaOuvrirFichier ======================}

{------------------ PROCEDURE TDiaOuvrirFichier.IDHelp ------------------}
 PROCEDURE TDiaOuvrirFichier.IDHelp (VAR Msg: TMessage);
  BEGIN
    MessageBox(HWindow,
          'Cette fonctionnalité n''a pas encore été implémentéé',
          'Message d''information',mb_Ok OR mb_IconExclamation);
  END;

{*****************************************************************************}
{***************************** Initialisation ********************************}
{*****************************************************************************}

BEGIN
 ElemPaletteOutil.insert(New(PInitOutilPalette,init(AJoutOutilOuvrir)));
END.
```

```
{##############################################################################}
{#                    Unité Sauver                                            #}
{##############################################################################}

UNIT Sauver; { Unité de l'outil Sauver de la palette d'actions }

{$R sauver.res}

{******************************************************************************}
{***************************** Interface **************************************}
{******************************************************************************}

INTERFACE

USES WINTYPES, WINPROCS, OBJECTS, OWINDOWS,
   OSTDDLGS, ODIALOGS, STRINGS, WINDOS,
   PALOUTIL, FPALOUT, VARPOINT, VARGLOB;

{$I Sauver.inc}

TYPE
 POutilSauver = ^TOutilSauver;
 TOutilSauver = OBJECT (TFenOutilPalette)
  PROCEDURE AfficherExplication;
  PROCEDURE WMLButtonDown ( VAR Msg: TMessage);
   VIRTUAL wm_First + wm_LButtonDown;
  PROCEDURE WMLButtonUp ( VAR Msg: TMessage);
   VIRTUAL wm_First + wm_LButtonUp;
 END;

{******************************************************************************}
{************************** Implementation ************************************}
{******************************************************************************}

IMPLEMENTATION

 VAR
  PLeOutilSauver : POutilSauver;

{------------------ PROCEDURE AJoutOutilSauver -------------------------}
 PROCEDURE AJoutOutilSauver( AParent: PPaletteOutil);
  BEGIN
   New(PLeOutilSauver,
    init(AParent,id_outil_sauver+id_pal_sauver,pal_Bouton3232,false));
  END;

{==================== PROCEDURES de TOutilSauver ============================}

{------------------ PROCEDURE TOutilSauver.AfficherExplication ----------}
 PROCEDURE TOutilSauver.AfficherExplication;
  VAR
   Commentaire : String;
  BEGIN
   Commentaire := 'Sauve l''exercice en cours dans un fichier au choix';
   CommtDC := GetDC(ptFenCommentaire^.HWindow);
   SetTextColor(CommtDC,RGB(225,0,0));
   TextOut(CommtDC,2,2,@Commentaire[1],Length(Commentaire));
   ReleaseDC(ptFenCommentaire^.HWindow,CommtDC);
  END;
```

```
{---------- PROCEDURE TOutilSauver.WMLButtonDown ----------}
 PROCEDURE TOutilSauver.WMLButtonDown ( var Msg:TMessage);
  BEGIN
   Select;
   AfficherExplication;
   Application^.ExecDialog(New(PDiaSauverFichier,
    Init(@Self, PChar(id_sixieme_menu_prog))));
  END;

{---------- PROCEDURE TOutilSauver.WMLButtonUp ----------}
 PROCEDURE TOutilSauver.WMLButtonUp ( VAR Msg: TMessage);
  BEGIN
   InvalidateRect(ptFenCommentaire^.HWindow,NIL,true);
   UpdateWindow(ptFenCommentaire^.HWindow);
  END;

{*************************************************************************}
{************************** Initialisation *******************************}
{*************************************************************************}

BEGIN
 ElemPaletteOutil.insert(New(PInitOutilPalette,init(AJoutOutilSauver)));
END.
```

```
{#########################################################################}
{#                    Unité Quitter                                      #}
{#########################################################################}

UNIT Quitter; { Unité de l'outil Quitter de la palette d'actions }
        { Elle aura pour but de laisser à l'élève de }
        { quitter le programme à l'aide du balayage }

{$R quitter.res}
{$R dquitprg.res}

{*************************************************************************}
{***************************** Interface *********************************}
{*************************************************************************}

INTERFACE

USES WINTYPES, WINPROCS, OBJECTS, OWINDOWS, WINDOS, ODIALOGS,
   PALOUTIL, FPALOUT, VARPOINT, VARGLOB;

{$I quitter.inc}
{$I dquitprg.inc}

TYPE
 POutilTerminer = ^TOutilTerminer;
 TOutilTerminer = OBJECT (TFenOutilPalette)
  PROCEDURE AfficherExplication;
  PROCEDURE WMLButtonDown (VAR Msg:TMessage);
   VIRTUAL wm_First + wm_LButtonDown;
  PROCEDURE WMLButtonUp ( VAR Msg: TMessage);
   VIRTUAL wm_First + wm_LButtonUp;
 END;

 PDiaQuitterProgramme = ^TDiaQuitterProgramme;
 TDiaQuitterProgramme = OBJECT (TDialog)
  PROCEDURE MesQuitBoutonOui (VAR Msg: TMessage);
   VIRTUAL id_First + id_menu_bouton_oui;
  PROCEDURE MesQuitBoutonNon (VAR Msg: TMessage);
   VIRTUAL id_First + id_menu_bouton_non;
  PROCEDURE MesQuitBoutonAide (VAR Msg: TMessage);
   VIRTUAL id_First + id_menu_bouton_aide;
 END;

{*************************************************************************}
{************************** Implementation *******************************}
{*************************************************************************}

IMPLEMENTATION

VAR
  PLeOutilTerminer: POutilTerminer;

{---------- PROCEDURE AJoutOutilTerminer ----------}
 PROCEDURE AJoutOutilTerminer( AParent: PPaletteOutil);
  BEGIN
   New(PLeOutilTerminer,
    init(AParent,id_outil_quitter+id_pal_quitter,pal_Bouton3232,false));
  END;

{================== PROCEDURES de TOutilTerminer ========================}
```

```
{---------- PROCEDURE TOutilTerminer.AfficherExplication --------}
 PROCEDURE TOutilTerminer.AfficherExplication;
  VAR
   Commentaire : String;
  BEGIN
   Commentaire := 'Permet de terminer l''exercice et de quitter le programme';
   CommtDC := GetDC(ptFenCommentaire^.HWindow);
   SetTextColor(CommtDC,RGB(225,0,0));
   TextOut(CommtDC,2,2,@Commentaire[1],Length(Commentaire));
   ReleaseDC(ptFenCommentaire^.HWindow,CommtDC);
  END;

{---------- PROCEDURE TOutilTerminer.WMLButtonDown ----------}
 PROCEDURE TOutilTerminer.WMLButtonDown (VAR Msg:TMessage);
  BEGIN
   Select;
   AfficherExplication;
   IF EstModifie THEN
    BEGIN
     Application^.ExecDialog(New(PDiaMessageQuitter,
      Init(@Self, PChar(id_dial_mes_quitter))));
    END;
   Application^.ExecDialog(New(PDiaQuitterProgramme,
    Init(@Self, PChar(id_dial_quitter_programme))));
  END;

{---------- PROCEDURE TOutilTerminer.WMLButtonUp ----------}
 PROCEDURE TOutilTerminer.WMLButtonUp ( VAR Msg: TMessage);
  BEGIN
   InvalidateRect(ptFenCommentaire^.HWindow,NIL,true);
   UpdateWindow(ptFenCommentaire^.HWindow);
  END;

{================== PROCEDURES de TDiaQuitterProgramme ==================}

{---------- PROCEDURE TDiaQuitterProgramme.MesQuitBoutonOui -----}
 PROCEDURE TDiaQuitterProgramme.MesQuitBoutonOui (VAR Msg: TMessage);
  BEGIN
   TDialog.EndDlg(0);
  END;

{---------- PROCEDURE TDiaQuitterProgramme.MesQuitBoutonNon -----}
 PROCEDURE TDiaQuitterProgramme.MesQuitBoutonNon (VAR Msg: TMessage);
  BEGIN
   TDialog.Cancel(Msg);
  END;

{---------- PROCEDURE TDiaQuitterProgramme.MesQuitBoutonAide ----}
 PROCEDURE TDiaQuitterProgramme.MesQuitBoutonAide (VAR Msg: TMessage);
  BEGIN
   MessageBox(HWindow,
        'Cette fonctionnalité n''a pas encore été implémentée',
        'Message d''information',mb_OK Or mb_IconExclamation);
  END;

{*************************************************************************}
{************************** Initialisation ******************************}
{*************************************************************************}
```

```
BEGIN
  ElemPaletteOutil.insert(New(PInitOutilPalette,init(AJoutOutilTerminer)));
END.
```

```
{##############################################################################}
{#                    Unité Param                                             #}
{##############################################################################}

UNIT Param; { Unité de l'outil Paramètres de la palette d'actions }
        { Cet outil permet de déclencher le balayage de la }
        { de la fenêtre des paramètre }

{$R param.res}
{$R dbalpar.res}

{******************************************************************************}
{***************************** Interface **************************************}
{******************************************************************************}

INTERFACE

USES WINTYPES, WINPROCS, OBJECTS, OWINDOWS, ODIALOGS, STRINGS,
   PALOUTIL, FPALOUT, VARPOINT, VARGLOB;

{$I param.inc}
{$I dbalpar.inc}

TYPE
 POutilParametres = ^TOutilParametres;
 TOutilParametres = OBJECT (TFenOutilPalette)
  PROCEDURE AfficherExplication;
  PROCEDURE WMLButtonDown (VAR Msg:TMessage);
   VIRTUAL wm_First + wm_LButtonDown;
  PROCEDURE WMLButtonUp ( VAR Msg: TMessage);
   VIRTUAL wm_First + wm_LButtonUp;
 END;

 PDiaSelectionBalParam = ^TDiaSelectionBalParam;
 TDiaSelectionBalParam = OBJECT (TDialog)
  BoutonAPositif : PButton;
  BoutonANegatif : Pbutton;
  BoutonBPositif : PButton;
  BoutonBNegatif : Pbutton;
  BoutonCPositif : PButton;
  BoutonCNegatif : Pbutton;
  LButtonDown : Boolean;
  CONSTRUCTOR Init (AParent:PWindowsObject;ATitle:PChar);
  PROCEDURE SetUpWindow; VIRTUAL;
  PROCEDURE BAPos (VAR Msg: TMessage);
   VIRTUAL id_First + bouton_a_positif;
  PROCEDURE BANeg (VAR Msg: TMessage);
   VIRTUAL id_First + bouton_a_negatif;
  PROCEDURE BBPos (VAR Msg: TMessage);
   VIRTUAL id_First + bouton_b_positif;
  PROCEDURE BBNeg (VAR Msg: TMessage);
   VIRTUAL id_First + bouton_b_negatif;
  PROCEDURE BCPos (VAR Msg: TMessage);
   VIRTUAL id_First + bouton_c_positif;
  PROCEDURE BCNeg (VAR Msg: TMessage);
   VIRTUAL id_First + bouton_c_negatif;
  PROCEDURE WMDestroy (VAR Msg: TMessage);
   VIRTUAL wm_First + wm_Destroy;
  PROCEDURE BalayageBoutonParam(HCtrl:HWnd);
  PROCEDURE WMTimer (VAR Msg: TMessage);
```

```
    VIRTUAL wm_First + wm_Timer;
  PROCEDURE WMLButtonDown (VAR Msg: TMessage);
    VIRTUAL wm_First + wm_LButtonDown;
  PROCEDURE WMLButtonUp(VAR Msg: TMessage);
    VIRTUAL wm_First + wm_LButtonUp;
  DESTRUCTOR Done; VIRTUAL;
 END;

{*************************************************************************}
{************************** Implementation ******************************}
{*************************************************************************}

IMPLEMENTATION

 VAR
  PLeOutilParametres: POutilParametres;

{----------------- PROCEDURE AJoutOutilParametres ---------------------}
 PROCEDURE AJoutOutilParametres( AParent: PPaletteOutil);
  BEGIN
    New(PLeOutilParametres,
      init(AParent,id_outil_parametres+id_pal_parametres,pal_Bouton3232,false));
  END;

{=================== PROCEDURES de TOutilParametres ======================}

{----------------- PROCEDURE TOutilParametres.AfficherExplication ------}
 PROCEDURE TOutilParametres.AfficherExplication;
  VAR
    Commentaire : String;
  BEGIN
    Commentaire := 'Active le balayage des paramètres';
    CommtDC := GetDC(ptFenCommentaire^.HWindow);
    SetTextColor(CommtDC,RGB(225,0,0));
    TextOut(CommtDC,2,2,@Commentaire[1],Length(Commentaire));
    ReleaseDC(ptFenCommentaire^.HWindow,CommtDC);
  END;

{----------------- PROCEDURE TOutilParametres.WMLButtonDown ------------}
 PROCEDURE TOutilParametres.WMLButtonDown (VAR Msg:TMessage);
  BEGIN
    Select;
    AfficherExplication;
    Application^.MakeWindow(New(PDiaSelectionBalParam,
      Init(@Self, PChar(id_dial_select_bal_param))));
  END;

{----------------- PROCEDURE TOutilParametres.WMLButtonUp --------------}
 PROCEDURE TOutilParametres.WMLButtonUp ( VAR Msg: TMessage);
  BEGIN
    InvalidateRect(ptFenCommentaire^.HWindow,NIL,true);
    UpdateWindow(ptFenCommentaire^.HWindow);
  END;

{=================== PROCEDURES de TDiaSelectionBalParam ==================}

{----------------- CONSTRUCTEUR TDiaSelectionBalParam.Init -------------}
 CONSTRUCTOR TDiaSelectionBalParam.Init(AParent:PWindowsObject;ATitle:PChar);
  BEGIN
    INHERITED Init (AParent,ATitle);
```

```
    New(BoutonAPositif,InitResource(PDiaSelectionBalParam(self.HWindow),bouton_a_positif));
    New(BoutonANegatif,InitResource(PDiaSelectionBalParam(self.HWindow),bouton_a_negatif));
    New(BoutonBPositif,InitResource(PDiaSelectionBalParam(self.HWindow),bouton_b_positif));
    New(BoutonBNegatif,InitResource(PDiaSelectionBalParam(self.HWindow),bouton_b_negatif));
    New(BoutonCPositif,InitResource(PDiaSelectionBalParam(self.HWindow),bouton_c_positif));
    New(BoutonCNegatif,InitResource(PDiaSelectionBalParam(self.HWindow),bouton_c_negatif));
    LButtonDown := False;
   END;

{------------------ PROCEDURE TDiaSelectionBalParam.SetUpWindow ---------}
 PROCEDURE TDiaSelectionBalParam.SetUpWindow;
   BEGIN
    TDialog.SetUpWindow;
    SetTimer(HWindow,TimerParam_ID,Timer_Intervalle,NIL);
   END;

{------------------ PROCEDURE TDiaSelectionBalParam.WMDestroy -----------}
 PROCEDURE TDiaSelectionBalParam.WMDestroy (VAR Msg: TMessage);
   BEGIN
    KillTimer(HWindow,TimerParam_ID);
    TDialog.WMDestroy(Msg);
   END;

{------------------ PROCEDURE TDiaSelectionBalParam.BalayageBoutonParam -}
 PROCEDURE TDiaSelectionBalParam.BalayageBoutonParam(HCtrl:HWnd);
   VAR
    i : integer;
   BEGIN
    i := 1;
    WHILE i <= Timer_Intervalle1 DO
     BEGIN
      SendDlgItemMessage(HWindow,HCtrl,bm_SetState,1,0);
      i := i + 1;
     END;
    SendDlgItemMessage(HWindow,HCtrl,bm_SetState,0,0);
   END;

{------------------ PROCEDURE TDiaSelectionBalParam.WMTimer -------------}
 PROCEDURE TDiaSelectionBalParam.WMTimer (VAR Msg: TMessage);
   BEGIN
    {SetTimer(HWindow,TimerParam_ID,Timer_Intervalle,NIL);}
    BalayageBoutonParam(Bouton_a_positif);
    {KillTimer(HWindow,TimerParam_ID);}
    IF ActiverParamAPos = true
     THEN
      BEGIN
       { KillTimer(HWindow,TimerParam_ID);}
       BAPos(Msg);
      END
     ELSE
      BEGIN
       ActiverParamAPos := False;
       SetTimer(HWindow,TimerParam_ID,Timer_Intervalle,NIL);
       BalayageBoutonParam(Bouton_a_negatif);
       {KillTimer(HWindow,TimerParam_ID);}
       IF ActiverParamANeg = true
        THEN
         BEGIN
          { KillTimer(HWindow,TimerParam_ID);}
          BANeg(Msg);
```

```
    END
   ELSE
    BEGIN
     ActiverParamAneg := False;
     SetTimer(HWindow,TimerParam_ID,Timer_Intervalle,NIL);
     BalayageBoutonParam(bouton_b_positif);
     {KillTimer(HWindow,TimerParam_ID);}
     IF ActiverParamBPos = true
      THEN
       BEGIN
       { KillTimer(HWindow,TimerParam_ID);}
        BBPos(Msg);
       END
      ELSE
       BEGIN
        ActiverParamBPos := False;
        SetTimer(HWindow,TimerParam_ID,Timer_Intervalle,NIL);
        BalayageBoutonParam(bouton_b_negatif);
        {KillTimer(HWindow,TimerParam_ID);}
        IF ActiverParamBNeg = true
         THEN
          BEGIN
          { KillTimer(HWindow,TimerParam_ID);}
           BBNEg(Msg);
          END
         ELSE
          BEGIN
           ActiverParamBNeg := False;
           SetTimer(HWindow,TimerParam_ID,Timer_Intervalle,NIL);
           BalayageBoutonParam(bouton_c_positif);
           {KillTimer(HWindow,TimerParam_ID);}
           IF ActiverParamCPos = true
            THEN
             BEGIN
             { KillTimer(HWindow,TimerParam_ID);}
              BCPos(Msg);
             END
            ELSE
             BEGIN
              ActiverParamCPos := False;
              SetTimer(HWindow,TimerParam_ID,Timer_Intervalle,NIL);
              BalayageBoutonParam(bouton_c_negatif);
              {KillTimer(HWindow,TimerParam_ID);}
              IF ActiverParamCNeg = true
               THEN
                BEGIN
                 {KillTimer(HWindow,TimerParam_ID);}
                  BCNeg(Msg);
                END
               ELSE
                BEGIN
                 ActiverParamCNeg := False;
                 { SetTimer(HWindow,TimerParam_ID,Timer_Intervalle,NIL);}
                END;
             END;
          END;
       END;
    END;
  ActiverParamAPos := False;
END;
```

```
  END;

{------------------ PROCEDURE TDiaSelecitonBalParam.BAPos ---------------}
 PROCEDURE TDiaSelectionBalParam.BAPos (VAR Msg: TMessage);
  BEGIN
   KillTimer(HWindow,TimerParam_ID);
   IF ActiverParamAPos = False
    THEN
     BEGIN
      ActiverParamAPos := true;
     END;
  END;

{------------------ PROCEDURE TDiaSelectionBalParam.BANeg ---------------}
 PROCEDURE TDiaSelectionBalParam.BANeg (VAR Msg: TMessage);
  BEGIN
   KillTimer(HWindow,TimerParam_ID);
   IF ActiverParamANeg = False
    THEN
     BEGIN
      ActiverParamANeg := True;
     END;
  END;

{------------------ PROCEDURE TDiaSelecitonBalParam.BBPos ---------------}
 PROCEDURE TDiaSelectionBalParam.BBPos (VAR Msg: TMessage);
  BEGIN
   KillTimer(HWindow,TimerParam_ID);
   IF ActiverParamBPos = False
    THEN
     BEGIN
      ActiverParamBPos := true;
     END;
  END;

{------------------ PROCEDURE TDiaSelectionBalParam.BBNeg ---------------}
 PROCEDURE TDiaSelectionBalParam.BBNeg (VAR Msg: TMessage);
  BEGIN
   KillTimer(HWindow,TimerParam_ID);
   IF ActiverParamBNeg = False
    THEN
     BEGIN
      ActiverParamBNeg := True;
     END;
  END;

{------------------ PROCEDURE TDiaSelecitonBalParam.BCPos ---------------}
 PROCEDURE TDiaSelectionBalParam.BCPos (VAR Msg: TMessage);
  BEGIN
   KillTimer(HWindow,TimerParam_ID);
   IF ActiverParamCPos = False
    THEN
     BEGIN
      ActiverParamCPos := true;
     END;
  END;

{------------------ PROCEDURE TDiaSelectionBalParam.BCNeg ---------------}
 PROCEDURE TDiaSelectionBalParam.BCNeg (VAR Msg: TMessage);
  BEGIN
```

```
    KillTimer(HWindow,TimerParam_ID);
    IF ActiverParamCNeg = False
     THEN
      BEGIN
       ActiverParamCNeg := True;
      END;
  END;

{------------------ PROCEDURE TDiaSelectionBalParam.WMLButtonDown -------}
 PROCEDURE TDiaSelectionBalParam.WMLButtonDown (VAR Msg: TMessage);
  BEGIN
   IF NOT LButtonDown THEN WITH Msg DO
    BEGIN
     LButtonDown := True;
     SetTimer(HWindow,TimerParam_ID,Timer_Intervalle,NIL);
    END;
  END;

{------------------ PROCEDURE TDiaSelectionBalParam.WMLbuttonUp ---------}
 PROCEDURE TDiaSelectionBalParam.WMLButtonUp(VAR Msg: TMessage);
  BEGIN
   IF LButtonDown THEN WITH Msg DO
    BEGIN
     LButtonDown := False;
    END;
  END;

{------------------ DESTRUCTEUR TDiaSelectionBalParam.Done --------------}
 DESTRUCTOR TDiaSelectionBalParam.Done;
  BEGIN
   TDialog.Done;
  END;

{*************************************************************************}
{***************************** Initialisation ****************************}
{*************************************************************************}

BEGIN
 ElemPaletteOutil.insert(New(PInitOutilPalette,init(AJoutOutilParametres)));
END.
```

```
{#########################################################################}
{#                    Unité Graphe                                       #}
{#########################################################################}

UNIT Graphe;

{$R graphe.res}

{*************************************************************************}
{************************** Interface ************************************}
{*************************************************************************}

INTERFACE

USES WINTYPES, WINPROCS, OBJECTS, OWINDOWS,
   PALOUTIL, FPALOUT, VARPOINT, VARGLOB;

{$I graphe.inc}

TYPE
 POutilGraphe = ^TOutilGraphe;
 TOutilGraphe = OBJECT (TFenOutilPalette)
  PROCEDURE AfficherExplication;
  PROCEDURE WMLButtonDown ( VAR Msg: TMessage);
   VIRTUAL wm_First + wm_LButtonDown;
  PROCEDURE WMLButtonUp ( VAR Msg: TMessage);
   VIRTUAL wm_First + wm_LButtonUp;
 END;

{*************************************************************************}
{************************** Implementation *******************************}
{*************************************************************************}

IMPLEMENTATION

 VAR
  PLeOutilGraphe : POutilGraphe;

{----------------- PROCEDURE AJoutOutilGraphe -------------------------}
 PROCEDURE AJoutOutilGraphe (AParent: PPaletteOutil);
  BEGIN
   New(PLeOutilGraphe,
    init(AParent,id_outil_graphe+id_pal_graphe,pal_Bouton3232,false));
  END;

{==================== PROCEDURES de TOutilGraphe ========================}

{----------------- PROCEDURE TOutilGraphe.AfficherExplication ----------}
 PROCEDURE TOutilGraphe.AfficherExplication;
  VAR
   Commentaire : String;
  BEGIN
   Commentaire := 'Active le balayage des objets de dessins sur l''écran';
   CommtDC := GetDC(ptFenCommentaire^.HWindow);
   SetTextColor(CommtDC,RGB(225,0,0));
   TextOut(CommtDC,2,2,@Commentaire[1],Length(Commentaire));
   ReleaseDC(ptFenCommentaire^.HWindow,CommtDC);
  END;

{----------------- PROCEDURE TOutilGraphe.WMLButtonDown ----------------}
```

```
 PROCEDURE TOutilGraphe.WMLButtonDown ( VAR Msg: TMessage);
  BEGIN
    Select;
    AfficherExplication;
    MessageBox(HWindow,
           'Cette fonctionnalité n''a pas encore été implémentée',
           'Message d''information',mb_ok or mb_IconExclamation);
  END;

{------------------ PROCEDURE TOutilGraphe.WMLButtonUp ------------------}
 PROCEDURE TOutilGraphe.WMLButtonUp ( VAR Msg: TMessage);
  BEGIN
    InvalidateRect(ptFenCommentaire^.HWindow,NIL,true);
    UpdateWindow(ptFenCommentaire^.HWindow);
  END;

{*************************************************************************}
{***************************** Initialisation ****************************}
{*************************************************************************}

BEGIN
 ElemPaletteOutil.insert(New(PInitOutilPalette,init(AJoutOutilGraphe)));
END.
```

```
{#########################################################################}
{#                    unité zoomplus                                     #}
{#########################################################################}

UNIT ZoomPlus;

{$R zoomplus.res}

{*************************************************************************}
{***************************** Interface *********************************}
{*************************************************************************}

INTERFACE

USES WINTYPES, WINPROCS, OWINDOWS, OBJECTS,
   PALOUTIL, FPALOUT, VARPOINT, VARGLOB;

{$I Zoomplus.inc}

ConstanteZoomPlus = 3;

TYPE
 POutilZoomPlus = ^TOutilZoomPlus;
 TOutilzoomPlus = OBJECT (TFenOutilPalette)
  PROCEDURE AfficherExplication;
  PROCEDURE WMLButtonDown ( VAR Msg: TMessage);
   VIRTUAL wm_First + wm_LButtonDown;
  PROCEDURE WMLButtonUp ( VAR Msg: TMessage);
   VIRTUAL wm_First + wm_LButtonUp;
 END;

{*************************************************************************}
{***************************** Implementation ****************************}
{*************************************************************************}

IMPLEMENTATION

 VAR
  PLeOutilZoomPlus: POutilZoomPlus;

{------------------ PROCEDURE AjoutOutilZoomPlus ------------------------}
 PROCEDURE AJoutOutilZoomPlus(AParent: PPaletteOutil);
  BEGIN
   New(PLeOutilZoomPlus,
      init(aparent,id_pal_zoom_plus+id_outil_zoom_plus,pal_Bouton3232,false));
  END;

{=================== PROCEDURES de TOutilZoomPlus ========================}

{------------------ PROCEDURE TOutilZoomPlus.AfficherExplication --------}
 PROCEDURE TOutilZoomPlus.AfficherExplication;
  VAR
   Commentaire : String;
  BEGIN
   Commentaire := 'Permet une visualisation plus grande du graphe';
   CommtDC := GetDC(ptFenCommentaire^.HWindow);
   SetTextColor(CommtDC,RGB(225,0,0));
   TextOut(CommtDC,2,2,@Commentaire[1],Length(Commentaire));
   ReleaseDC(ptFenCommentaire^.HWindow,CommtDC);
  END;
```

```
{---------- PROCEDURE TOutilZoomPlus.WMLButtonDown ----------}
 PROCEDURE TOutilZoomPlus.WMLButtonDown (VAR Msg:TMessage);
  VAR
   PaintDC : HDC;
   PaintInfo : TPaintStruct;
  BEGIN
   select;
   AfficherExplication;
   compteur1 := compteur1 + ConstanteZoomPlus;
   compteur2 := compteur2 - ConstanteZoomPlus;
   compteur3 := compteur3 + ConstanteZoomPlus;
   compteur4 := compteur4 - ConstanteZoomPlus;
   InvalidateRect(ptFenGraphe^.HWindow,NIL,true);
   ptFenGraphe^.Paint(PaintDC,PaintInfo);
  END;

{---------- PROCEDURE TOutilZoomPlus.WMLButtonUp ----------}
 PROCEDURE TOutilZoomPlus.WMLButtonUp ( VAR Msg: TMessage);
  BEGIN
   InvalidateRect(ptFenCommentaire^.HWindow,NIL,true);
   UpdateWindow(ptFenCommentaire^.HWindow);
  END;

{*************************************************************************}
{***************************** Initialisation ****************************}
{*************************************************************************}

BEGIN
 ElemPaletteOutil.insert(New(PInitOutilPalette,init(AJoutOutilZoomPlus)));
END.
```

```
{#########################################################################}
{#                    Unité ZomMoins                                     #}
{#########################################################################}

UNIT ZomMoins;

{$R zommoins.res}

{*************************************************************************}
{***************************** Interface *********************************}
{*************************************************************************}

INTERFACE

USES WINTYPES, WINPROCS, OWINDOWS, OBJECTS,
   PALOUTIL, FPALOUT, VARPOINT, VARGLOB;

{$I Zommoins.inc}

CONST
 ConstanteZoomMoins = 3;

TYPE
  POutilZoomMoins = ^TOutilZoomMoins;
  TOutilzoomMoins = OBJECT (TFenOutilPalette)
    PROCEDURE AfficherExplication;
    PROCEDURE WMLButtonDown (VAR Msg: TMessage);
      virtual wm_First + wm_LButtonDown;
    PROCEDURE WMLButtonUp ( VAR Msg: TMessage);
      VIRTUAL wm_First + wm_LButtonUp;
 END;

{*************************************************************************}
{***************************** Implementation ****************************}
{*************************************************************************}

IMPLEMENTATION

 VAR
  PLeOutilZoomMoins: POutilZoomMoins;

{----------------- PROCEDURE AJoutOutilZoomMoins ----------------------}
 PROCEDURE AJoutOutilZoomMoins(AParent: PPaletteOutil);
  BEGIN
    New(PLeOutilZoomMoins,
       init(aparent,id_pal_zoom_moins+id_outil_zoom_moins,
          pal_Bouton3232,false));
  END;

{=================== PROCEDURES de TOutilZoomMoins ======================}

{----------------- PROCEDURE TOutilZoomMoins.AfficherExplication -------}
 PROCEDURE TOutilZoomMoins.AfficherExplication;
  VAR
    Commentaire : String;
  BEGIN
    Commentaire := 'Permet une visualisation plus petite du graphe';
    CommtDC := GetDC(ptFenCommentaire^.HWindow);
    SetTextColor(CommtDC,RGB(225,0,0));
    TextOut(CommtDC,2,2,@Commentaire[1],Length(Commentaire));
```

```
    ReleaseDC(ptFenCommentaire^.HWindow,CommtDC);
  END;

{------------------ PROCEDURE TOutilZoomMoins.WMLButtonDown -------------}
 PROCEDURE TOutilZoomMoins.WMLButtonDown (var Msg:TMessage);
  VAR
    PaintDC : HDC;
    PaintInfo : TPaintStruct;
  BEGIN
    select;
    AfficherExplication;
    compteur1 := compteur1 - ConstanteZoomMoins;
    compteur2 := compteur2 + ConstanteZoomMoins;
    compteur3 := compteur3 - ConstanteZoomMoins;
    compteur4 := compteur4 + ConstanteZoomMoins;
    InvalidateRect(ptFenGraphe^.HWindow,NIL,true);
    ptFenGraphe^.Paint(PaintDC,PaintInfo);
  END;

{------------------ PROCEDURE TOutilZoomMoins.WMLButtonUp ---------------}
 PROCEDURE TOutilZoomMoins.WMLButtonUp ( VAR Msg: TMessage);
  BEGIN
    InvalidateRect(ptFenCommentaire^.HWindow,Nil,true);
    UpdateWindow(ptFenCommentaire^.HWindow);
  END;

{*************************************************************************}
{***************************** Initialisation ****************************}
{*************************************************************************}

BEGIN
 ElemPaletteOutil.insert(New(PInitOutilPalette,init(AJoutOutilZoomMoins)));
END.
```

```
{#########################################################################}
{#                    Unité Aide                                         #}
{#########################################################################}

UNIT Aide;

{$R aide.res}

{*************************************************************************}
{***************************** Interface *********************************}
{*************************************************************************}

INTERFACE

USES WINTYPES, WINPROCS, OBJECTS, OWINDOWS,
   PALOUTIL, FPALOUT, VARPOINT, VARGLOB;

{$I aide.inc}

TYPE
 POutilAide = ^TOutilAide;
 TOutilAide = OBJECT (TFenOutilPalette)
  PROCEDURE AfficherExplication;
  PROCEDURE WMLButtonDown ( VAR Msg: TMessage);
    VIRTUAL wm_First + wm_LButtonDown;
  PROCEDURE WMLButtonUp ( VAR Msg: TMessage);
    VIRTUAL wm_First + wm_LButtonUp;
 END;

{*************************************************************************}
{***************************** Implementation ****************************}
{*************************************************************************}

IMPLEMENTATION

 VAR
  PLeOutilAide : POutilAide;

{------------------ PROCEDURE AJoutOutilAide ---------------------------}
 PROCEDURE AJoutOutilAide( AParent: PPaletteOutil);
  BEGIN
    New(PLeOutilAide,
      init(AParent,id_outil_aide+id_pal_aide,pal_Bouton3232,false));
  END;

{==================== PROCEDURES de TOutilAide ============================}

{------------------ PROCEDURE TOutilAide.AfficherExplication ------------}
 PROCEDURE TOutilAide.AfficherExplication;
  VAR
    Commentaire : String;
  BEGIN
    Commentaire := 'Affiche les aides du programme';
    CommtDC := GetDC(ptFenCommentaire^.HWindow);
    SetTextColor(CommtDC,RGB(225,0,0));
    TextOut(CommtDC,2,2,@Commentaire[1],Length(Commentaire));
    ReleaseDC(ptFenCommentaire^.HWindow,CommtDC);
  END;

{------------------ PROCEDURE TOutilAide.WMLButtonDown ------------------}
```

```
 PROCEDURE TOutilAide.WMLButtonDown ( VAR Msg:TMessage);
   BEGIN
     Select;
     AfficherExplication;
     WinHelp(HWindow, 'WINHELP.HLP', Help_Index, 0);
   END;

{----------------- PROCEDURE TOutilAide.WMLButtonUp ----------------}
 PROCEDURE TOutilAide.WMLButtonUp ( VAR Msg: TMessage);
   BEGIN
     InvalidateRect(ptFenCommentaire^.HWindow,NIL,true);
     UpdateWindow(ptFenCommentaire^.HWindow);
   END;

{*************************************************************************}
{************************** Initialisation *******************************}
{*************************************************************************}

BEGIN
 ElemPaletteOutil.insert(New(PInitOutilPalette,init(AJoutOutilaide)));
END.
```

```
{##############################################################################}
{#                        Unité VarGlob                                       #}
{##############################################################################}

UNIT VarGlob;   { Unité des variables globales utilisées }
          { par les différentes unités du programme }

{$R mesquit.res}
{$R sixmenu.res}

{******************************************************************************}
{***************************** Interface **************************************}
{******************************************************************************}

INTERFACE

USES
 WINTYPES, WINPROCS, OWINDOWS, WINDOS, OBJECTS,
 STRINGS, OSTDDLGS, ODIALOGS;

{$I mesquit.inc}
{$I sixmenu.inc}

CONST

 LargEcranParam800 = 328;
 LargEcranParam640 = 266;

 LargEcranCalcul800 = 308;
 LargEcranCalcul640 = 247;

 LargEcranEnonce800 = 430;
 LargEcranEnonce640 = 368;

 HautEcranParam600  = 115;
 HautEcranParam480  = 115;

 LargEcranGraphe800  = 143;
 LargEcranGraphe640  = 143;

 HautEcranGraphe600  = 350;
 HautEcranGraphe480  = 239;

 CoordXEcranSouris800 = 422;
 CoordXEcranSouris640 = 361;

 CoordXEcranCalcul800 = 320;
 CoordXEcranCalcul640 = 259;

 CoordYEcranEnonce600 = 480;
 CoordYEcranEnonce480 = 362;

 CoordYEcranGraphe600 = 420;
 CoordYEcranGraphe480 = 306;

 LongScrollBar1800     = 175;
 LongScrollBar2800     = 350;
 LongScrollBar3800     = 350;

 DepartScrollBar1800    = 285;
```

```
  DepartScrollBar1640    = 235;

  LongScrollBar1640      = 125;
  LongScrollBar2640      = 250;
  LongScrollBar3640      = 250;

CONST
  Timer_ID           = 1;
  TimerBarDef_ID     = 2;
  TimerParam_ID      = 3;
  Timer_Intervalle   = 1000;
  Timer_Intervalle1  = 20000;
  Timer_Intervalle2  = 10000;

CONST
  BarDefIntervalInf = -10;
  BarDefIntervalSup =  10;

CONST
  NbEqDroite = 6;
  EqDroite : ARRAY[0..NbEqDroite-1] OF PChar = (
    ' y = ax + b ',
    ' y = ax ',
    ' y = tgu x',
    ' y-y0 = a ( x - x0 )',
    ' y-y0 = tgu ( x - x0 )',
    ' y-y0 = (y1-y0/x1-x0)(x-x0)');

  NbEqParabole = 2;
  EqParabole : ARRAY[0..NbEqParabole-1] OF PChar = (
    ' y = ax^2 ',
    ' y = ax^2 + bx + c ');

  NbEqTrigo = 3;
  EqTrigo : ARRAY[0..NbEqTrigo-1] OF PChar = (
    ' y = k * sin(ax+b)',
    ' y = k * cos(ax+b)',
    ' y = k * tg(ax+b)');

VAR
  EstNouvFich : Boolean;
  NomFich : ARRAY [0..fsPathName] OF Char;
  Titre : ARRAY [0..fsPathName + 19] OF Char;

  ElemPaletteOutil : TCollection;

  Scroll_1  : PScrollBar;
  Scroll_1a : PScrollBar;
  Scroll_2  : PScrollBar;
  Scroll_3  : PScrollBar;

  Ed_1 : PEdit;
  Ed_2 : PEdit;
  Ed_3 : PEdit;

  EditBoite : PEdit;
  EcrireTexte : PChar;

  pen1, pen2 : HPen;
  brush1, brush2 : HBrush;
```

```
  SourisDC : HDC;

  CommtDc : HDC;

  reponse : integer;

  compteur1 : integer;
  compteur2 : integer;
  compteur3 : integer;
  compteur4 : integer;

  parametre1 : integer;
  parametre2 : integer;
  parametre3 : integer;

  EstModifie : Boolean;

  DroiteBool : Boolean;
  ParaboleBool : Boolean;
  TrigonoBool : Boolean;

  ActiverParamAPos : Boolean;
  ActiverParamANeg : Boolean;
  ActiverParamBPos : Boolean;
  ActiverParamBNeg : Boolean;
  ActiverParamCPos : Boolean;
  ActiverParamCNeg : Boolean;

  FenAyantCapture : HWnd;

TYPE
  PDiaMessageQuitter = ^TDiaMessageQuitter;
  TDiaMessageQuitter = OBJECT (TDialog)
    BoutonOui : PButton;
    BoutonNon : Pbutton;
    BOuiHandle : HWnd;
    BNonHandle : HWnd;
    CONSTRUCTOR Init(AParent:PWindowsObject;ATitle:PChar);
    PROCEDURE SetupWindow; VIRTUAL;
    PROCEDURE MesBoutonOui (VAR Msg: TMessage);
      VIRTUAL id_First + id_boutonOui;
    PROCEDURE MesBoutonNon (VAR Msg: TMessage);
      VIRTUAL id_First + id_boutonNon;
    PROCEDURE WMDestroy (VAR Msg: TMessage);
      VIRTUAL wm_First + wm_Destroy;
    PROCEDURE BalayageBouton;
    PROCEDURE WMTimer (VAR Msg: TMessage);
      VIRTUAL wm_First + wm_Timer;
  END;

  PDiaSauverFichier = ^TDiaSauverFichier;
  TDiaSauverFichier = OBJECT (TDialog)
    PROCEDURE IDHelp (VAR Msg: TMessage);
      VIRTUAL id_First + id_Help;
  END;

{*************************************************************************}
{************************** Implementation ******************************}
{*************************************************************************}
```

```
IMPLEMENTATION

{---------- PROCEDURE ActiverLeTimer ----------}
 PROCEDURE ActiverLeTimer(FHdle:HWnd);
  BEGIN
   SetTimer(FHdle,TimerBarDef_ID,Timer_Intervalle,NIL);
  END;

{========== PROCEDURES de TDiaMessageQuitter ==========}

{---------- CONSTRUCTEUR TDiaMessageQuitter.Init ----------}
 CONSTRUCTOR TDiaMessageQuitter.Init(AParent:PWindowsObject;ATitle:PChar);
  BEGIN
   INHERITED Init(AParent,ATitle);
   New(BoutonOui,InitResource(PDiaMessageQuitter(@self),id_boutonOui));
   New(BoutonNon,InitResource(PDiaMessageQuitter(@self),id_boutonNon));
  END;

{---------- PROCEDURE TDiaMessageQuitter.SetupWindow ----------}
 PROCEDURE TDiaMessageQuitter.SetupWindow;
  BEGIN
   TDialog.SetupWindow;
   SetTimer(HWindow, Timer_ID, Timer_Intervalle, NIL);
   BOuiHandle := GetDlgItem(self.HWindow,id_BoutonOui);
   BNonHandle := GetDlgItem(self.HWindow,id_BoutonNon);
  END;

{---------- PROCEDURE TDiaMessageQuitter.WMDestroy ----------}
 PROCEDURE TDiaMessageQuitter.WMDestroy (VAR Msg: TMessage);
  BEGIN
   KillTimer(HWindow, Timer_ID);
   TDialog.WMDestroy(Msg);
  END;

{---------- PROCEDURE TDiaMessageQuitter.BalayageBouton ---------}
 PROCEDURE TDiaMessageQuitter.BalayageBouton;
  VAR
   i : integer;
  BEGIN
   i := 1;
   while i <= timer_intervalle1 do
    BEGIN
     SendDlgItemMessage(HWindow,id_boutonOui,bm_Setstate,1,0);
     i := i + 1;
    END;
   SendDlgItemMessage(HWindow,id_boutonOui,bm_Setstate,0,0);
   i := 1;
   while i <= timer_intervalle1 do
    BEGIN
     SendDlgItemMessage(HWindow,id_boutonNon,bm_Setstate,1,0);
     i := i + 1;
    END;
   SendDlgItemMessage(HWindow,id_boutonNon,bm_setState,0,0);
  END;

{---------- PROCEDURE TDiaMessageQuitter.WMTimer ----------}
 PROCEDURE TDiaMessageQuitter.WMTimer (VAR Msg: TMessage);
  BEGIN
   BalayageBouton;
```

```
  END;

{------------------ PROCEDURE TDiaMessageQuitter.Ok --------------------}
 PROCEDURE TDiaMessageQuitter.MesBoutonOui (VAR Msg: TMessage);
  VAR
   R1 : HWnd;
  BEGIN
   KillTimer(self.HWindow,Timer_ID);
   Application^.ExecDialog(New(PDiaSauverFichier,
    Init(@Self, PChar(id_sixieme_menu_prog))));
   {MessageBox(HWindow,'balayage stopé','boîte de message',mb_ok);}
   TDialog.Ok(Msg);
  END;

{------------------ PROCEDURE TDiaMessageQuitter.Cancel ----------------}
 PROCEDURE TDiaMessageQuitter.MesBoutonNon (VAR Msg: TMessage);
  BEGIN
   TDialog.Cancel(Msg);
  END;

{=================== PROCEDURES de TDiaSauverFichier ======================}

{------------------ PROCEDURE TDiaSauverFichier.IDHelp ----------------}
 PROCEDURE TDiaSauverFichier.IDHelp (VAR Msg: TMessage);
  BEGIN
   MessageBox(HWindow,
         'Cette fonctionnalité n''a pas encore été implémentéé',
         'Message d''information',mb_Ok OR mb_IconExclamation);
  END;

{*************************************************************************}
{*************************** Initialisation ******************************}
{*************************************************************************}

BEGIN

 EstNouvFich := true;

 compteur1 := 0;
 compteur2 := 0;
 compteur3 := 0;
 compteur4 := 0;

 parametre1 := 1;
 parametre2 := 0;
 parametre3 := 0;

 EstModifie := False;

 DroiteBool := False;
 ParaboleBool := False;
 TrigonoBool := False;

 pen1 := 0;
 pen2 := 0;
 brush1 := 0;
 brush2 := 0;

 SourisDC := 0;
 CommtDC := 0;
```

```
  ActiverParamAPos := False;
  ActiverParamANeg := False;
  ActiverParamBPos := False;
  ActiverParamBNeg := False;
  ActiverParamCPos := False;
  ActiverParamCNeg := False;

  FenAyantCapture := 0;

END.
```

```
{##############################################################################}
{#                    Unité VarPoint                                          #}
{##############################################################################}

UNIT VarPoint; { Unité des variables globales pointeurs utilisées }
          { par les différents unités du programme }

{******************************************************************************}
{***************************** Interface **************************************}
{******************************************************************************}

INTERFACE

USES
 WINTYPES, WINPROCS, OWINDOWS, STRINGS, WINDOS,
 OBJECTS, OSTDDLGS, ODIALOGS,
 GESTFICH, FEXPLIC, CALPROP, FGRAPHE1;

TYPE
 PVariablesPointeurs = ^TVariablesPointeurs;
 TVariablesPointeurs = OBJECT (TWindow)
  CONSTRUCTOR Init(AParent:PWindowsObject; ATitle: PChar;
   pgWindow : PWindow;
   ptgGestionFichier: PGestionFichier;
   ptgFenCommentaire: PFenCommentaire;
   ptgFenGraphe: PFenGraphe;
   ptgFenCalculsProp: PFenCalculsProp);
  DESTRUCTOR Done; VIRTUAL;
 END;

VAR
 paWindow : PWindow;
 ptGestionFichier : PGestionFichier;
 ptFenCommentaire : PFenCommentaire;
 ptFenGraphe : PFenGraphe;
 ptFenCalculsProp : PFenCalculsProp;

{******************************************************************************}
{************************** Implementation ************************************}
{******************************************************************************}

IMPLEMENTATION

{------------------ CONSTRUCTEUR TVariablesPointeurs.Init ---------------}
 CONSTRUCTOR TVariablesPointeurs.Init
   (AParent:PWindowsObject; ATitle: PChar;
    pgWindow : PWindow;
    ptgGestionFichier: PGestionFichier;
    ptgFenCommentaire: PFenCommentaire;
    ptgFenGraphe: PFenGraphe;
    ptgFenCalculsProp: PFenCalculsProp);

  BEGIN
   paWindow := pgWindow;
   ptGestionFichier := ptgGestionFichier;
   ptFenCommentaire := ptgFenCommentaire;
   ptFenGraphe := ptgFenGraphe;
   ptFenCalculsProp := ptgFenCalculsProp;
  END;
```

```
{------------------ DESTRUCTEUR TVariablesPointeurs.Done ----------------}
 DESTRUCTOR TVariablesPointeurs.Done;
   BEGIN
    INHERITED Done;
   END;

{*************************************************************************}
{***************************** Initialisation ****************************}
{*************************************************************************}

BEGIN
END.
```

```
{#########################################################################}
{#                    Unité GestFich                                     #}
{#########################################################################}

UNIT GestFich; {Unité de la gestion des fichiers}

{*************************************************************************}
{************************** Interface ************************************}
{*************************************************************************}

INTERFACE

USES
  WINTYPES, WINPROCS, OWINDOWS, OBJECTS, WINDOS,
  STRINGS, OSTDDLGS, ODIALOGS, WINCRT,
  VARGLOB, DFAMFONC;

TYPE
  PGestionFichier = ^TGestionFichier;
  TGestionFichier = OBJECT(TWindow)
    CONSTRUCTOR Init(AParent:PWindowsObject;ATitle:PChar);
    DESTRUCTOR Done ; VIRTUAL;
    PROCEDURE EntrerNomFich(NouvNomFich:PChar);
    PROCEDURE ChargerFich (NouvNomFich:PChar);
    PROCEDURE SauverFich(gNomFich:PChar);
  END;

{*************************************************************************}
{************************** Implementation *******************************}
{*************************************************************************}

IMPLEMENTATION

{------------------ CONSTRUCTEUR TGestionFichier.Init -------------------}
  CONSTRUCTOR TGestionFichier.Init(AParent:PWindowsObject;ATitle:PChar);
    BEGIN
      INHERITED Init(AParent,ATitle);
      EstNouvFich := true;
      EntrerNomFich('sans nom');
    END;

{------------------ DESTRUCTEUR TGestionFichier.Done --------------------}
  DESTRUCTOR TGestionFichier.Done;
    BEGIN
      INHERITED Done;
    END;

{------------------ PROCEDURE EntrerNomFich ----------------------------}
  PROCEDURE TGestionFichier.EntrerNomFich(NouvNomFich:PChar);
    BEGIN
      StrCopy(NomFich,NouvNomFich);
      StrCopy(StrECopy(StrECopy(Titre,'Etude de fonctions: ['),
            NomFich), ']');
      SetWindowText(HWindow,Titre);
    END;

{------------------ PROCEDURE ChargerFich ------------------------------}
  PROCEDURE TGestionFichier.ChargerFich (NouvNomFich:PChar);
    VAR
      aFichier:TDosStream;
```

```
  BEGIN
    aFichier.Init(NouvNomFich,stOpen);
    CASE aFichier.status OF
      stOK:
        BEGIN
          EstNouvFich:=false;
          InvalidateRect(HWindow,NIL,true);
          EntrerNomFich(NouvNomFich);
        END;
      stInitError:
        MessageBox(HWindow,'Nom de fichier invalide',
                   'Stream error',mb_IconExclamation OR mb_OK);
      ELSE
        {DisplayStreamError(aFichier);}
    END;
    aFichier.done;
  END;

{------------------ PROCEDURE SauverFich -------------------------------}
 PROCEDURE TGestionFichier.SauverFich(gNomFich:PChar);
  VAR
    aFichier:TDosStream;
  BEGIN
    aFichier.Init(gNomFich,stCreate);
    aFichier.Done;
    EstNouvFich:=false;
    EntrerNomFich(gNomFich);
  END;

{*************************************************************************}
{*************************** Initialisation ******************************}
{*************************************************************************}

BEGIN
END.
```

ULTES UNIVERSITAIRES NOTRE-DAME DE LA PAIX - NAMUR

FUNDP

INSTITUT D'INFORMATIQUE
Rue Grandgagnage 21 - 5000 Namur
Tél. : 081 / 72 49 83

Didacticiel de fonctions mathématiques simples et leurs représentations graphiques

Mémoire présenté en vue de l'obtention
du titre de Licenciée et Maître en Informatique

Promoteur
M. Claude CHERTON

Septembre 1996

Sivilai NGUYEN

ADDENDA

ADDENDA

Vous trouverez dans ce qui suit :

- La correction des « Remerciements » : 2°§, 1° ligne : enlever « mon promoteur de mémoire »

- La correction du « Plan » : au point 11, avant-dernier mot : remplacer « d'une » par « d'autre »

- La correction des pages suivantes :

Page 9 : modifier la mise en page de l'avant-dernier paragraphe

Page 14 : dans Figure 2.3 : ajouter un « s » à « Afficher les propriétés mathématique »

Page 17 : au point 3.2.1.1 : remplacer « Quant à » par « Pour » et ajouter un « e » à « défini »

Page 20 : au point 3.2.3.2 : au dernier paragraphe, à la deuxième phrase : déplacer « En effet, (...). (...) la compréhension de l'élève. » et mettre à la fin du point 3.2.3.1

Page 21 : 1°§, dernière ligne : remplacer « trois » par « deux »

Page 22 : 1°§, 2° ligne : remplacer « les » par « le »

Page 24 : 4°§, 1° ligne : ajouter « envisager » après « ...nous pourrions... »
4° ligne : remplacer « ...a besoin...des graphes » par « ait besoin d'un balayage plus lent pour sélectionner son énoncé d'exercice et un balayage plus rapide pour sélectionner un point ou une courbe dans la fenêtre des graphes »

Page 28 : 1°§, 1° ligne : remplacer « sauvetage » par « sauvegarde »
3°§, 4° ligne : remplacer « ces » par « ses »
4°§, 2° phrase : remplacer « Lorsque ... modifications » par « Lorsque, pour un exercice donné, l'élève veut apporter des modifications (par exemple, en changeant les valeurs des paramètres), nous devrons aussi lui proposer de sauvegarder ses modifications »

Page 35 : au point 4.3.1.3 : ajouter « devra » devant « permettre... »
au point 4.3.1.6, 3° ligne : remplacer « Elle » par « Il »

Page 38 : au point 4.3.2.2.1 : remplacer « ...affichés... » par « ...affichées... »

Page 39 : au point 4.3.2.3.3, 2°§ : enlever le « r » du mot « pourvoir »

Page 41 : au point 4.3.2.5.1 : ajouter un « e » à « activés »

Page 46 : ajouter un « s » à « transfert des valeurs des paramètre »

Page 47 : remplacer « logiciel » par « projet »

Page 50 : au point 5.1, 2°§, 3° ligne : remplacer « ses » par « leurs »
au point 5.1.1 : remplacer « la fenêtre de procédure » par « la procédure de fenêtre »

Page 77 : dans Figure 6.7 : remplacer « septième » par « sixième »

Page 78 : dans Figure 6.8 : remplacer « sixième » par « septième »

Page 80 : au point 6.4.1.3, avant-dernière ligne : remplacer « ...il peut aller...paramètres » par « il peut soit aller affecter des valeurs aux paramètres, soit en modifier les valeurs »

Page 82 : dernière ligne : ajouter un « e » à « grand » et « petit »

Page 86 : au point 6.4.2.2 : ajouter « Le rectangle arrondi signifie qu'une action est prise en charge par la machine »

Page 96 : 1°§, 1° ligne : ajouter « où » après « ...dans la mesure »

Page 97 : au point 7.2, 1°§, 2° ligne : ajouter « puisse » devant « être utilisé... »
au point 7.3.1 : enlever le deuxième «) » de « y = k * sin (ax + b) »

Page 99 : 2°§, 2° ligne : ajouter « assisté » après « ...l'enseignement »
dernière phrase : remplacer « fastidieuse » par « de travail intense »
remplacer « celui » par « celle »

Page 100 : dernier paragraphe, 1° ligne : remplacer « mon choix » par « le mien »
à la dernière ligne : ajouter un « s » à « autonome »

- La correction de la table des matières
- La correction de la table des figures

Remerciements

Tout d'abord, je voudrais remercier le C.E.T.D. (Centre d'Enseignement et de Traitements Différenciés), en particulier Madame Yvette Heyters et Madame Chantal Blomart de m'avoir consacré une partie de leur temps pour discuter de mon sujet de mémoire. Je voudrais également remercier Paul, un élève du centre, de m'avoir fait découvrir son univers de travail.

Ensuite, je tiens à remercier Monsieur Claude Cherton de m'avoir consacré une grande partie de son temps pour la réalisation de ce mémoire. Ses conseils précieux et judicieux, ses remarques pertinentes et ses encouragements m'ont énormément aidée tout au long de mon travail.

Enfin, je voudrais faire un clin d'oeil à ma famille, en particulier à mes parents, ma soeur et mon frère pour leur compréhension, leur soutien constant et leur affection tout au long de mes années d'études.

Plan

1. Introduction

Présentation du sujet du mémoire : poser le problème, le situer dans son contexte et définir des conditions de travail.

2. Le projet

Définition du logiciel : quel est son objectif, quelles sont les lignes directrices qui nous guideront lors de son élaboration et quel type d'activités offrira-t-il à l'utilisateur ?

3. Analyse fonctionnelle

Etude complète des différentes fonctionnalités du logiciel sous différents aspects : l'aspect mathématique, graphique, interactif et handicap d'une part, l'interface et les aides du logiciel d'autre part.

4. Analyse conceptuelle

Proposition d'une architecture logique du projet en se basant sur l'étude des fonctionnalités : définition et description des différentes unités et fonctions du logiciel.

5. Implémentation

Implémentation orientée objet du projet : définition et description des différents objets de notre logiciel. Ces objets seront accompagnés de leurs attributs et de leurs méthodes.

6. Création de l'interface

Dans un premier temps, présentation visuelle et définition des fonctionnalités de quelques fenêtres du logiciel et dans un second temps l'enchaînement de quelques unes d'entre-elles.

7. Conclusion

8. Bibliographie

9. Table des matières

10. Table des figures

11. Annexes

Les annexes concernant l'analyse mathématique et graphique du logiciel d'une part, et les explications du terme « infirmité motrice cérébrale » (I.M.C.) d'autre part.

12. Code source

Listing du code source du programme (dans l'état actuel et sans commentaire).

2.1 L'objectif du logiciel

L'objectif de notre didacticiel est d'aider les élèves à comprendre et à assimiler le plus aisément possible les notions de base relatives à quelques familles de fonctions simples ainsi qu'à leurs représentations graphiques.

Les familles envisagées sont :

- Les fonctions linéaires de la forme $y = ax + b$;
- Les fonctions du second degré de la forme $y = ax^2 + bx + c$;
- Certaines fonctions trigonométriques :

  - $k * \sin(ax+b)$
  - $k * \cos(ax+b)$
  - $k * \tg(ax+b)$

Nous nous sommes restreints à l'étude de ces quelques familles de fonctions pour trois raisons :

- La première concerne directement les élèves du C.E.T.D. En effet, de ce qui est sorti de notre discussion avec les professeurs du centre, c'est ce que les élèves voient principalement dans leur cours de mathématiques.

- La deuxième concerne notre objectif en général. Nous pensons que pour l'atteindre, partir de ces trois types de fonctions peut constituer une bonne base à la compréhension des élèves du secondaire en général, ainsi qu'à celle des élèves handicapés moteur en particulier. Pour aller plus loin, il est vrai qu'il serait intéressant d'envisager l'étude d'autres familles de fonctions telles que les fonctions rationnelles. Mais nous pensons que cela pourrait faire l'objet d'un autre sujet de mémoire vu l'étendue des possibilités envisageables.

- Enfin, la troisième raison concerne le délai de temps qui nous est réservé pour la réalisation de ce mémoire.

2.4.3 Continuer un exercice existant

Action 1 : Aller rechercher le fichier d'exercice

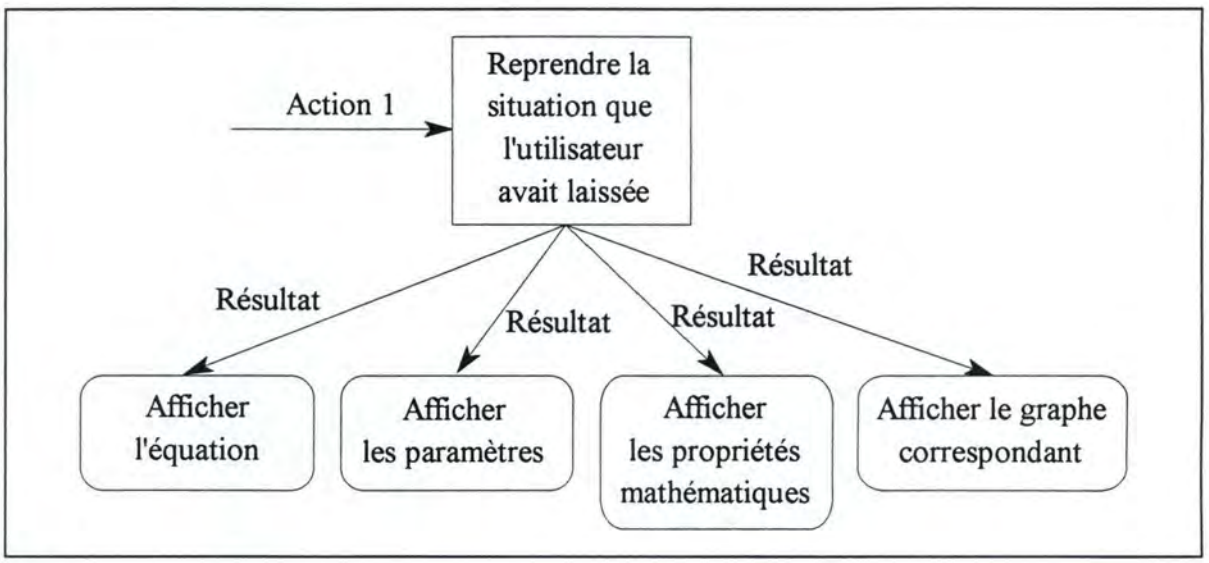


Figure 2.3 : Reprise d'un exercice

Action 2 : Modifier les valeurs des paramètres de l'équation en cours

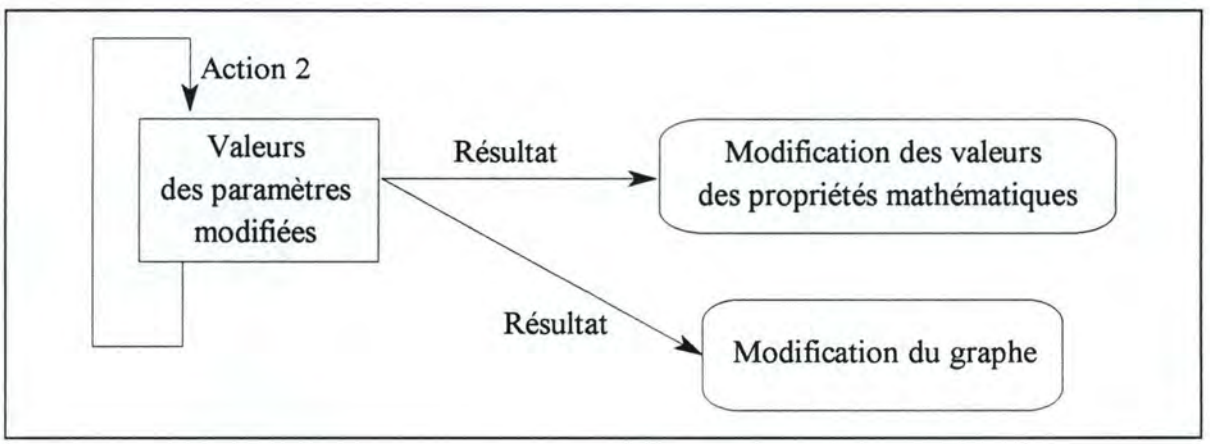


Figure 2.4 : Modification des valeurs des paramètres d'un exercice en cours

3.1 Introduction

L'étude des fonctionnalités du logiciel nous permettra de réfléchir à la manière de proposer une solution à notre problème de départ et de réaliser une architecture logique de cette solution. Pour que cette étude soit complète, nous aborderons le problème sous différents aspects : nous commencerons par la partie mathématique et graphique; ensuite, nous analyserons l'aspect interactif et handicap de l'élève, et nous terminerons par l'interface et les aides de notre logiciel.

3.2 Etude des fonctionnalités

3.2.1 Partie mathématique

Au niveau mathématique, nous avons fait pour chaque famille de fonctions leur étude et ce en tenant compte du contenu du cours de mathématiques des élèves.

3.2.1.1 Equations et propriétés

- La droite

Pour la droite, nous l'avons définie sous sa forme générale :

◊ la droite quelconque ($y = ax + b$)

Les propriétés mathématiques qui lui sont attachées sont :

◊ le coefficient angulaire;
◊ l'angle avec l'axe des X;
◊ l'intersection avec l'axe des X et l'axe des Y.

3.2.3 Partie interaction

Le fait que le logiciel est destiné à des handicapés moteur implique non seulement la nécessité d'une interface adéquate mais influence la conception des fonctionnalités elles-mêmes. En effet, dans le cas d'un utilisateur normal, celui-ci peut effectuer un grand nombre de manipulations simples ou complexes (la frappe d'un texte par exemple) sans que cela lui demande un effort considérable, mais il n'en va plus de même dans notre cas. Nous devrons donc toujours garder cette idée en tête et concevoir le logiciel de manière telle que les manipulations effectuées par l'élève soient minimales sans pour autant diminuer ses possibilités de compréhension.

3.2.3.1 Saisie de l'énoncé

Pour la saisie de l'énoncé d'un exercice, l'élève doit sélectionner une des formes d'équations générales proposées dans la partie mathématique (voir le point 3.2.1).

Le fait que l'on ne permet pas à l'élève d'aller éditer lui-même son énoncé est dû à un de nos objectifs à savoir la minimisation des manipulations de l'élève. En effet, si l'élève doit lui-même éditer son énoncé, nous devrons lui fournir un éditeur de texte spécial. Or, le fait d'aller éditer soi-même son énoncé d'exercice ou le sélectionner dans un menu ne change rien à la compréhension de l'élève.

3.2.3.2 Détermination des valeurs de paramètres

Une fois que de l'énoncé est affiché, il faut encore déterminer les valeurs des paramètres. Deux principes seront ici à la base du choix de la façon de procéder : minimiser le nombre de manipulations et permettre une visualisation dynamique de la relation entre l'expression algébrique et la représentation graphique.

Pour ce faire, nous envisageons la possibilité suivante : permettre à l'élève de sélectionner le paramètre à modifier. Une fois sélectionné, la machine modifiera elle-même les valeurs et l'élève n'aura plus qu'à confirmer la modification.

L'aspect dynamique de toutes ces actions réside dans le fait que pour chaque modification des valeurs des paramètres, l'utilisateur peut l'observer directement sur le graphique.

3.2.3.3 Calcul des propriétés mathématiques

Une étude de fonctions s'accompagne généralement d'une série de calculs de propriétés mathématiques telles que l'intersection avec les axes. Nous avons vu dans la partie mathématique (voir point 3.2.1) que pour une famille de fonctions donnée, nous avons la possibilité de faire plusieurs calculs. En temps normal, l'élève fait ses calculs et trace son dessin graphique en fonction des résultats de ces derniers. Dans le cas qui nous occupe, nous voyons deux façons de procéder :

1. La machine a la charge de tous les calculs;
2. L'élève indique à la machine les différents calculs à effectuer.

La première solution permet de réduire le nombre de démarches à faire par l'élève mais peut réduire également son niveau de compréhension. La deuxième solution semble être la meilleure car l'élève demandera à la machine d'exécuter les calculs qui l'intéressent. De plus, elle permet de réduire les manipulations de l'élève.

Pour montrer le lien dynamique existant entre les valeurs des paramètres d'une équation et ses propriétés mathématiques, nous suggérons que la machine prenne en charge les différents calculs et affiche les résultats à l'écran. A chaque modification de valeurs des paramètres, elle les recalcule et les affiche.

3.2.3.4 Modification du dessin sur l'écran graphique

Nous avons dit qu'une des façons de comprendre le lien existant entre un graphe et sa fonction algébrique est de le manipuler directement. Nous venons de voir la modification de la fonction par ses paramètres (voir point 3.2.3.2). Maintenant, comment envisager les modifications sur graphe ?

Pour une personne handicapée, une des façons de se déplacer sur l'écran est le balayage automatique, c'est-à-dire que tous les éléments d'une fenêtre sont rendus actifs par machine de telle sorte que l'utilisateur n'aura qu'à cliquer sur un bouton pour confirmer sa sélection.

Dans le problème qui nous occupe, nous voyons deux types de modification sur les graphes, une modification ponctuelle (déplacement d'un point de la droite par exemple) et une modification de l'ensemble de la courbe (le déplacement d'une parabole vers le haut par exemple).

Pour la modification ponctuelle et globale, nous voyons deux solutions :

1. Définir une liste d'objets de dessin à sélectionner.
2. Pour un objet de dessin sélectionné, fournir un système de déplacement par directions.

La première solution nous permet d'optimiser le déplacement et la deuxième nous semble être plus naturelle. Evidemment la vitesse de ces balayages sera modifiable en fonction du choix de l'utilisateur lors de la configuration de son interface.

Concrètement, cela devrait se dérouler de la manière suivante : lorsque l'utilisateur veut poser son curseur à un endroit précis, un système de balayage se met en marche et balaye les différents objets de dessin à l'écran. Pour un objet de dessin sélectionné, un système de direction se met en marche permettant ainsi à l'utilisateur de sélectionner le sens de déplacement de son curseur (gauche, droite, bas, haut). Enfin un système de validation permettra de clôturer ou d'annuler la modification en cours.

3.2.4.3 Personnalisation de l'interface

Une de nos lignes directrices est de permettre une configuration de l'interface aussi souple que possible.

Donc, avant de commencer tout exercice, nous envisageons de laisser à l'enseignant la possibilité de configurer l'interface de l'élève. Pour ce faire, une fenêtre présentera les différentes possibilités concernant le balayage des écrans. L'enseignant aura les possibilités suivantes :

1. Choisir la vitesse de balayage;
2. Sauvegarder les options pour chaque élève.

D'une part, nous pourrions envisager que l'enseignant choisisse pour l'élève la même vitesse de balayage pour toutes les fenêtres, d'autre part il serait intéressant que l'utilisateur puisse choisir pour chaque type de fenêtre une vitesse de balayage différente. En d'autres termes, nous pourrions imaginer que l'utilisateur ait besoin d'un balayage plus lent pour sélectionner son énoncé d'exercice et un balayage plus rapide pour sélectionner un point ou une courbe dans la fenêtre des graphes.

Dans ce cas, dans la fenêtre des options de l'interface, nous devrons laisser à l'utilisateur le choix de plusieurs vitesses de balayage pour les différentes fenêtres en question.

Cependant, l'utilisateur peut ne pas vouloir configurer lui-même son interface et demander au système de le faire. Par défaut, la machine affectera une certaine vitesse de balayage pour les différents types de fenêtres.

Les différents types de fenêtres seraient :

- Une fenêtre
- Une boîte de dialogue
- Une boîte de message

Fichier → avec nom	→	sauver sous le même nom
	→	sauver sous un autre nom
→ sans nom	→	sauver sous un nouveau nom

Maintenant que nous avons vu les différentes possibilités de sauvegarde d'un fichier, nous devons encore voir comment le faire concrètement c'est-à-dire comment tout cela va se présenter à l'écran.

Rappelons que dans notre cas, l'enseignant et l'élève devraient pouvoir faire cette opération de sauvegarde. Pour le professeur, aucun problème ne se pose car lorsqu'il veut sauver un fichier, il lui suffit, avec la souris par exemple, d'aller dans la barre de menus et sélectionner la commande « sauver » ou « sauver sous ».

Tout ce mécanisme doit être revu pour l'élève. Rappelons que nous voulons minimiser les manipulations de l'élève, alors une solution à ce problème serait d'envisager des noms de fichiers par défaut, c'est-à-dire que l'élève n'aura pas à éditer le nom de son fichier vu que la machine attribuera de manière automatique un nom à tous ses fichiers. Donc, quelque part, nous devrons gérer la liste des noms de fichiers par défaut. De cette façon, l'élève n'aura qu'à valider son choix. Dans ce cas, nous n'aurons pas à nous poser la question comment gérer la saisie de données et la fin de cette saisie.

Nous proposons la solution suivante : à chaque exercice correspond un fichier (par exemple le premier exercice sera sauvé dans le fichier portant le nom *exercice1*, le deuxième *exercice2*, etc.). Lorsque, pour un exercice donné, l'élève veut apporter des modifications (par exemple, en changeant les valeurs des paramètres), nous devrons aussi lui proposer de sauvegarder ses modifications.

Dans le cas où celui-ci accepte de garder trace de ses modifications, nous devons sauver l'exercice sous un nom par défaut (par exemple, en effectuant son premier exercice, l'élève verra son exercice sauvé sous le nom «*exercice1*» et lorsqu'il sauve ses modifications, celles-ci seront sauvées sous le nom par défaut «*exercice1a*», etc.). Ainsi, la sauvegarde sera prise en charge par la machine et non l'utilisateur (dans le cas de l'élève).
Pour ces différentes opérations, il serait judicieux et intéressant de donner une aide à l'utilisateur.

4.3.1.3 La gestion des paramètres

Cette partie a pour but de gérer l'affectation et la modification des valeurs des paramètres d'une équation algébrique. Par défaut, la machine affectera des valeurs aux paramètres et à l'aide de la souris ou du balayage, devra permettre à l'utilisateur d'apporter des modifications.

4.3.1.4 La gestion des propriétés mathématiques

Dans cette partie, la machine a la charge de tous les calculs. A l'affichage d'un énoncé, elle calculera les valeurs des propriétés mathématiques qui lui sont attachées et refaire les calculs lors de chaque modification des valeurs des paramètres.

4.3.1.5 La gestion des graphes

Cette partie aura la charge de permettre à l'utilisateur d'aller sélectionner un des objets de dessin dans la fenêtre des graphes et d'en modifier la position. Pour garder l'aspect dynamique du logiciel, elle doit aussi communiquer avec le gestionnaire des paramètres et le gestionnaire des énoncés, de même qu'avec le gestionnaire de calculs lors de chaque modification de dessin.

4.3.1.6 La gestion du balayage

Certains handicapés moteur ont du mal à utiliser la souris ou en sont incapables. Ce gestionnaire permet de la remplacer par un affichage dynamique des objets à sélectionner. Il essaie d'être le plus souple possible afin de s'adapter aux besoins de chaque élève. Il s'occupera d'une part de la configuration des vitesses de balayage et d'autre part de sa sauvegarde sur disque.

4.3.1.7 La gestion des données

Cette partie s'occupe de la sauvegarde des paramètres, des dessins et des calculs lors de chaque modification ou d'affectation à la demande de l'utilisateur. Elle s'occupe également de charger ou de sauver un exercice effectué par l'utilisateur. En outre, elle aura la charge des transferts de données entre les différentes unités.

4.3.2.2 L'unité: Gestion de l'énoncé

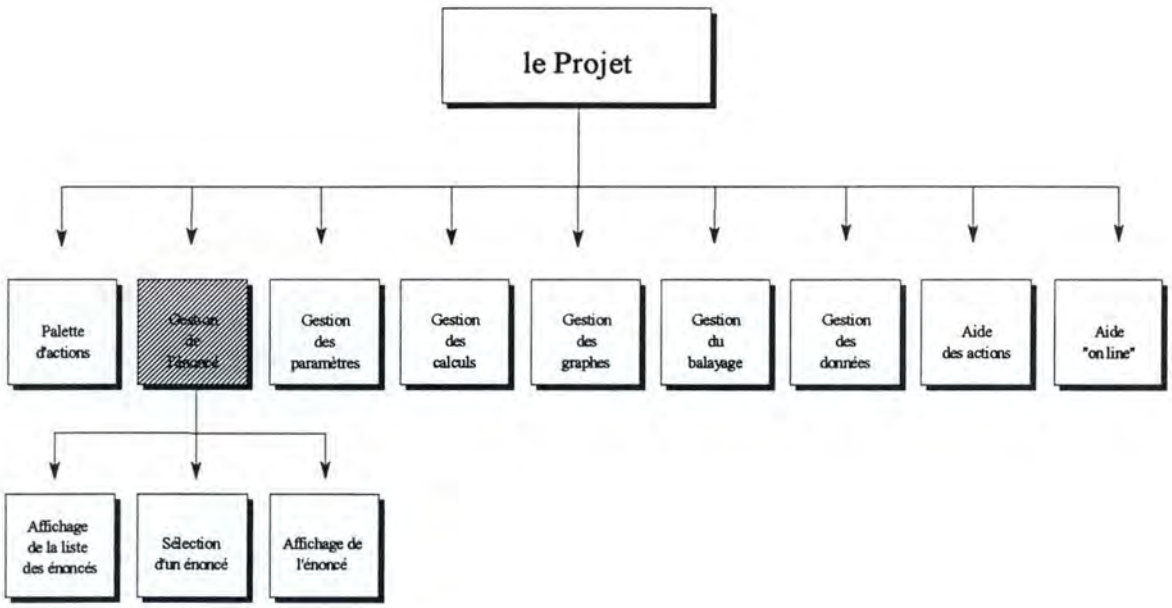


Figure 4.4 : Les fonctions constituant l'unité « Gestion de l'énoncé »

4.3.2.2.1 La fonction: Affichage de la liste des énoncés

Cette fonction présente à l'utilisateur une fenêtre dans laquelle sont affichées une série d'équations à sélectionner.

4.3.2.2.2 La fonction: Sélection d'un énoncé

Cette fonction permet à l'utilisateur de sélectionner une des formes d'équation.

4.3.2.2.3 La fonction: Affichage de l'énoncé

Après avoir choisi une équation, cette fonction l'affiche dans une fenêtre.

4.3.2.3 L'unité: Gestion des paramètres

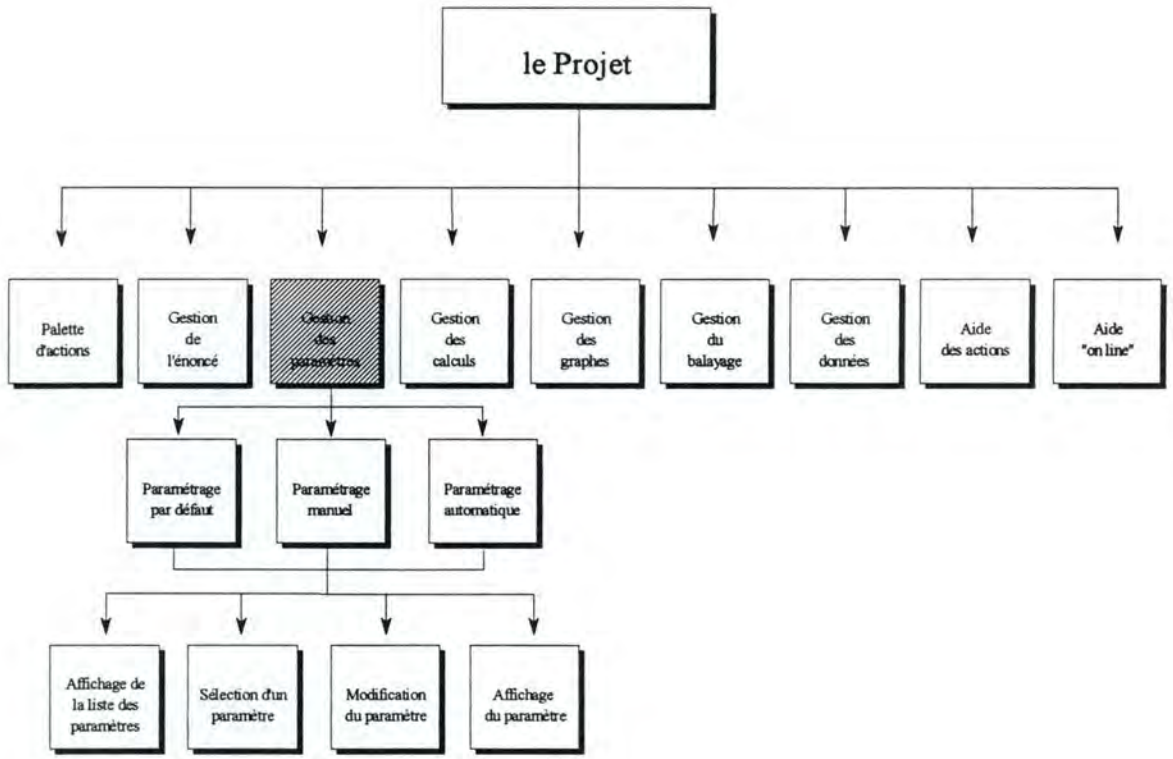


Figure 4.5 : Les fonctions constituant l'unité « Gestion des paramètres »

4.3.2.3.1 La fonction: Paramétrage par défaut

La machine affecte par défaut des valeurs aux paramètres. Ces valeurs seront différentes en fonction du type de fonction à analyser.

4.3.2.3.2 La fonction: Paramétrage manuel

Cette fonction donne l'accès à l'enseignant à la fenêtre des paramètres.

4.3.2.3.3 La fonction: Paramétrage automatique

Cette fonction devra balayer les paramètres de l'équation en vue de modifications par l'élève.

Pour chaque type de paramétrage, ces fonctions devront également pouvoir afficher la liste des paramètres, en permettre la sélection, la modification et l'affichage d'un paramètre donné.

4.3.2.5 L'unité: Gestion des graphes

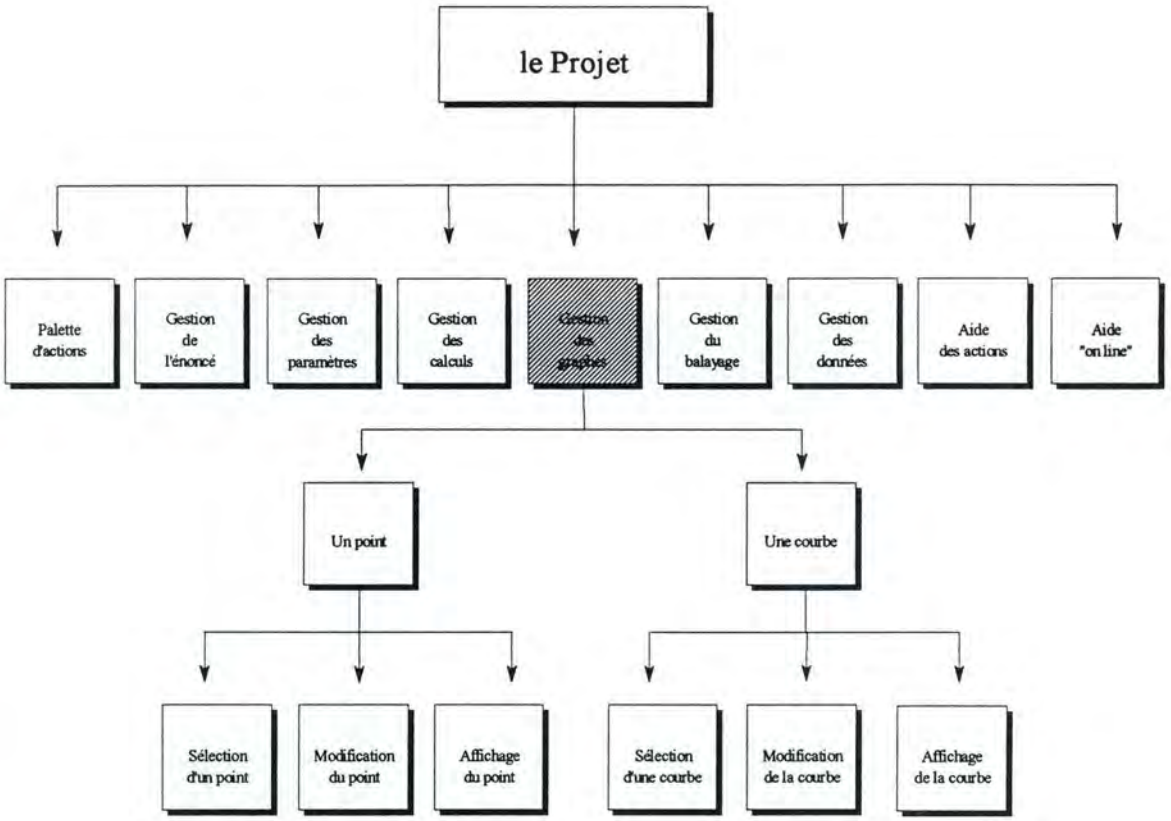


Figure 4.7 : Les fonctions constituant l'unité « Gestion des graphes »

L'ensemble de ces fonctions gère les différents objets dessins créés. Il s'agit de les mémoriser après leur création. Un objet dessin, après sa création, peut se trouver dans deux états : normal ou sélectionné. Ces fonctions s'occupent également des objets dessins qui se trouvent dans l'état « sélectionné », elles en permettent leur déplacement. Elles devront par conséquent créer et gérer une liste de points et une liste de courbes.

4.3.2.5.1 La fonction: Sélection d'une courbe

Cette fonction se charge de la sélection d'une des courbes de dessins qui sont activées par machine.

4.3.3 Liens entre les unités

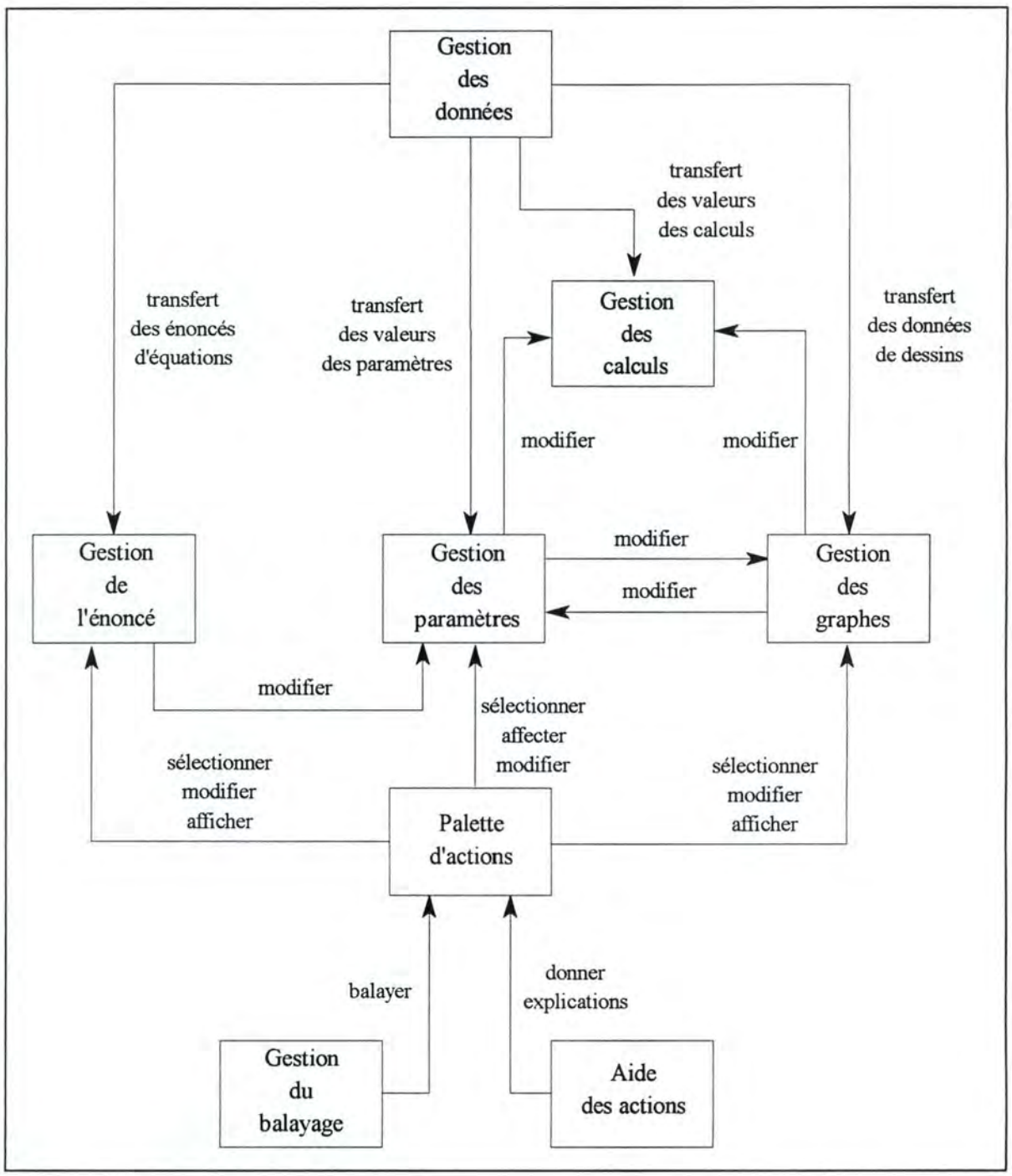


Figure 4.12 : Graphe des liens existant entre les unités de l'application

4.3.4 Les traitements du projet

Les traitements du projet seront définis et détaillés dans la partie implémentation. Ces traitements seront définis en termes de procédures ou méthodes. Pour cela, nous définirons tout d'abord les différents objets des unités du projet et à partir de là, nous en détaillerons leurs différentes méthodes.

5.1 Introduction

La partie implémentation de notre projet concerne les traitements des différentes unités de notre projet.

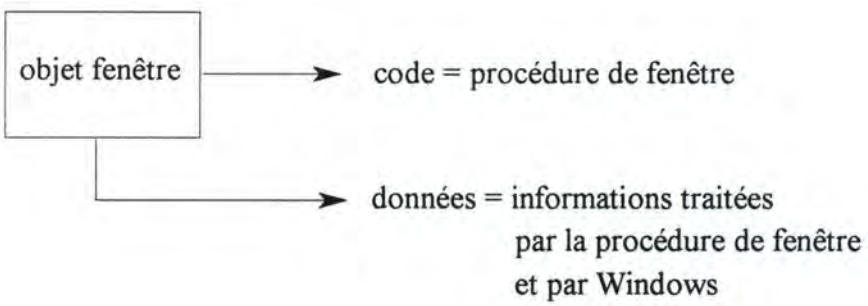
Cette partie a pour objectif de définir et de décrire les différents traitements nécessaires aux fonctions de notre projet. Pour cela, nous devrons avant tout définir les objets de notre logiciel, une fois les objets définis, nous décrirons leurs différents champs et méthodes. Cette partie abordera donc tout l'approche orientée objet de notre logiciel.

5.1.1 Objet fenêtre et procédure de fenêtre

Une fenêtre créée dans le programme (l'objet fenêtre visuelle à l'écran) est associée à une procédure de fenêtre.

Une fenêtre est créée à partir d'un modèle appelé « classe de fenêtre » (exemple : TWindow) qui identifie la procédure de fenêtre. Et cette procédure de fenêtre traite les messages de l'objet fenêtre visuelle à l'écran.

L'objet fenêtre visuelle à l'écran est constitué de code et de données : l'objet est la fenêtre que l'on voit affichée à l'écran, le code est la procédure de fenêtre attachant à l'objet fenêtre et les données sont les informations traitées par la procédure de fenêtre et les informations que gère Windows.



6.3.6 Cinquième type de fenêtre

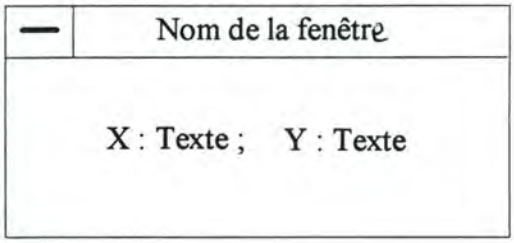


Figure 6.6 : Présentation du cinquième type de fenêtre

Cette fenêtre permet d'afficher les coordonnées d'un point sur l'écran graphique. A chaque déplacement du point sur l'écran graphique, ce dernier voit ses coordonnées changées.

6.3.7 Sixième type de fenêtre

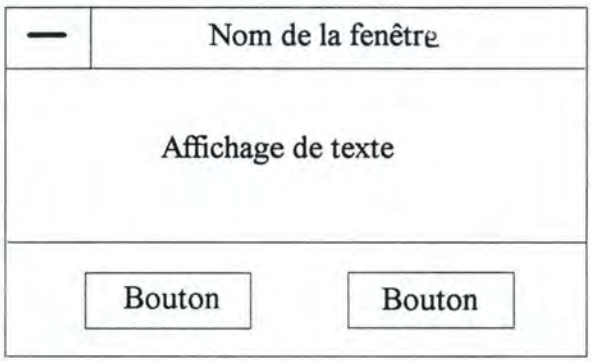


Figure 6.7 : Présentation du sixième type de fenêtre

Ce type de fenêtre remplace la boîte de message que l'on a l'habitude de rencontrer sous Windows. En effet, à la différence d'une boîte de message normale, celle-ci sera dotée en plus d'un balayage automatique.

6.3.8 Septième type de fenêtre

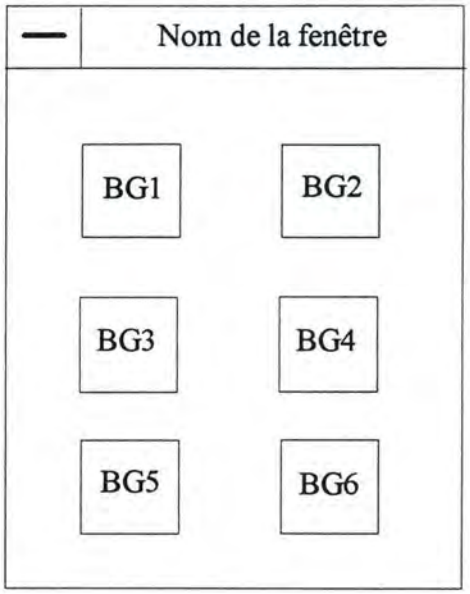


Figure 6.8 : Présentation du septième type de fenêtre

Cette fenêtre joue un rôle très important pour l'élève. Elle peut être vue comme la palette du menu principal de l'élève. Effectivement, cette palette sera constituée de boutons graphiques qui reprendra les principales commandes de la barre du menu principal de l'application. Cette fenêtre est réservée à l'élève handicapé moteur. Pour que ce dernier puisse l'utiliser, chaque bouton graphique sera balayé par le système.

6.4.1.2 La fenêtre d'affichage de l'énoncé

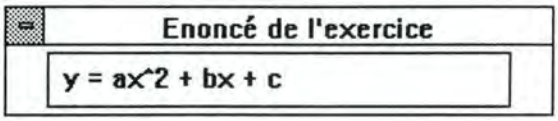


Figure 6.10 : La fenêtre d'affichage de l'énoncé d'exercice

6.4.1.3 La fenêtre des paramètres

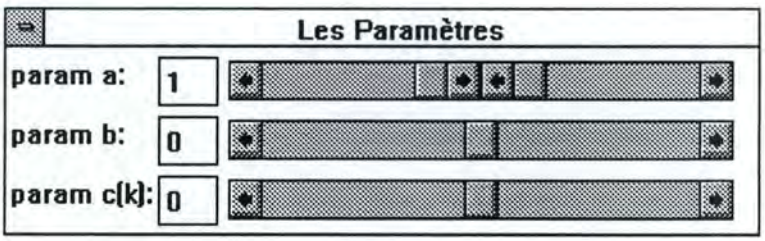


Figure 6.11 : La fenêtre des paramètres

Cette fenêtre permet d'affecter ou de modifier les valeurs des paramètres de la fonction que l'utilisateur est en train de traiter. Elle permet à l'enseignant d'affecter ou de modifier manuellement les valeurs des paramètres. Quant à l'élève, par le déclenchement du balayage à partir de la palette d'actions, il peut soit aller affecter des valeurs aux paramètres, soit en modifier les valeurs.

6.4.1.4 Fenêtre des calculs mathématiques

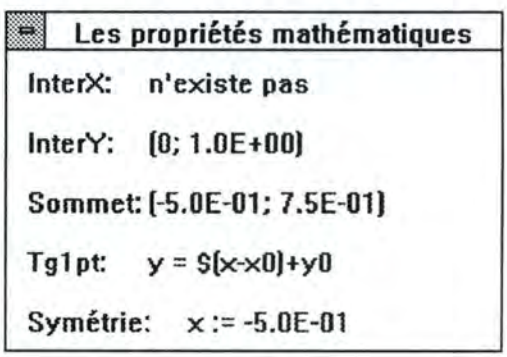


Figure 6.12 : Présentation de la fenêtre des calculs mathématiques

6.4.1.7 La palette d'actions

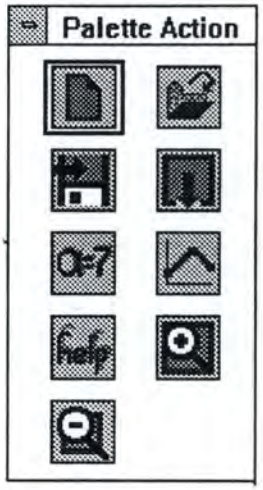


Figure 6.15 : Présentation de la palette d'actions

Cette palette est composée de boutons graphiques qui permettent d'exécuter des actions bien précises. Ces boutons graphiques reprennent quelques commandes de la barre du menu principal de l'application. En effet, le premier bouton permet de « Commencer » un nouvel exercice, le deuxième d'« Ouvrir » un exercice existant, le troisième de« Sauver » un exercice sur disque, le quatrième de « Quitter » le programme, le cinquième de déclencher le balayage de la fenêtre des paramètres en vue d'affecter ou de modifier leurs valeurs, lors des modifications des valeurs des paramètres, le dessin graphique se modifie également, de même que les valeurs de ses propriétés mathématiques. Le sixième bouton déclenche le balayage de la fenêtre des graphes en vue de sélectionner les objets dessins pour les déplacer dans la fenêtre. Les trois derniers boutons correspondent au bouton d'aide de l'application, au bouton de visualisation plus grande et plus petite du dessin graphique.

6.4.2 Enchaînement des fenêtres

6.4.2.1 Introduction

Maintenant que nous avons présenté quelques unes de nos fenêtres avec leurs fonctionnalités, nous allons mettre en schémas les enchaînements de toutes ces fenêtres.

Notre objectif est, d'une part, de mettre sur graphes les différents enchaînements de nos fenêtres afin d'en définir l'ordre d'apparition. De cette manière, nous aurons une vision claire et complète de notre interface. D'autre part, ces graphes nous permettront de définir les liens logiques existant entre toutes les fenêtres, ce qui ne pourra que nous faciliter dans notre tâche de programmation.

6.4.2.2 Légendes des graphes d'enchaînement

Pour chaque graphe, nous donnons la signification suivante :

Le rectangle représente une fenêtre et il porte le même nom que la fenêtre.

Le rectangle arrondi signifie qu'une action est prise en charge par la machine.

La flèche signifie « un item de menu est sélectionné ». Par exemple la sélection d'un des boutons poussoirs d'une fenêtre.

Le texte sans cadre reprend l'intitulé des items des menus.

7.1.3 Au niveau interface

Nous avons défini une interface la plus simple possible, dans la mesure où nous pouvons montrer le lien dynamique qui relie une fonction algébrique et son dessin graphique.

Il faut également observer que l'interface que nous avons proposée ne permet pas encore à l'élève de travailler de manière autonome. En effet, dans l'application, l'enseignant joue encore un rôle important lors de la réalisation d'un exercice de l'élève : c'est lui qui lancera le logiciel et configurera les vitesses de balayage des écrans. L'élève joue encore un rôle passif dans la mesure où il ne peut qu'affecter des valeurs aux paramètres d'une équation algébrique et voir les répercussions sur son dessin graphique et ses propriétés mathématiques.

7.1.4 Au niveau des balayages

Une gestion de la hiérarchie des actions est encore à mettre au point pour que le balayage soit opérationnel de façon unifiée dans toute l'application.

Dans l'état actuel des choses, un menu doté d'un balayage ne fait que parcourir les items qui le composent sans pouvoir encore réagir aux messages de l'élève pour confirmer son choix (les messages que l'élève envoie au système correspondraient au clic du bouton droit de la souris, ce bouton jouant le rôle de contact de l'élève).

7.2 Performance du programme

Il est clair que dans le programme que nous proposons, beaucoup reste encore à faire pour que celui-ci soit entièrement opérationnel et puisse être utilisé par l'élève de manière autonome.

Lors du test de l'application, un arrêt du programme pourrait survenir en raison de la non-gestion des horloges qui définissent le temps de balayage d'un élément d'un menu.

Une certaine lenteur lors de l'exécution de l'application a également été constatée. Ceci est dû au fait que la programmation orientée objet n'est pas encore très bien maîtrisée.

De même, lors de l'affichage dans la fenêtre des graphes et la fenêtre des calculs, une mauvaise gestion des handles peut entraîner la superposition de deux graphes ou celle de deux textes.

L'environnement de sauvegarde du travail sur disque et la configuration des balayages restent encore à mettre au point pour permettre à l'utilisateur de garder une trace de son exercice sur disque ou disquette.

7.3 Les extensions

Vu le caractère didactique du logiciel, nous pourrions également envisager son utilisation par un élève normal. C'est la raison pour laquelle nous pensons que plusieurs extensions du logiciel sont intéressantes à proposer.

7.3.1 Au niveau mathématique

En partant des formes d'équations générales :

$y = ax+b$
$y = ax^2+bx+c$
$y = k*\sin(ax+b)$

proposer d'autres formes de représentations, par exemple :

une droite passant par un point donné (x_0, y_0)
une droite passant par deux points donnés (x_1, y_1) et (x_2, y_2)
une droite ayant un certain angle avec l'axe des X, etc.

Pour plus de détails, voir l'annexe B.

7.4 L'aspect positif du travail

De nos jours, le terme « Enseignement Assisté par Ordinateur » (E.A.O.) nous est devenu familier, ce qui n'était pas le cas il n'y a pas si longtemps que cela. Nous pouvons trouver sur le marché divers didacticiels, les bons et les moins bons. Ces didacticiels sont adressés au grand public comme les élèves, les enseignants, les médecins, les comptables, et bien d'autres. Mais si nous voulons trouver des didacticiels destinés à des personnes handicapées et en particulier à des personnes handicapées moteur, cela devient plus rare. Ceci est dû au fait qu'un didacticiel, pour pouvoir être utilisé par un utilisateur handicapé, doit être doté d'un outil supplémentaire, celui du balayage.

C'est la raison pour laquelle, la réalisation de ce mémoire à été très intéressante car elle nous a permis d'aborder différents problèmes liés à l'enseignement assisté par ordinateur : celui lié à l'enseignement, celui lié à la programmation et celui lié à l'utilisation. C'est la rencontre de l'informaticien avec le non-informaticien dans un domaine commun qui est l'enseignement.

Personnellement, l'élaboration du projet m'a permis de tenir les trois rôles en même temps. Pour faire l'analyse des besoins, je devais me mettre à la place de l'enseignant et celle de l'utilisateur. Pour l'implémenter, je devais me mettre dans la peau du programmeur et pour la tester, je devenais l'utilisateur.

Pendant ce travail, j'ai connu des moments d'enthousiasme mais aussi des moments difficiles. L'enthousiasme venait au moment où j'ai su que mon travail serait utile à certaines personnes. C'était aussi au moment de ma visite au Centre d'Enseignement et de Traitements Différenciés (C.E.T.D.), là j'ai pu rencontrer des personnes formidables qui se sont mises à la disposition des enfants handicapés. J'ai également eu l'occasion de faire la connaissance d'un élève du centre Paul qui avait bien voulu me montrer son univers de travail. C'était à ce moment-là que je m'étais réellement rendue compte du rôle joué par l'informatique pour ces personnes.

Passé ce moment d'enthousiasme, il a fallu se mettre au travail. Et c'est le moment qui sera qualifié de difficile car avoir un sujet de mémoire, c'est une chose mais réfléchir aux moyens pour le réaliser s'en est une autre. C'est donc le moment de recherche et de lecture puisque la programmation orientée objet a été choisie pour implémenter notre application. Or de l'approche orientée objet, je n'en ai qu'une connaissance théorique. Il a donc fallu que j'apprenne à maîtriser les concepts que l'approche met en avant et un langage de programmation orientée objet, le Borland Pascal. C'est une période de travail intense, car pour bien comprendre ces concepts, une autre vision des choses était nécessaire, celle de la modélisation du monde en objets.

Une autre difficulté du projet est la conception du logiciel car les personnes à qui est destinée notre application sont des personnes particulières, il a donc fallu penser à leur environnement de travail, celui géré par le balayage.

Les difficultés rencontrées étaient de plusieurs niveaux : il fallait garder en tête l'aspect didactique du projet et essayer de se mettre à la place des utilisateurs, penser à la façon d'implémenter les interfaces, d'une part pour l'enseignant et d'autre part pour l'élève, le tout dans la vision de la programmation orientée objet.

Pour résumer, les trois niveaux de difficultés ont été :

Le niveau pédagogique (le rôle tenu par l'enseignant)
Le niveau handicap (l'utilisateur)
Le niveau de la programmation orientée objet (le programmeur)

Ce qui serait formidable, c'est que les objectifs soient atteints et donc notre application opérationnelle pour que les enseignants et les élèves puissent l'utiliser.

Personnellement, s'il fallait refaire un choix, le mien resterait le même car je pense que l'informatique peut apporter tellement de choses, pour ne fût-ce faciliter la vie de ces personnes en leur rendant un peu plus autonomes dans leur environnement de travail.

Table des matières

Table des figures