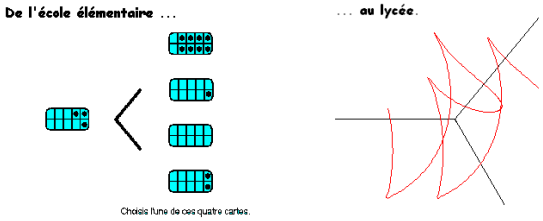
Manuel Logoplus



http://logoplus.pagesperso-orange.fr

Préambule :

Le langage LOGO est un langage de programmation informatique qui représente l'un des aboutissements d'un projet éducatif Nord-Américain destiné à relancer l'intérêt des mathématiques et des sciences parmi la jeunesse américaine. Il est d'abord la simplification d'un autre langage de programmation anglo-saxon, le langage Lisp, développé par un chercheur du M.I.T. (Massachusetts Institute of Technology), John McCarthy en 1958. Cette simplification a été réalisée en 1966 par un autre chercheur-informaticien, Wallace Feurzeig, qui lui a donné son nom actuel: le langage LOGO. D'autres informaticiens du MIT se sont ensuite regroupés autour de W. Feurzeig pour définir les spécificités de ce langage : une syntaxe avec peu de conventions d'écriture, facile à utiliser et qui peut à la fois permettre d'aborder les concepts mathématiques que les problèmes non numériques. C'est ainsi que sous la houlette d'un des membres de l'équipe LOGO, le professeur de mathématiques Seymour Papert, le langage LOGO se répandit dans les écoles nord-américaines, dont la plupart possédaient déjà des ordinateurs en réseaux. Cependant, les théories Piagétiennes qui sous-tendaient son action, basées sur l'auto-construction libre du savoir chez l'enfant, n'incitèrent pas les enseignants à imaginer des méthodes pour faire utiliser par les élèves le langage LOGO comme outil de transversalité des compétences dans un environnement pédagogique "encadré". De l'autre côté de l'Atlantique, le début des années quatre-vingts voit apparaître une version du langage LOGO dotée d'une syntaxe française développée par deux sociétés sous-traitantes de Thompson, LCSI (Quebec) et Act informatique (France). Les élèves de l'école élémentaire de cette époque apprennent à maîtriser et programmer ce nouvel objet technologique qu'était alors l'ordinateur, dans le cadre du plan Informatique Pour Tous (IPT) mis en place par la législature du moment. Cette première mouture française du langage LOGO, appelée LOGOPLUS, fonctionna essentiellement sur des ordinateurs Thompson tels que les TO7 ou MO5 assemblés en nano-réseaux. Leur fragilité et leur utilisation partagée à la fois entre les écoles et les municipalités (pendant le hors-temps scolaire), sans précaution, précipita le déclin du langage LOGO et de la "petite programmation" dans l'Hexagone. Plusieurs années après, l'arrivée de l'internet orienta l'informatique à l'école élémentaire plus vers le domaine du traitement de texte, la recherche documentaire ou l'utilisation de logiciels ludiques que vers la création scientifique grâce à des outils de développement adaptés. A la fin des années quatre-vingt-dix, le constat était patent : les élèves n'avaient plus aucune notion de "comment on fait résoudre des problèmes par les ordinateurs", contrairement à leurs prédécesseurs dix ans plus tôt, alors que les ordinateurs envahissaient de plus en plus la réalité de leur vie quotidienne, souvent domestique. C'est à ce moment, en 1999 quand LOGO avait totalement sombré dans l'oubli en France, que j'ai pris l'initiative, à contre-courant des nouveaux programmes et de ce fait, inaperçue de l'administration scolaire mal préparée à anticiper les implications pédagogiques des innovations scientifiques qui apparaissent dans la société et déjà à cette

époque occupée à répartir les contenus didactiques des enseignements principaux sur plusieurs cycles d'apprentissage, de développer moi-même, en reprenant au tout début la syntaxe française du LOGO des années quatre-vingts, une version moderne et rafraîchie qui bénéficierait des progrès des ordinateurs parallèlement au système Windows alors en pleine évolution, et dont j'ai conservé la désignation "Logoplus" pour maintenir une sorte de continuité. C'est cette version démultipliée du langage LOGO, au final considérablement enrichie par des possibilités motivantes (mathématiques, robotique, musique, graphique) tout en préservant les caractéristiques d'un langage de programmation de haut niveau, que je vous propose d'utiliser ou mieux, de faire utiliser par vos enfants ou vos élèves, avec un avantage éducatif, scientifique et culturel évident dans leur scolarité ainsi que dans leur compréhension du monde moderne, immédiat et sur le long terme.

Dominique Bille

Table des rubriques.

I. Les fenêtres :

I. Les fenêtres :

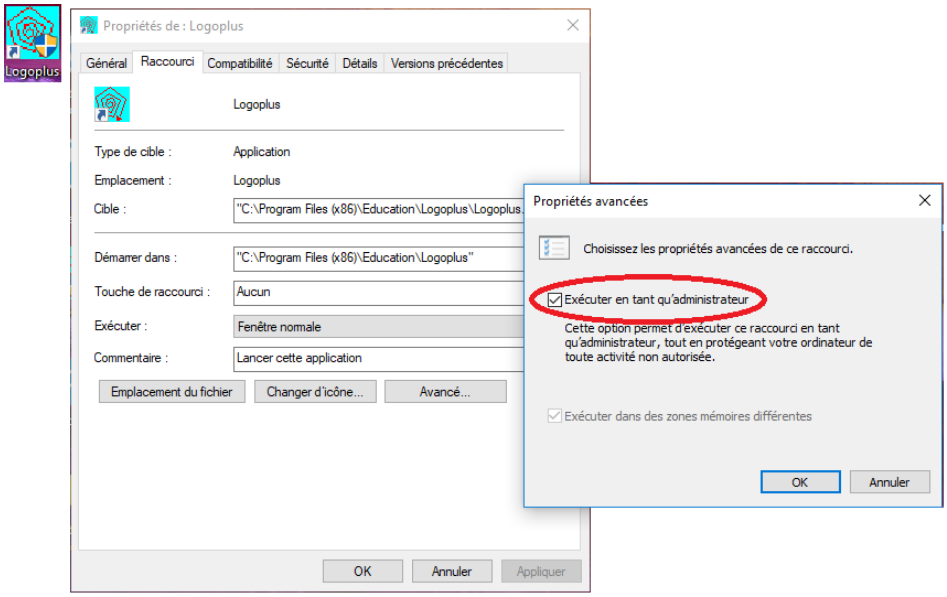
Logoplus comporte 28 fenêtres :

I.1. La fenêtre de présentation :

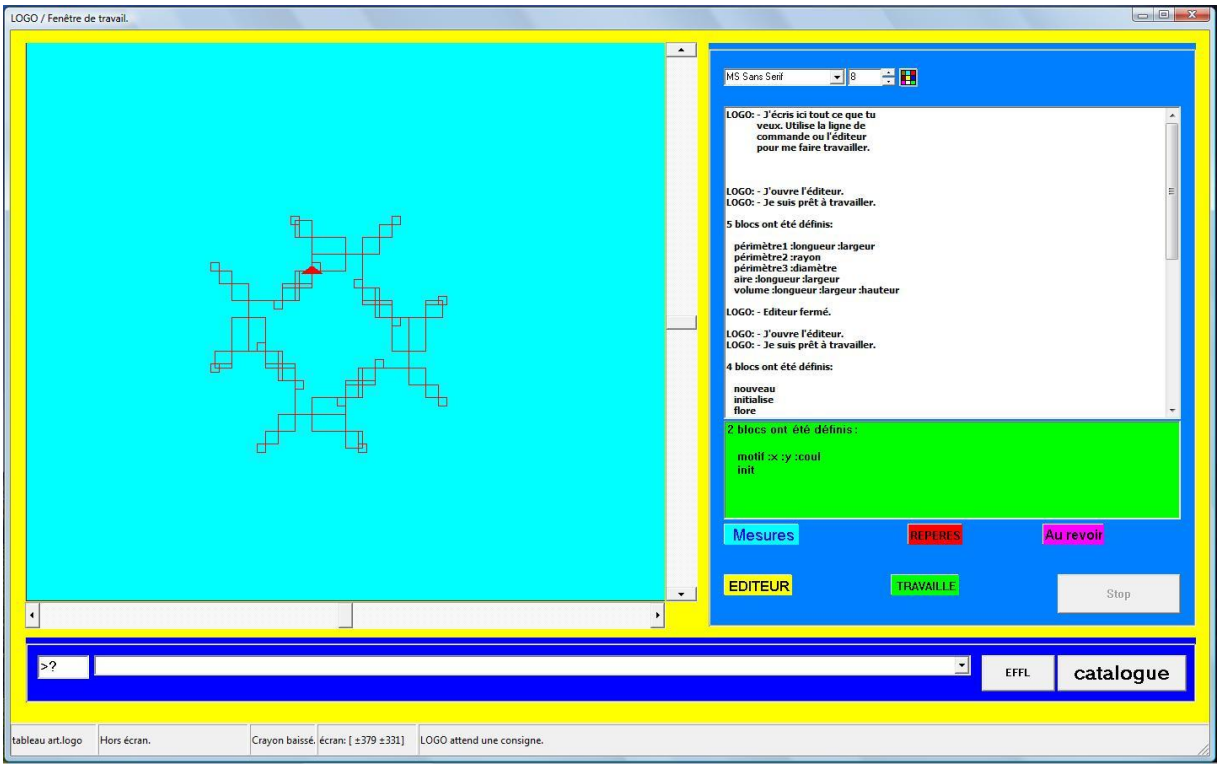
Retour à la table des rubriques >>

Elle apparaît en premier lorsqu'on double-clique sur l'icône du raccourci Logoplus située sur le bureau. Elle comprend deux informations importantes : le numéro de la version sous le format JJ.MM.HH.MN.AN et l'adresse Internet du site de téléchargement des mises à jour de Logoplus.

Conseil : pour faciliter les accès au disque de Logoplus pour les versions Windows 7 et suivantes, il est recommandé de cocher l'option "Exécuter en tant qu' administrateur" dans les propriétés du raccourci de Logoplus situé sur le bureau, comme le montre l'image ci-dessous.



I.2. La fenêtre de travail :

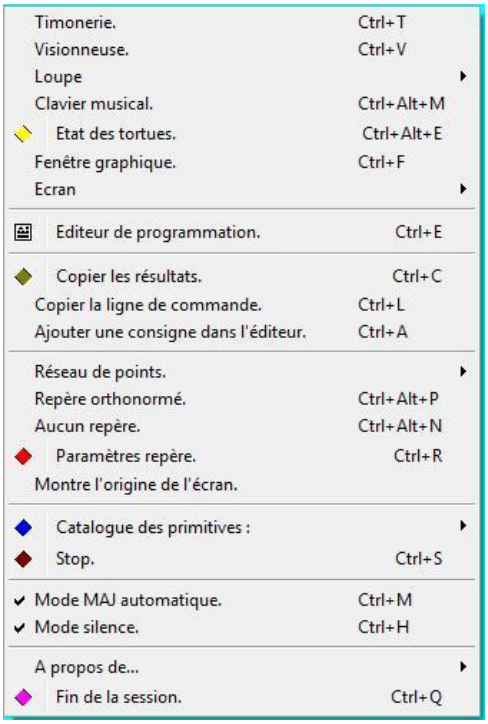


Cette fenêtre comprend plusieurs fonctions et se découpe en six zones :

- L'écran, en bleu clair situé ici à gauche de la fenêtre, permet de visualiser les résultats graphiques de LOGO. Il est aussi appelé "écran des tortues".
- L'afficheur des résultats, en blanc à droite de la fenêtre, affiche les écrits de LOGO. Il indique également ce que LOGO a fait ou est en train de faire. Il instaure également un semblant de dialogue avec l'utilisateur.
- La liste des blocs définis, en vert, permet de lancer LOGO sur un travail.
- Les boutons Mesures, Repères, Editeur et Travaille dont l'utilisation est indiquée dans la suite du document.
- La ligne des commandes qui permet de passer des consignes à LOGO et de le lancer sur un travail. A sa droite, un bouton Effl pour effacer la(les) consigne(s) présente(s) sur la ligne de commande, ainsi que le bouton Catalogue, qui ouvre une fenêtre qui regroupe l'ensemble des consignes (on dit aussi primitives) LOGO.

- Le barre d'état tout en bas de la fenêtre de travail, indique le nom du programme contenu dans l'éditeur de programmation (voir plus loin), les coordonnées de la souris sur l'écran graphique, les dimensions en pixels de l'écran graphique ainsi que le statut de LOGO relativement à son travail.

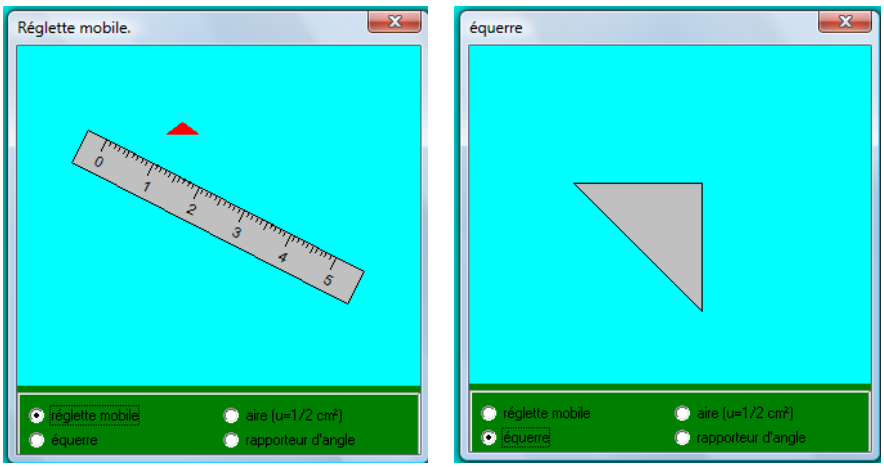
I.2.a. Son menu flottant :

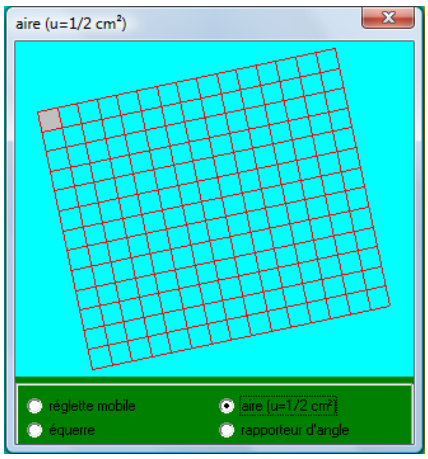


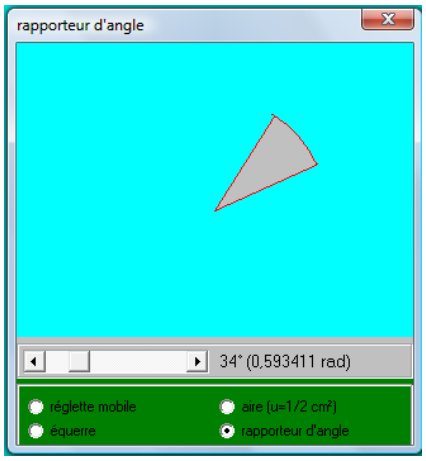
Le menu flottant reprend tout ou partie des boutons qui figurent sur la fenêtre de travail mais également d'autres options et/ou utilitaires qui ne se trouvent que sur cette version du langage LOGO. Ces options sont illustrées dans la suite de ce document.

Retour à la table des rubriques >>

I.3. La fenêtre de mesures :

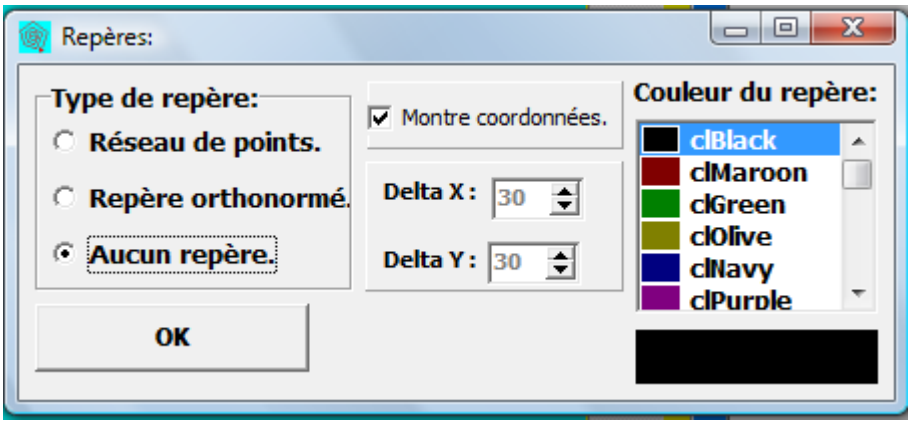






D'un simple clic sur l'une des cases à cocher, cette fenêtre prend l'aspect de l'un des quatre instruments de mesure en usage dans les écoles : la réglette, l'équerre, les gabarits de mesure de surface et de report d'angle.

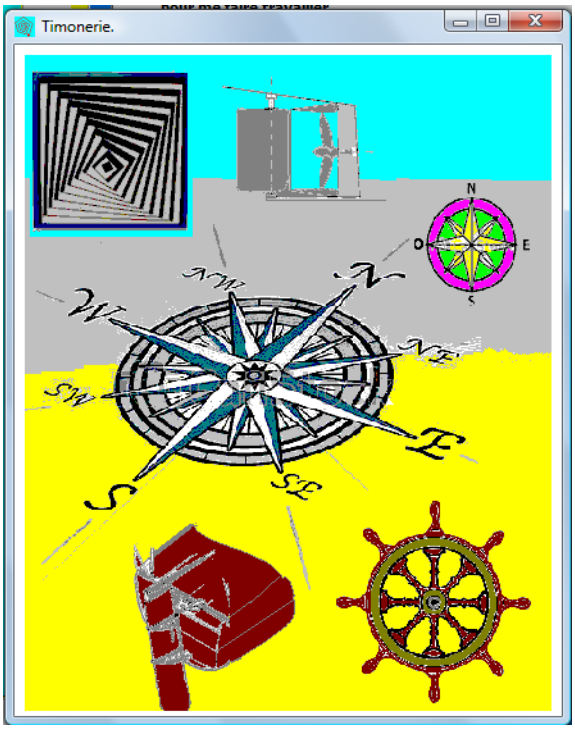
I.4. La fenêtre des repères :



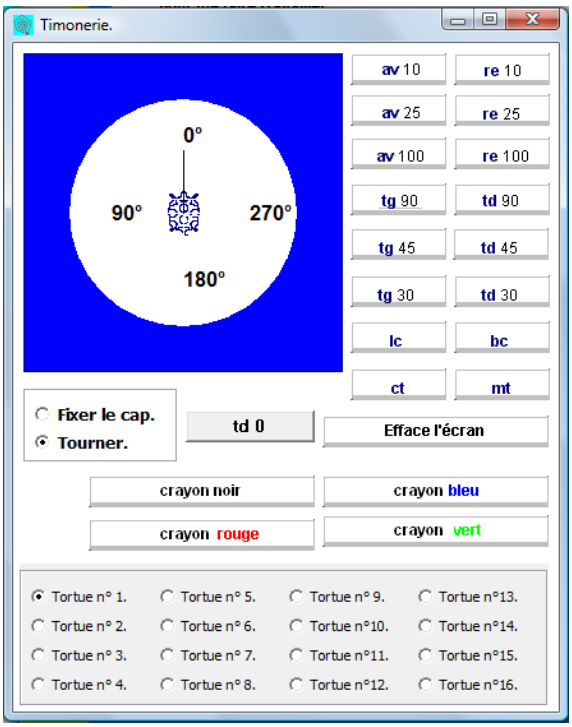
Cette fenêtre permet d'indiquer si on souhaite ou non structurer l'écran des tortues par un repère, constellé de points ou parcouru par une mire et de paramétrer l'espacement du réseau constitué ainsi que fixer sa couleur.

Retour à la table des rubriques >>

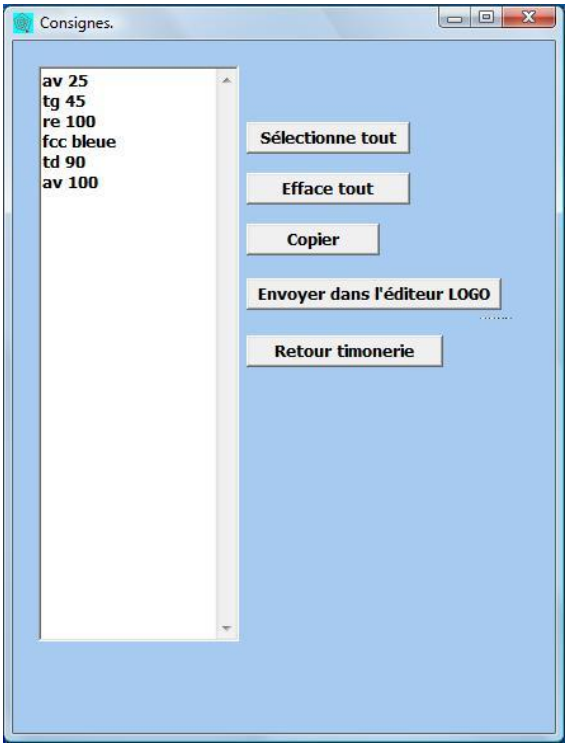
I.5. La fenêtre de timonerie :



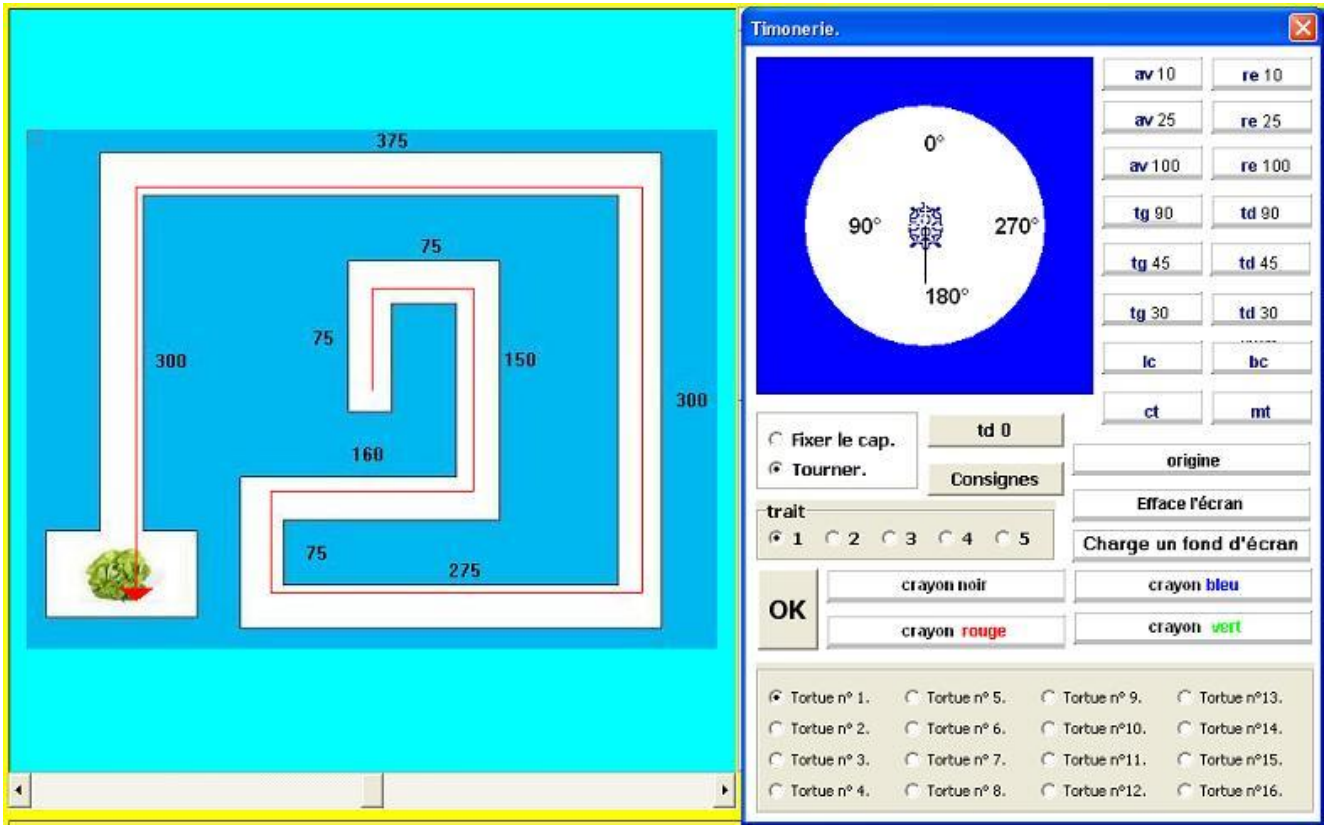
1. à l'ouverture



2. La timonerie.



3. La liste des consignes déjà effectuées.

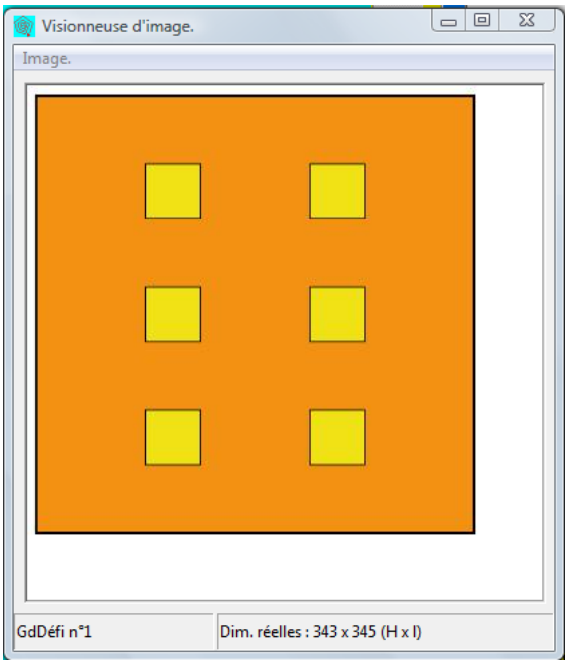


4. Un fond d'écran (ici un labyrinthe) permet de se familiariser avec les déplacements de la tortue et les angles de façon ludique.

Comme pour la fenêtre des mesures, la fenêtre de la timonerie permet de piloter l'une des seize tortues LOGO d'un simple clic de la souris. Elle change d'aspect lorsqu'on veut lister l'ensemble des consignes déjà passées et les envoyer dans l'éditeur de programmation.

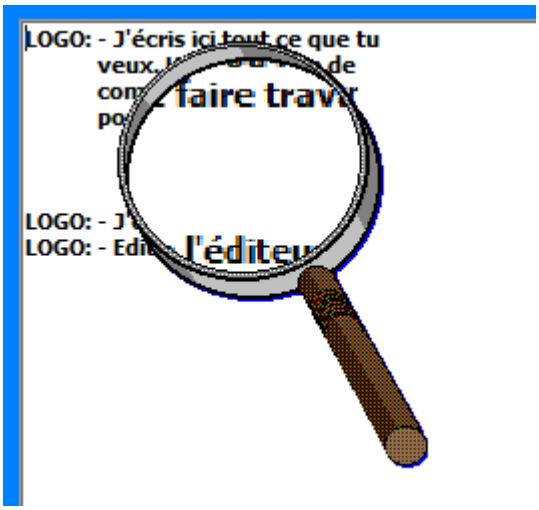
Retour à la table des rubriques >>

I.6. La visionneuse d'image :



Très utile pour afficher une figure ou un dessin géométrique que l'on souhaite faire reproduire en LOGO. Le système de fichier de cette fenêtre fait d'abord afficher la liste des images qui se trouvent dans le répertoire "Défis en image".

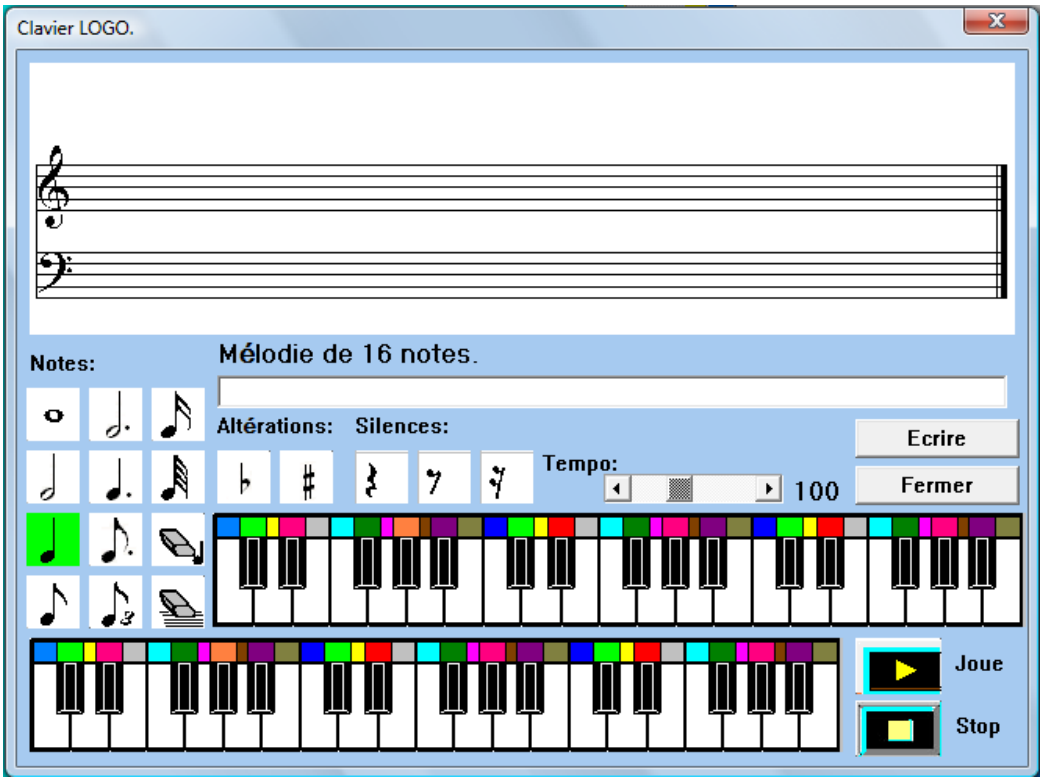
I.7. La fenêtre "loupe" :



Cette fenêtre, en dépit de son apparence, est l'un des outils très précieux qui n'équipent que cette version du langage LOGO. Elle offre plusieurs grossissements, mais ne fonctionne que sur la fenêtre de travail.

Retour à la table des rubriques »

I.8. La fenêtre du clavier musical :



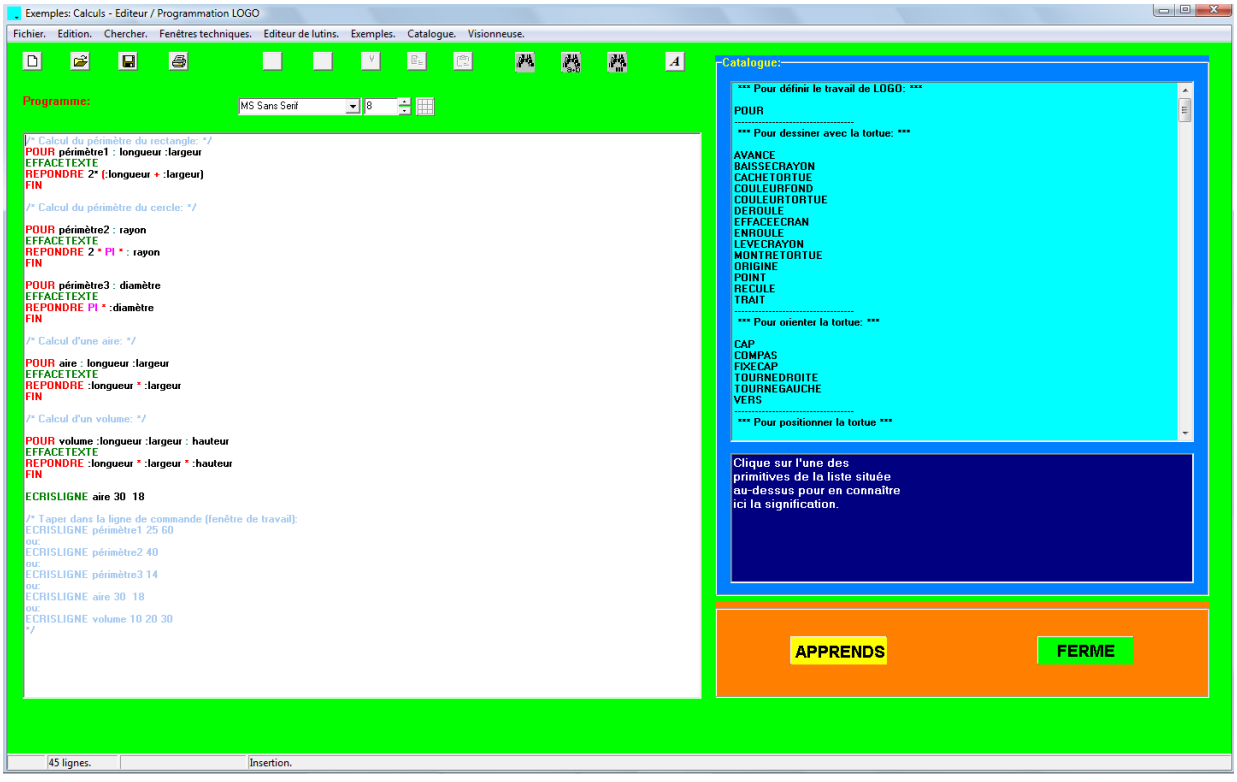
D'un simple clic sur l'une des touches, LOGO fera entendre la note souhaitée et la codera dans une chaîne de caractères traduite en langage LOGO. Limité à 16 notes, cet éditeur musical permet aussi de jouer sur la durée des notes et le tempo de la mélodie.

I.9. La fenêtre d'état des tortues :

Cette fenêtre affiche l'état de chacune des 16 tortues (ou agents) de LOGO et donne la possibilité de modifier certains de leurs paramètres.

Retour à la table des rubriques >>

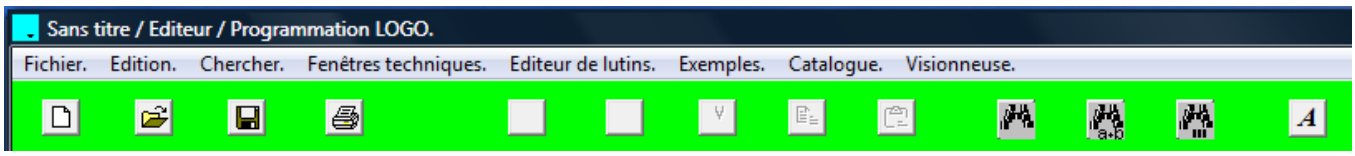
I.10. La fenêtre de programmation :



C'est, en fait, LA fenêtre principale du langage LOGO et c'est aussi la raison d'être de ce langage de programmation. Elle se découpe en quatre zones :
- L'éditeur de programmation proprement dit, à gauche en blanc. On y aperçoit un texte de programmation en LOGO dont la syntaxe est colorisée.
- A droite, en bleu clair, le catalogue des primitives liste l'ensemble des mots-clés du langage LOGO. Lorsque l'option "catalogue sensitif" est cochée, un clic sur l'une des primitives la transfère instantanément dans l'éditeur à la position actuelle du curseur. Sa définition est alors affichée dans la troisième zone en bleu foncé :
- La zone de définition des primitives.
- La zone des boutons "APPRENDS" et "FERME". Un clic sur le bouton "APPRENDS" indique à LOGO qu'il y a un texte à analyser en prévision de sa réalisation (son calcul). Un clic sur le bouton " FERME" fait revenir l'utilisateur sur la fenêtre de travail.

Retour à la table des rubriques >>

I.10.a. Le bandeau des menus :



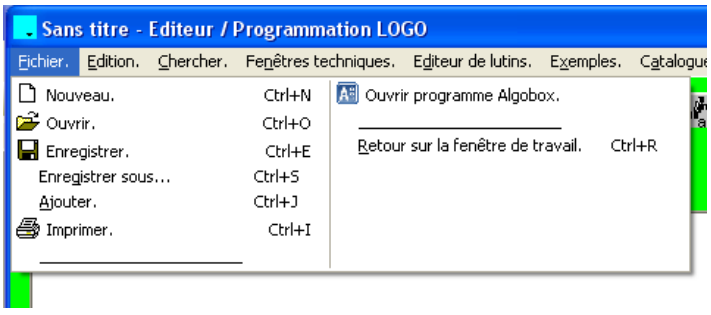
Comprend les options Fichier, Edition, Chercher comme un traitement de texte classique. Les options Fenêtres techniques, Editeur de lutins, Exemples, Catalogue et Visionneuse sont illustrées plus loin dans ce document.

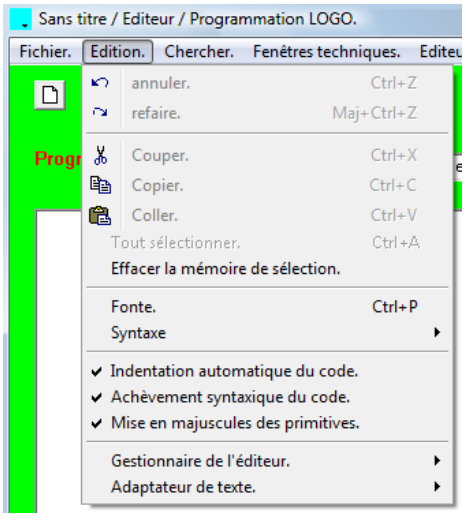
I.10.b. Les boutons d'édition :

De gauche à droite, on trouve les options des fichiers : "Nouveau", "Ouvrir", "Enregistrer" et "Imprimer". Ensuite les options de traitement de texte : "Annuler ", "refaire", "Couper", "Copier" et "Coller", "Chercher", "Remplacer", "Occurrence suivante", " Fonte".

Retour à la table des rubriques >>

I.10.c. Les menus Fichier et Edition :

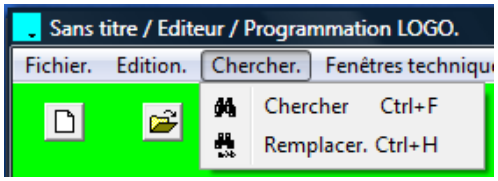


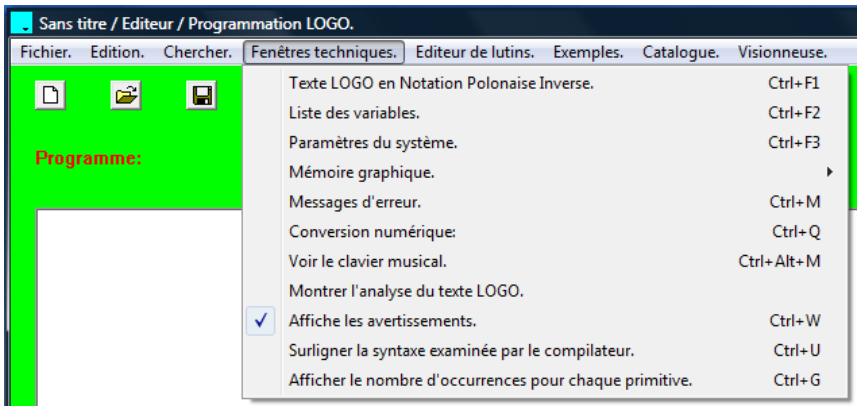


Le menu "Fichier " reprend les options classiques de gestion de fichiers sous Windows ; Le menu "Edition" comporte deux options qui seront illustrées plus loin : "Gestionnaire de l'éditeur " et "Adaptateur de texte".

A remarquer toutefois dans le menu " Fichier" une option "Ouvrir programme Algobox" qui permet d'importer un projet Algobox transformé au préalable en programme LOGO (le programme en Algobox figurera cependant dans un commentaire pour comparaison).

I.10.d. Les menus Chercher et Fenêtres techniques :

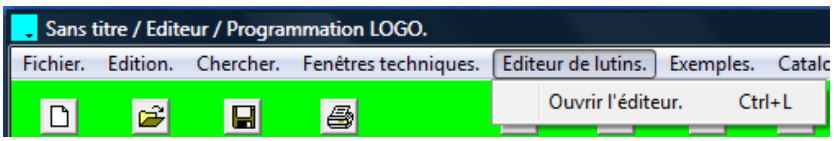




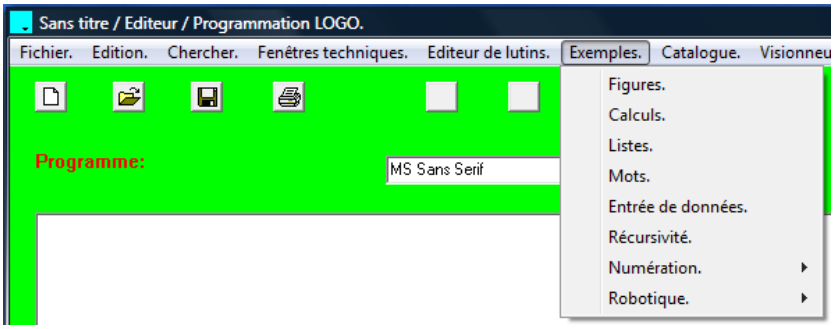
Le menu "Chercher" comprend les options "Chercher" et "Remplacer" classiques d'un traitement de texte. Le menu "Fenêtres techniques" permet à l'utilisateur d'accéder aux différents paramètres (variables, Pile des calculs, messages d'erreurs), les tracés sur l'écran des tortues ainsi qu'à d'autres fenêtres (clavier musical et conversion décimal<->hexadécimal). Une option permet à LOGO de compter pour chaque primitive (mot-clé) d'un programme LOGO, le nombre de fois qu'elle aura été rencontrée et effectuée.

Retour à la table des rubriques >>

I.10.e. Les menus Editeur de lutins et exemples :

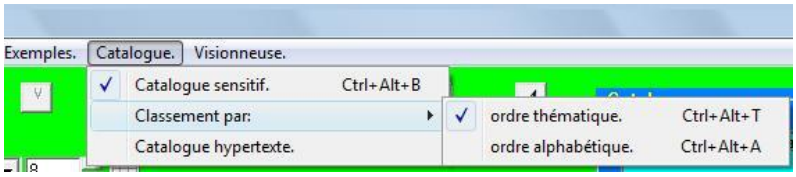


Comme vous le verrez plus loin, les lutins sont des petites figures qui peuvent se déplacer sur l'écran des tortues, mais un peu comme s'ils étaient des fantômes et doués d'une certaine autonomie. Grâce à l'éditeur des lutins, on peut dessiner la forme qu'ils prendront.

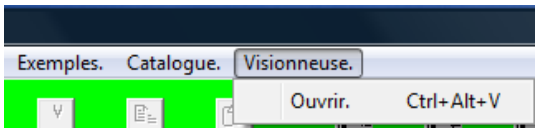


Plusieurs exemples de ce menu permettent très facilement de se familiariser avec la syntaxe LOGO.

I.10.f. Les menus Catalogue et Visionneuse :



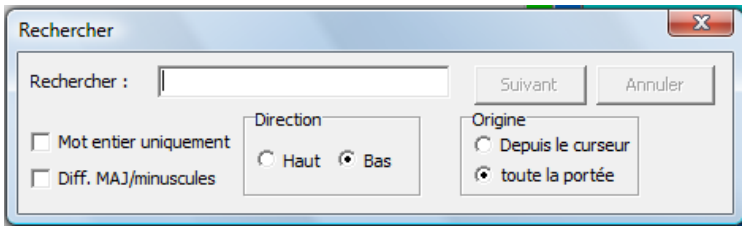
L'option "Catalogue sensitif" rend la liste des primitives de la fenêtre de programmation réactive aux clics de la souris. Il est possible de classer les primitives dans l'ordre alphabétique ou par groupes de fonctions. Un catalogue doté de liens hypertextes peut être affiché en aide et permet de voyager d'une primitive à l'autre en observant leur définition.

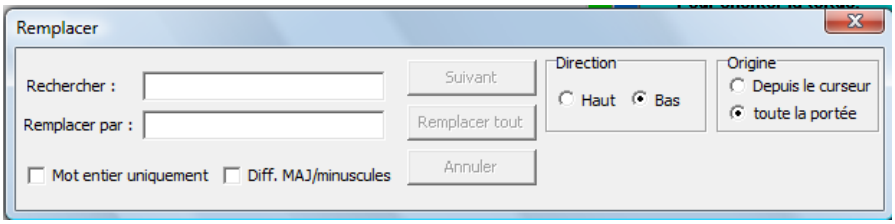


On retrouve l'option "Visionneuse" déjà rencontrée dans la fenêtre de travail.

Retour à la table des rubriques >>

I.11. Les fenêtres Rechercher et Remplacer :



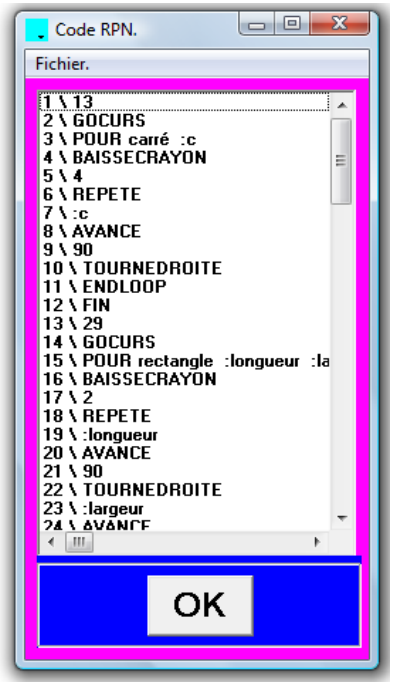


Deux fenêtres à l'aspect classique mais bien utiles dans la recherche de mots ou de lignes de textes et de leur remplacement.

Retour à la table des rubriques >>

I.12. La fenêtre Notation Polonaise Inverse (RPN) :

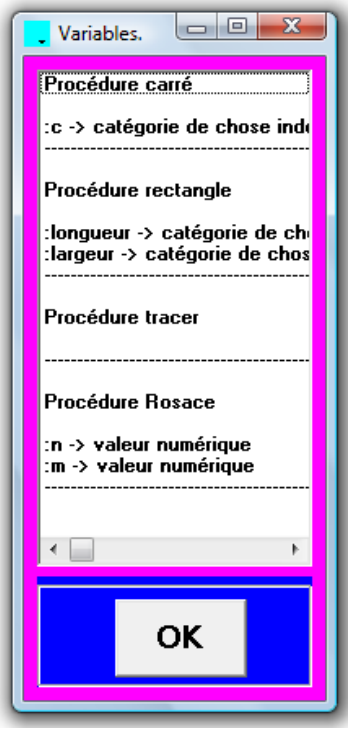
```
/* Tracé d'un carré: */

POUR carré :c
BAISSECRAYON
REPETE 4 [AVANCE : c TOURNEDROITE 90]
FIN

/* Tracé d'un rectangle: */

POUR rectangle :longueur :largeur
BAISSECRAYON
REPETE 2 [
       AVANCE : longueur TOURNEDROITE 90
       AVANCE : largeur TOURNEDROITE 90
      ]
FIN

POUR tracer
EFFACEECRAN carré 50 COLORIE [5 10] ROSE
ATTENDS 2 LEVECRAYON AVANCE 70 ATTENDS 2
BAISSECRAYON rectangle 80 30 COLORIE [5 130] JAUNE
FIN

POUR Rosace :n :m
EFFACEECRAN BAISSECRAYON
REPETE :m [
     REPETE 4 [
            AVANCE :n TOURNEDROITE 90 DONNE "n :n + :m
           ]
      ECRISLIGNE : n
 TOURNEDROITE 20
       ]
CACHETORTUE
FIN

Rosace 50 6
```



Lorsque LOGO analyse un texte, il examine soigneusement chaque mot qu'il rencontre et détermine dans quel ordre il doit l'effectuer, qui n'est souvent pas dans l'ordre dans lequel nous lisons ces mots. Cet ordre s'appelle la "Notation Polonaise Inverse" et elle consiste à placer les données d'une opération en premier et l'opérateur en dernier. La fenêtre RPN visualise cet ordre et indique au programmeur qui rencontre des difficultés sur un programme LOGO complexe, si ses primitives ont été bien analysées en LOGO et s'il y a lieu de modifier l'écriture de son programme.

I.13. La fenêtre d'édition des variables :

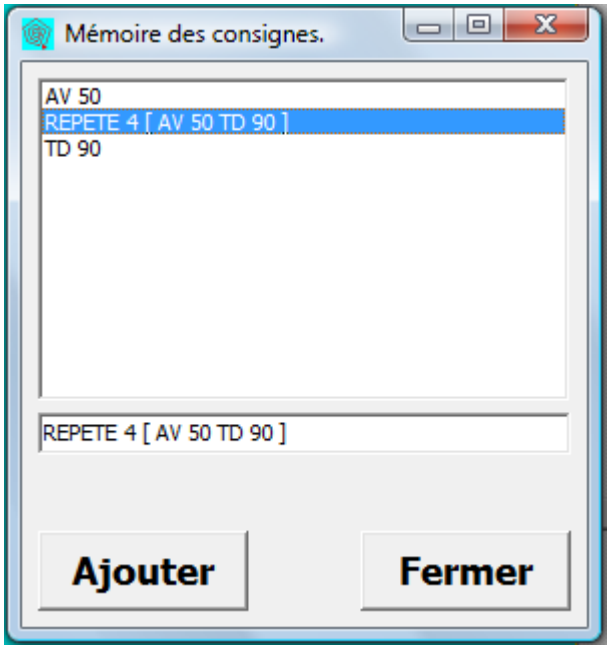
```
/* Tracé d'un carré: */

POUR carré :c
BAISSECRAYON
REPETE 4 [AVANCE : c TOURNEDROITE 90]
FIN

/* Tracé d'un rectangle: */

POUR rectangle :longueur :largeur
BAISSECRAYON
REPETE 2 [
       AVANCE : longueur TOURNEDROITE 90
       AVANCE : largeur TOURNEDROITE 90
      ]
FIN

POUR tracer
EFFACEECRAN carré 50 COLORIE [5 10] ROSE
ATTENDS 2 LEVECRAYON AVANCE 70 ATTENDS 2
BAISSECRAYON rectangle 80 30 COLORIE [5 130] JAUNE
FIN

POUR Rosace :n :m
EFFACEECRAN BAISSECRAYON
REPETE :m [
     REPETE 4 [
            AVANCE :n TOURNEDROITE 90 DONNE "n :n + :m
           ]
      ECRISLIGNE : n
 TOURNEDROITE 20
       ]
CACHETORTUE
FIN
```



Cette fenêtre "variables" regroupe les données utilisées par chaque bloc de calcul et indique éventuellement son type (nombre, liste, mot, couleur, booléen, lutin) si LOGO à réussi à le déterminer.

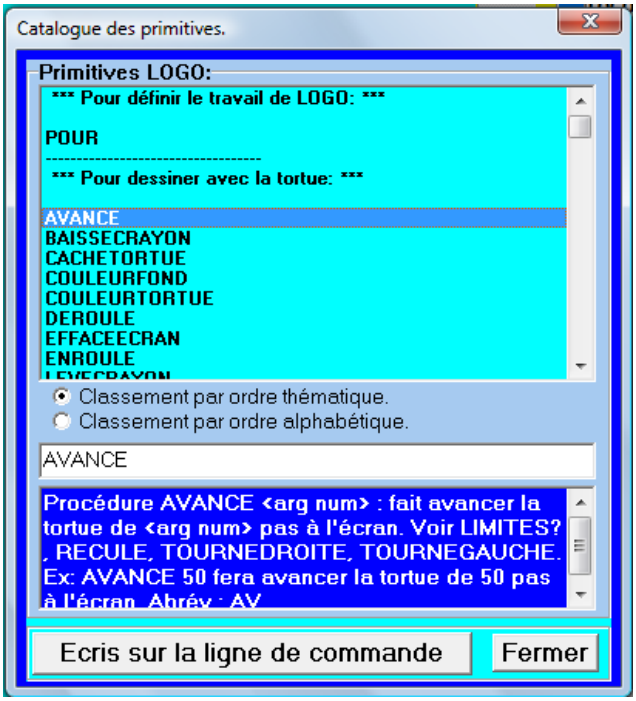
Retour à la table des rubriques »

I.14. La fenêtre-mémoire des consignes :



Cette fenêtre mémorise sous forme de liste les consignes qui ont été effectuées depuis la ligne de commande (fenêtre de travail). Elle est accessible depuis l'option du menu flottant de la fenêtre de travail "Ajouter une consigne dans l'éditeur" .

I.15. La fenêtre du catalogue des primitives :



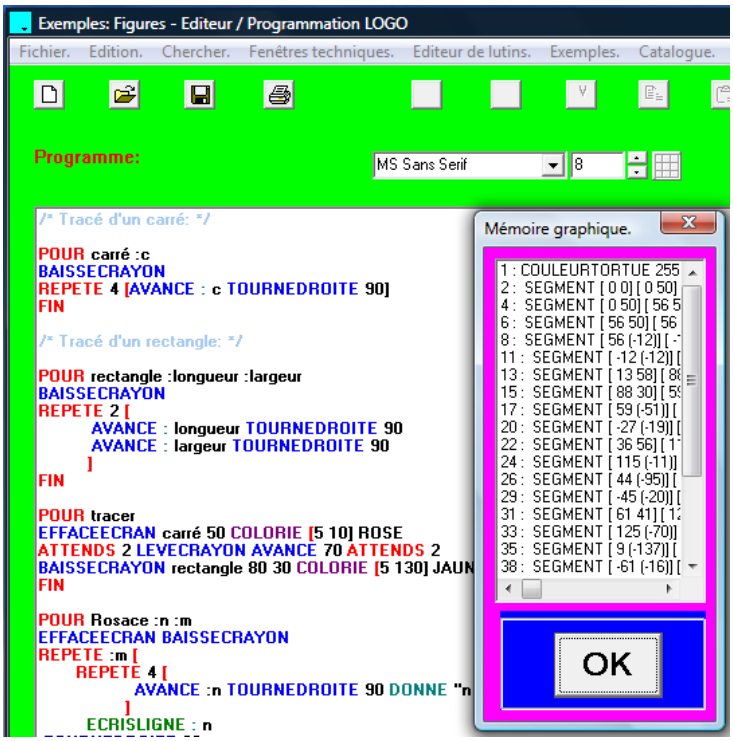
On retrouve dans cette fenêtre une zone familière de la fenêtre de l'éditeur de programmation. Ici, un clic du bouton gauche de la souris a suffi pour indiquer son rôle ainsi que sa syntaxe. Là, l'utilisateur a la possibilité d'ajouter cette primitive à la ligne de commande. Noter que l'option "classement des primitives" y est aussi présente.

Retour à la table des rubriques >>

I.16. La fenêtre des registres internes du système :

Cette fenêtre intitulée "Infos système" visualise les adresses des lignes d'appel et de retour des blocs de calcul d'un programme LOGO réalisé, la pile des registres de calcul ainsi que le nombre d'arguments invoqués à chaque appel. Cette fenêtre a été utile lors du développement de Logoplus mais ne l'est plus à présent. Elle a cependant été conservée par intérêt pédagogique à destination des futurs développeurs de logiciels.

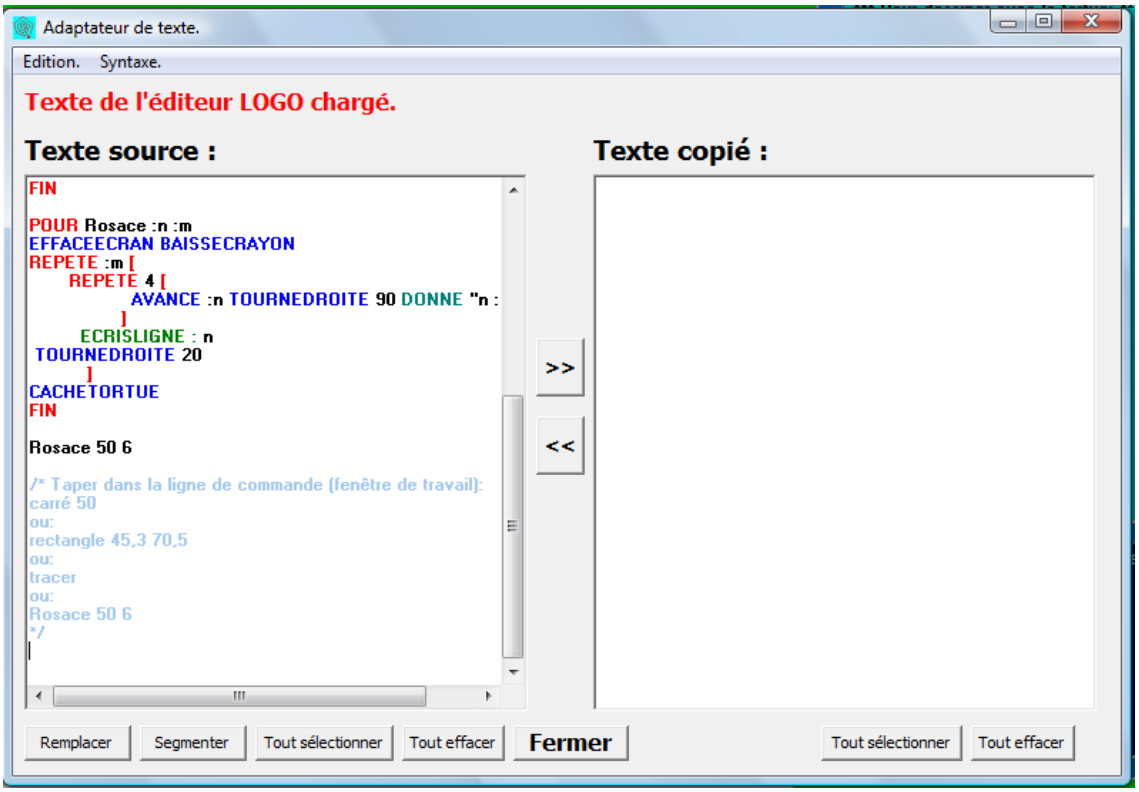
I.17. La fenêtre mémoire graphique :



Cette fenêtre établit la liste des tracés successifs des tortues, sous la forme de segments écrits en LOGO. Sur cette image, elle se situe à droite de l'éditeur de programmation.

Retour à la table des rubriques >>

I.18. La fenêtre de l'adaptateur de texte:



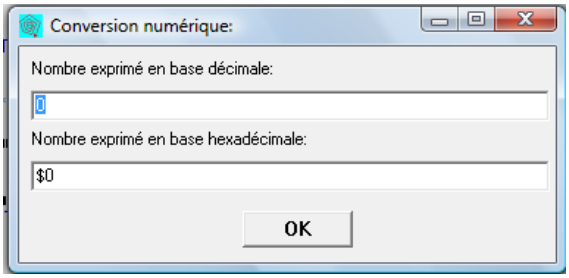
Cette fenêtre est accessible depuis le menu « Edition » de la fenêtre de l'éditeur de programmation . Elle peut cependant être invoquée par le système "WatchDogText" (WDT) à l'ouverture d'un fichier-programme lorsqu'il n'est pas reconnu comme un programme LOGO valide. Elle permet de modifier un texte en dehors de l'éditeur de programmation, de segmenter automatiquement chaque primitive qui s'y trouve ainsi qu' espacer chaque bloc de calcul par une ligne vide. Elle donne aussi la possibilité d'évaluer la lisibilité d'un texte en tant que programme LOGO éventuel et de traduire en français les principales primitives anglo-saxonnes qui s'y trouveraient (et réciproquement).

I.19. La fenêtre de signalement des erreurs :

Elle n'apparaît automatiquement que lorsque le déroulement d'un programme LOGO est interrompu soit par l'utilisateur, soit par une erreur inopinée. Un message indique sur quelle primitive se situe l'erreur et sur quelle ligne dans la Notation Polonaise Inverse (RPN).

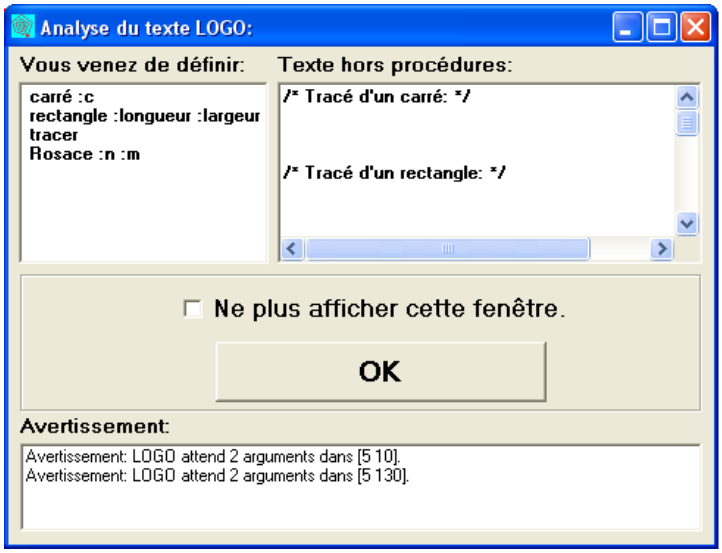
Retour à la table des rubriques >>

I.20. La fenêtre de conversion décimal/hexadécimal :



Logoplus accepte également les données numériques écrites sous forme hexadécimale. Cette fenêtre permet l'affichage simultané d'une valeur numérique à la fois en notation décimale et en notation hexadécimale.

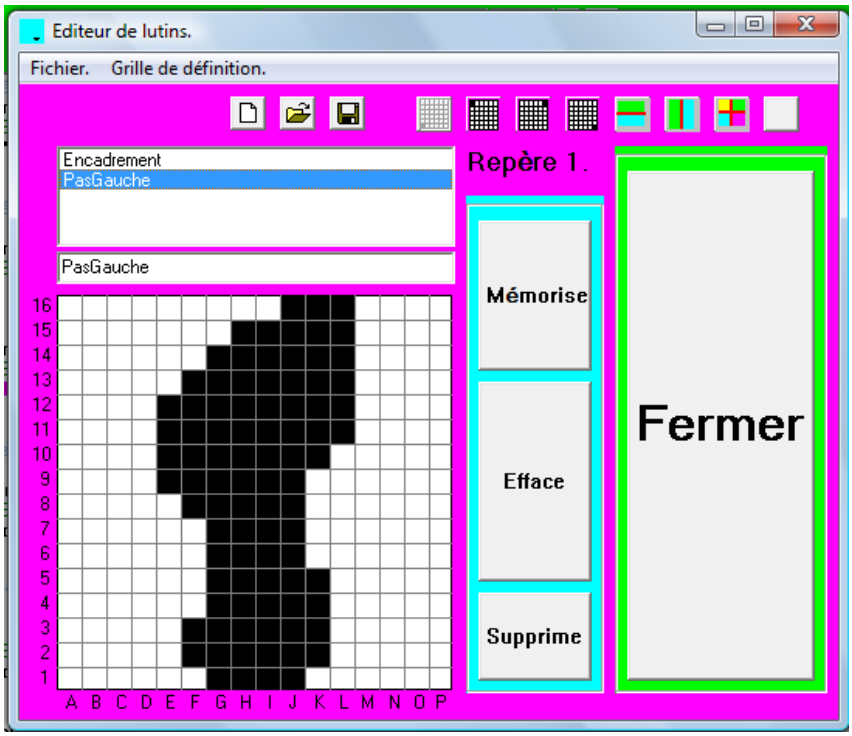
I.21. La fenêtre d'analyse du texte de programmation :



Cette fenêtre indique que la phase d'apprentissage (analyse du texte) s'est déroulée sans erreur et que Logoplus a compris le travail qu'on lui a indiqué de faire. La partie gauche de la fenêtre indique les en-têtes des blocs de calculs définis dans votre programme. A droite, on trouve les lignes LOGO définies à l'extérieur des blocs, c'est-à-dire définies globalement, ce qui est souvent le cas pour les variables partagées.

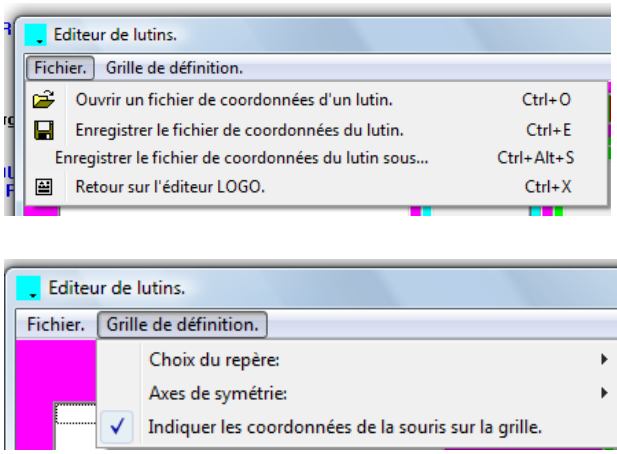
Retour à la table des rubriques >>

I.22. La fenêtre d'édition des lutins :



On aperçoit dans cette fenêtre la grille de définition d'un lutin. Chaque case se colorie alternativement en noir ou en blanc d'un clic de la souris. Il est alors possible d'enregistrer un lutin en le nommant ou de le rappeler en sélectionnant son nom dans la liste située au-dessus de la grille de définition. LOGO ne pourra apprendre (ou analyser) un programme qui désigne un lutin qu'à la condition que ce lutin figure dans la liste de l'éditeur des lutins. Dans le cas contraire , LOGO indiquera qu'il ne connaît pas le lutin invoqué dans le texte de programmation. Grâce aux boutons situés en haut et à droite de cette fenêtre, il est possible de modifier les coordonnées des cases de la grille de définition des lutins ou de définir un ou plusieurs axes de symétrie.

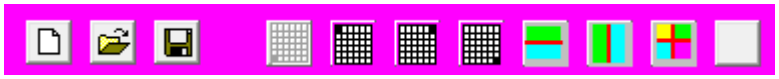
I.22.a. Les menus de la fenêtre d'édition des lutins :



Ces menus reprennent les actions des boutons qui figurent en haut et à droite de la fenêtre d'édition des lutins.

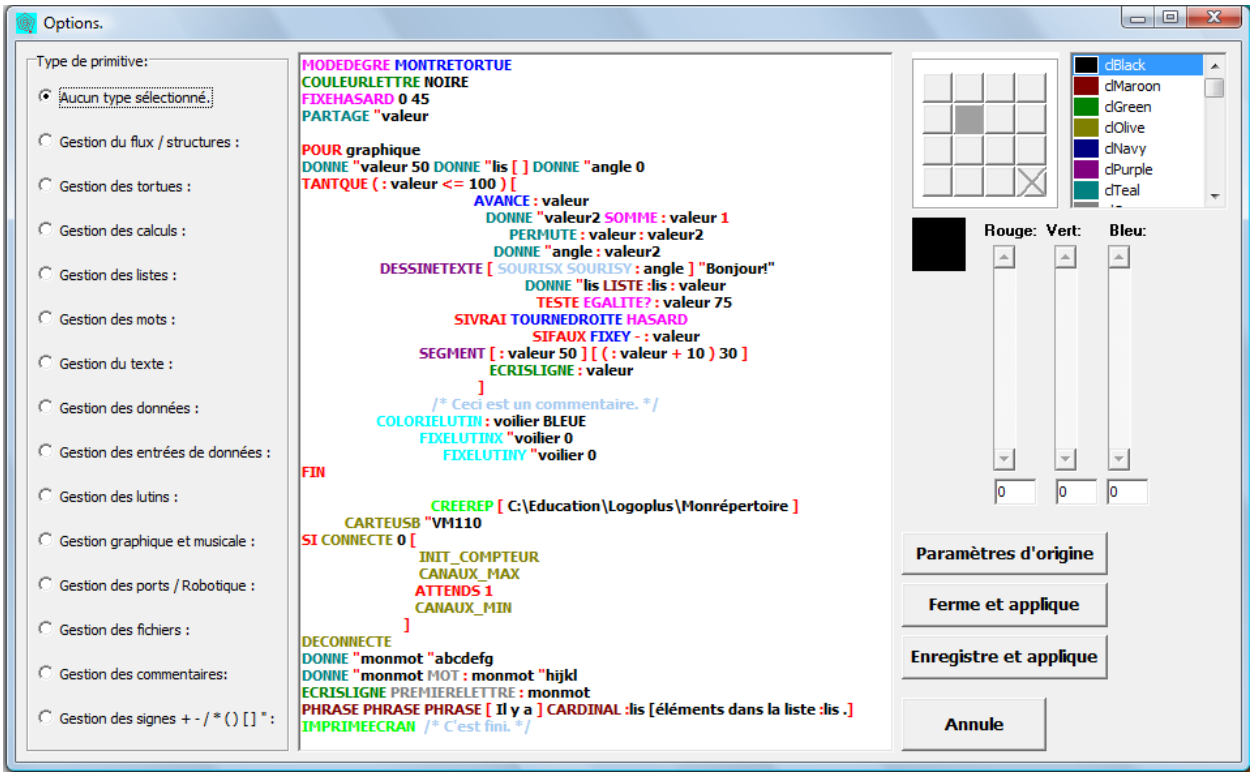
Retour à la table des rubriques >>

I.22.b. Les boutons d' édition de la fenêtre d'édition des lutins :



1. Gestion des fichiers 2. Repérage sur la grille 3. Axes de symétrie

I.23. La fenêtre des options du coloriage syntaxique de l'éditeur de programmation :



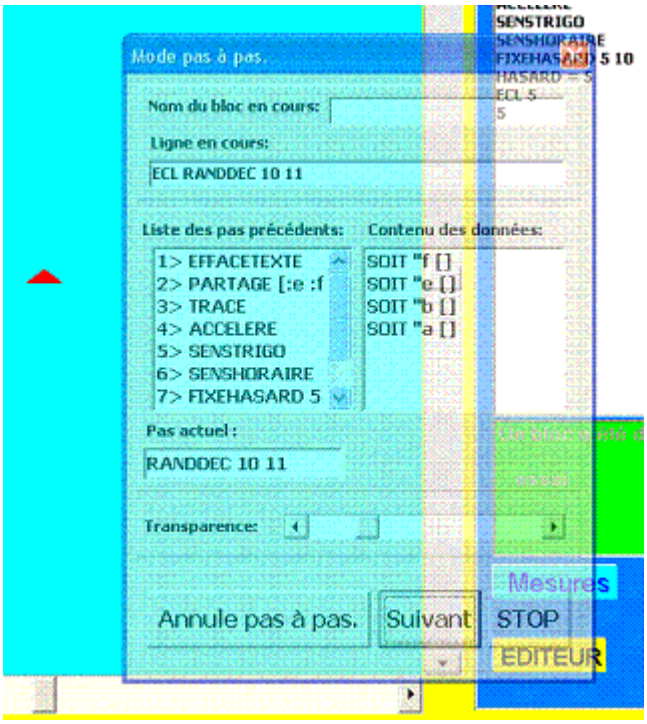
La fenêtre "Options" permet de sélectionner pour chaque type de primitives une couleur donnée. Il faut donc cocher une des 14 options de type sur le panneau situé à gauche de la fenêtre avant de choisir la couleur à lui attribuer. Cela peut être fait de deux manières : l'une en sélectionnant la couleur dans la boîte à liste en haut et à droite de la fenêtre, l'autre en manipulant les curseurs des trois couleurs fondamentales. Il ne reste ensuite plus qu'à confirmer votre choix en cliquant sur l'un des boutons en dessous des glissières. Le bouton "Annule" ferme la fenêtre sans tenir compte des modifications effectuées.
Initialement :

Gestion du flux/structures : rouge
Gestion des tortues : bleu outremer
Gestion des calculs : rouge
Gestion des listes : marron
Gestion des mots : gris
Gestion du texte : vert foncé
Gestion des données : vert foncé
Gestion des lutins : bleu clair
Gestion graphique et musicale : violet
Gestion des ports/ robotique : noir
Gestion des fichiers : vert clair
Gestion des commentaires : bleu-gris
Gestion des signes : rouge

*** Les fenêtres programmables. ***

Les fenêtres qui suivent ont été regroupées sous l'appellation de "fenêtres programmables" car elles font l'objet de primitives LOGO dédiées et peuvent être visibles grâce à elles.

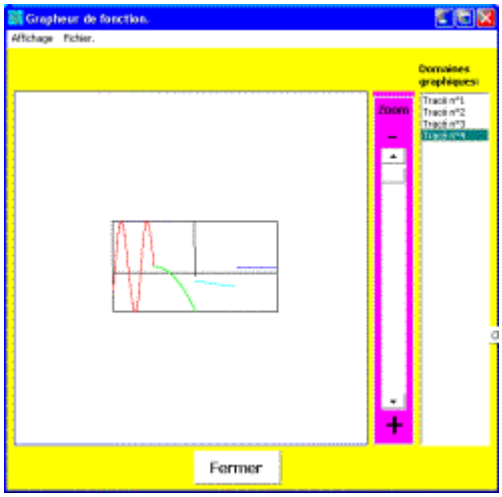
I.24. La fenêtre "Pas à pas".



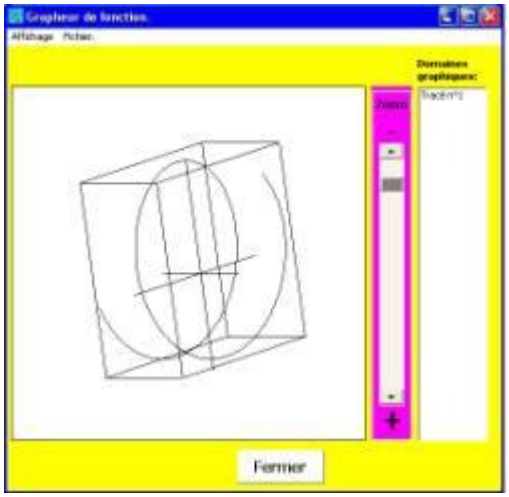
Cette fenêtre ne peut d'abord s'ouvrir que si le mode "Pas à pas" est activé, c'est à dire si la primitive **PASAPAS VRAI** (ou **PASAPAS** 1) a été rencontrée auparavant. C'est la condition préalable mais pas suffisante, car celle-ci ne fait qu'activer une série d'autres primitives (voir Catalogue > PASAPAS), qui lorsqu'elles sont rencontrées à leur tour, interrompent le déroulement du programme en cours et affichent en demi-transparence la fenêtre "Pas à pas" avec plusieurs informations utiles : le nom de la procédure s'il y a, la ligne de code LOGO exécutée, la liste des primitives qui ont fait l'objet d'une interruption momentanée, les variables utilisées avec leurs valeurs contenues ainsi que la primitive actuellement exécutée. La translucidité de la fenêtre, réglable par le curseur, permet de superposer l'action réalisée par le programme et les informations affichées dans la fenêtre. Génial, non ?

I.25. La fenêtre de tracé des fonctions.

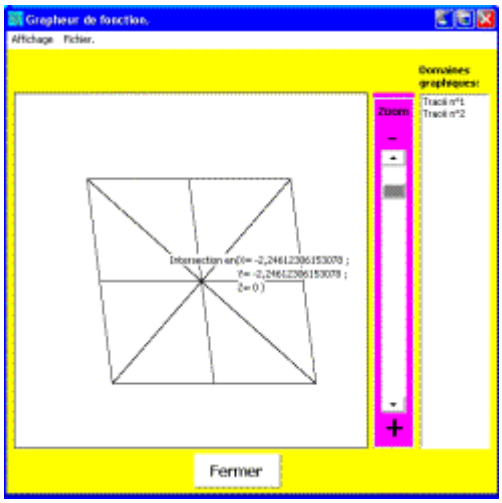
Douze primitives LOGO donnent la possibilité d'afficher cette fenêtre : CHAMPVECT, INITGRAPH , GRAPH2D, GRAPH3D, GRAPHE, GRAPHEUR, PLOT, PLOTPOL, SEGMENT3D, SURFACE, SYSPLOT et SYSPLOTPOL. Même si les valeurs d'une fonction se situent toutes sur un plan, il est possible de faire pivoter son graphe ainsi que modifier son éloignement (fonction zoom). Deux options d'affichage existent. L'une permet de choisir d'afficher un graphe autour de son barycentre (coordonnées relatives) ou dans un repère cartésien (coordonnées absolues). La deuxième permet d'indiquer ou non les graduations des coordonnées pour chaque repère. Ces graduations vont par demi-unité lorsque c'est possible. Plusieurs tracés peuvent être superposés tant que INITGRAPH n'a pas été invoqué. Une option du menu de la fenêtre donne accès à l'enregistrement sur disque de l'image du(des) tracé(s). A noter que cette fenêtre ne peut être ouverte par une option du menu de la fenêtre de travail que lorsqu'un ou plusieurs tracés y sont déjà présents.



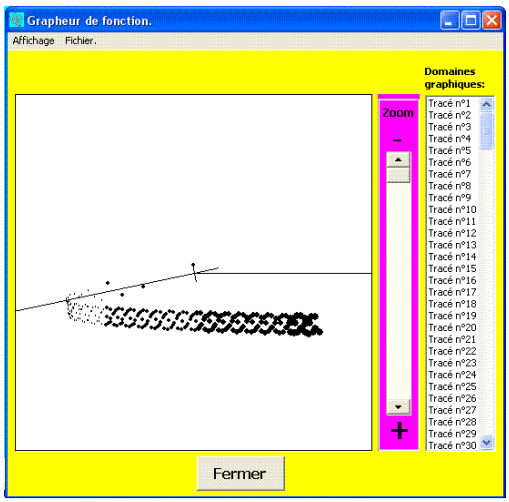
1. Tracé de plusieurs fonctions planes.
Observez la colorisation de chaque tracé.



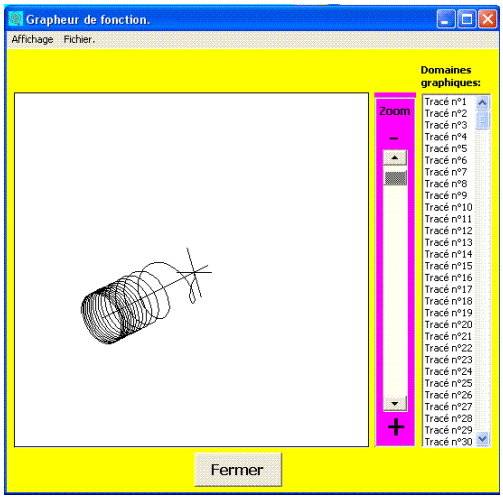
2. Tracé de fonction paramétrique. (GRAPH3D)



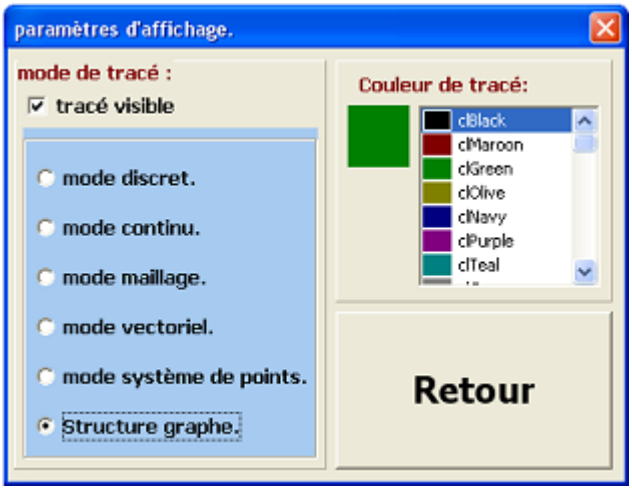
4. Localisation d'un point d'intersection par un double-clic sur la zone graphique proche (inédit sur les calculatrices scientifiques) .



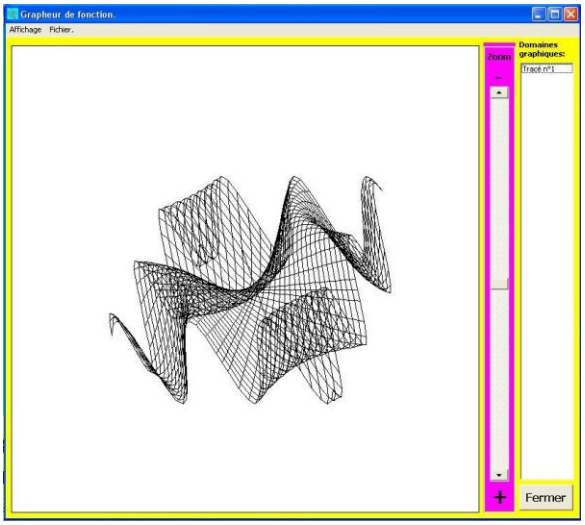
5. Tracé d'une fonction paramétrique avec la primitive PLOT . Le diamètre des points indique leur proximité. A chaque tracé numéroté correspond un point.



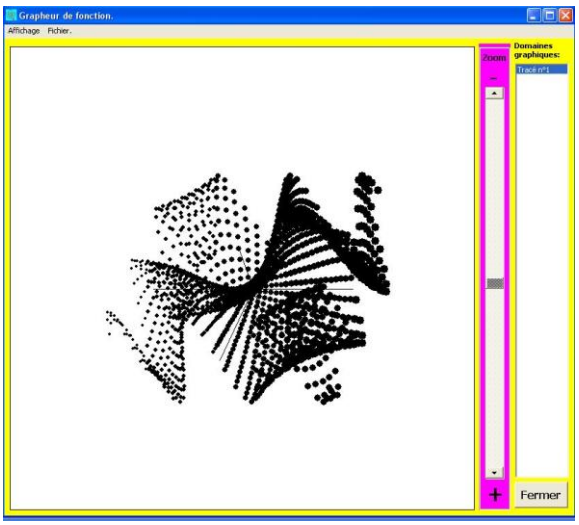
6. Tracé d'une autre fonction paramétrique avec la primitive SEGMENT3D. A chaque tracé numéroté correspond un segment.



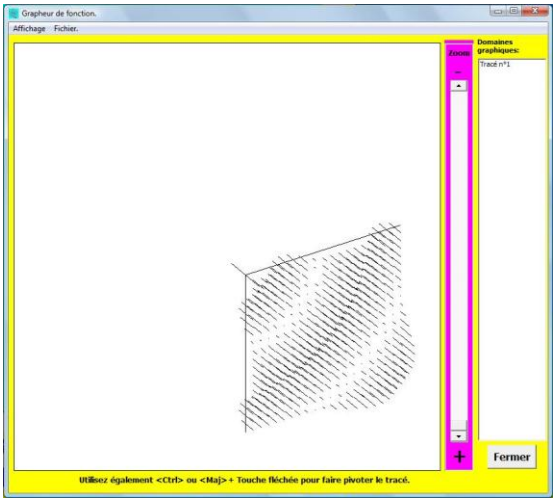
7. Les différents modes de tracé ainsi que la couleur des tracés sont modifiables.



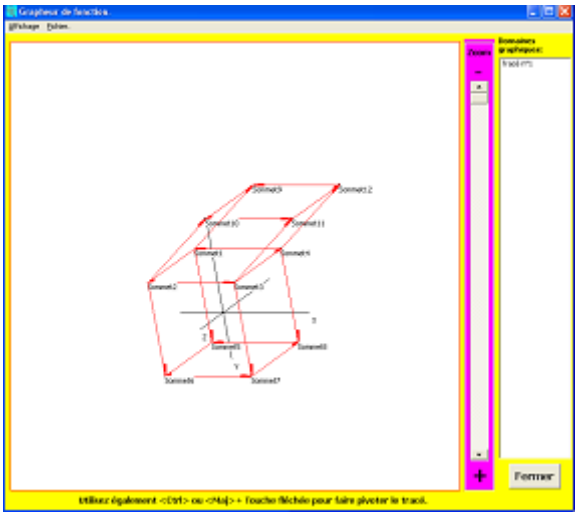
8. Maillage d'une fonction à deux variables avec la primitive SURFACE. A chaque tracé numéroté correspond une surface entière.



9. Tracé discret de la même fonction.



10. Tracé vectoriel de la même fonction (à petite échelle, Les têtes des vecteurs ne sont pas observables).



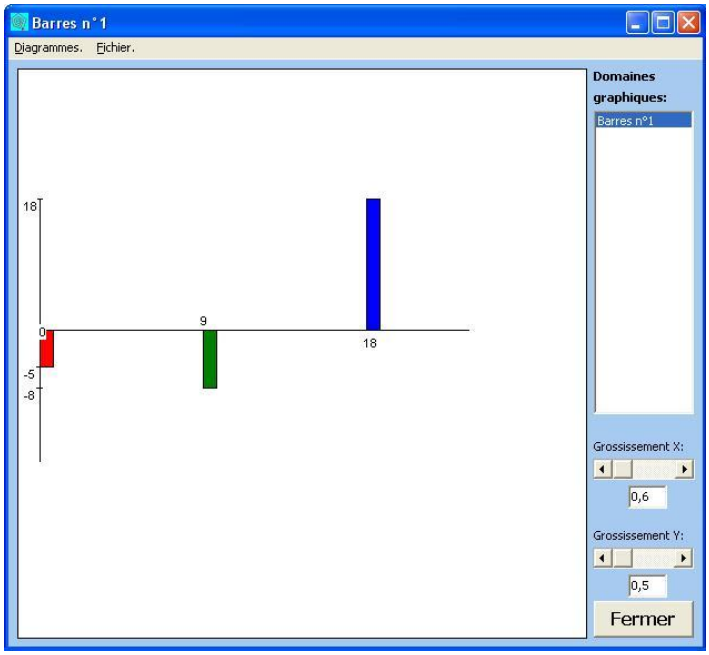
11. Tracé d'un graphe (ici un 4-hypercube) orienté avec le nom des sommets.

Primitive :	Action :
CHAMPVECT	Réalise le tracé d'un champ de vecteurs, en coordonnées rectangulaires.
INITGRAPH	Initialise la liste des coordonnées des points de chaque tracé.
GRAPH2D	Réalise le tracé des fonctions à une variable. (*)
GRAPH3D	Réalise le tracé des fonctions paramétriques. (*)
GRAPHEUR	Autorise ou non l'ouverture de la fenêtre du grapheur de fonctions.
PLOT	Réalise le tracé d'un point en coordonnées rectangulaires dans la fenêtre du grapheur de fonctions.
PLOTPOL	Réalise le tracé d'un point en coordonnées polaires dans la fenêtre du grapheur de fonctions.
SEGMENT3D	Réalise le tracé d'un segment dans la fenêtre du grapheur de fonctions. (*)
SURFACE	Réalise le tracé d'une fonction à deux variables. (*)
SYSPLOT	Réalise le tracé d'une liste de points en coordonnées rectangulaires dans la fenêtre du grapheur de fonctions en associant une couleur à chacun d'eux.
SYSPLOTPOL	Réalise le tracé d'une liste de points en coordonnées polaires dans la fenêtre du grapheur de fonctions en associant une couleur à chacun d'eux.
GRAPHE	Réalise le tracé d'un graphe dans la fenêtre du grapheur de fonctions. (*)

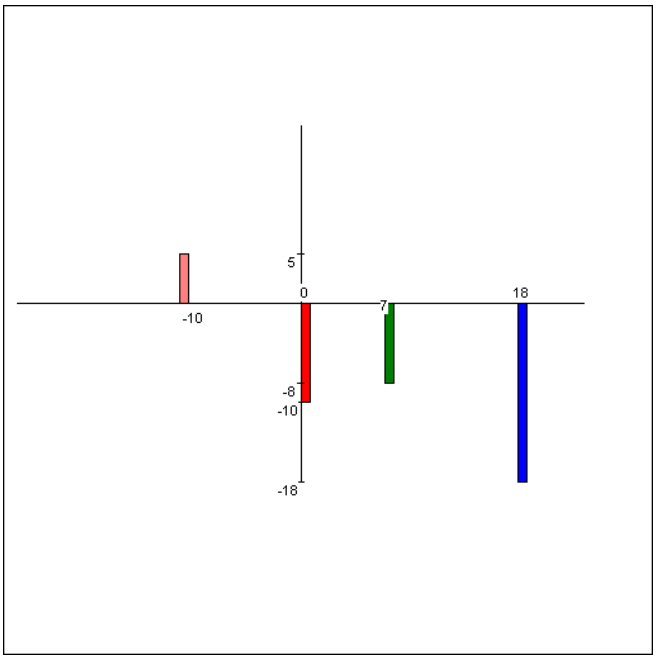
(*) : ces primitives peuvent éventuellement calculer la longueur de la courbe (graphe) ou l'aire de la surface tracée lorsqu'elles suivent une autre consigne qui leur indique quoi faire du résultat.

I.26. La fenêtre-diagrammes.

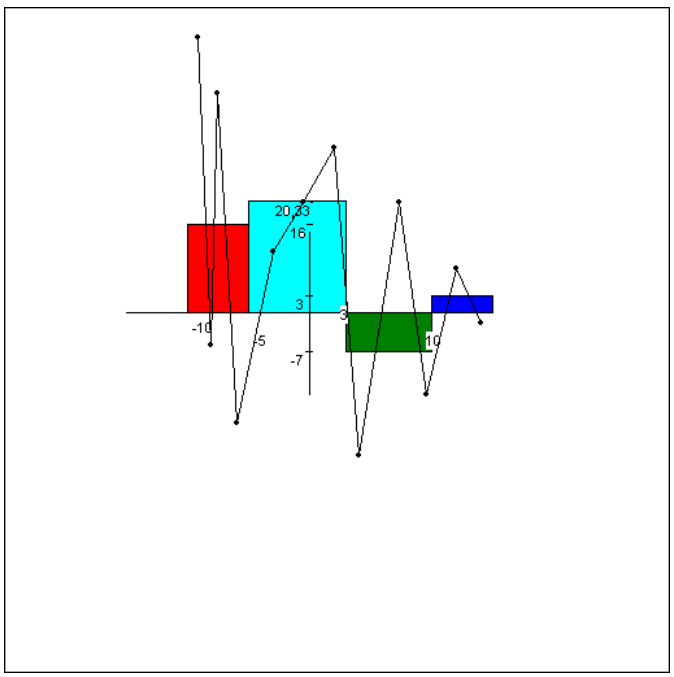
Cette fenêtre est invoquée par l'une ou l'autre des primitives **BARRES** ou **HISTO**. Chaque diagramme calculé est désigné et numéroté par catégorie (barres ou histogramme) pour être listé en tant que domaine graphique. Deux curseurs permettent d'ajuster le grossissement du diagramme visualisé selon les deux axes de représentation graphique. Le menu de la fenêtre donne la possibilité de supprimer certains aspects d'un histogramme (affichage et connexion des points, affichage du repère et des graduations…) ou d'effacer un diagramme de la liste des domaines graphiques. Un enregistrement de l'image du diagramme visualisé sous différents formats (.bmp, .jpg, .gif) est possible en vue de transformations éventuelles dans l'éditeur graphique de votre choix.



1. La fenêtre-diagrammes.



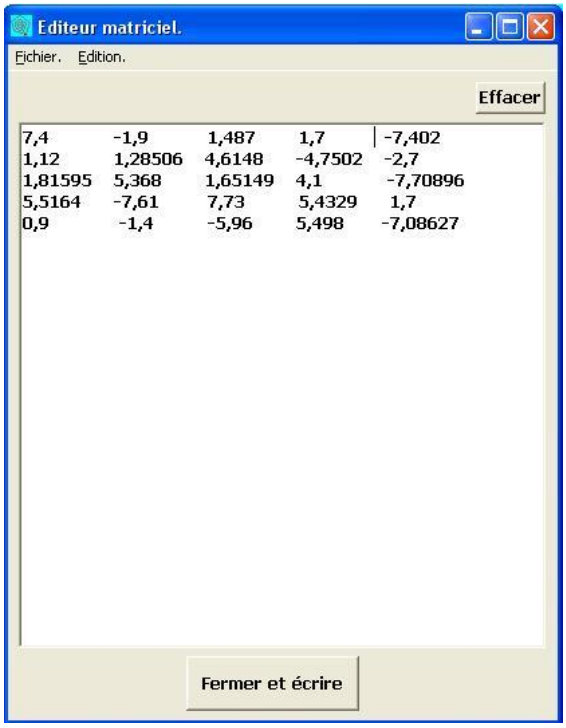
2. Un exemple de diagramme-bâtons.



3. Un exemple d'histogramme.

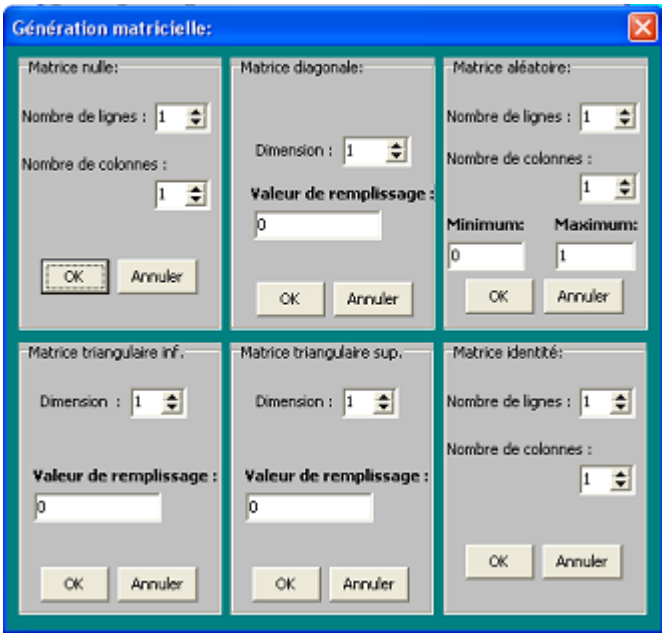
I.27. La fenêtre Editeur matriciel.

Cette fenêtre est invocable grâce à la primitive LISMAT (rice) mais elle peut également apparaître en sélectionnant l'option "Editeur matriciel" du menu "Edition" sur la fenêtre Editeur. Pour saisir une matrice, il suffit simplement de taper ses termes en les séparant par un ou plusieurs espaces. Attention cependant, car si les termes introduits dans l'éditeur matriciel sont vérifiés en cours d'exécution d'un programme pour savoir s'ils correspondent bien à des données numériques ainsi qu' à une matrice valide, ce n'est pas le cas par contre pendant l'écriture d'un programme LOGO, car cette vérification ne peut pas se faire à ce niveau en raison du manque d'information sur les données calculées ou introduites pendant son exécution, à ce moment de votre travail. La rubrique "Fichier" du menu permet d'enregistrer ou de rappeler une matrice dans ou depuis un répertoire de votre disque dur (.matrix).



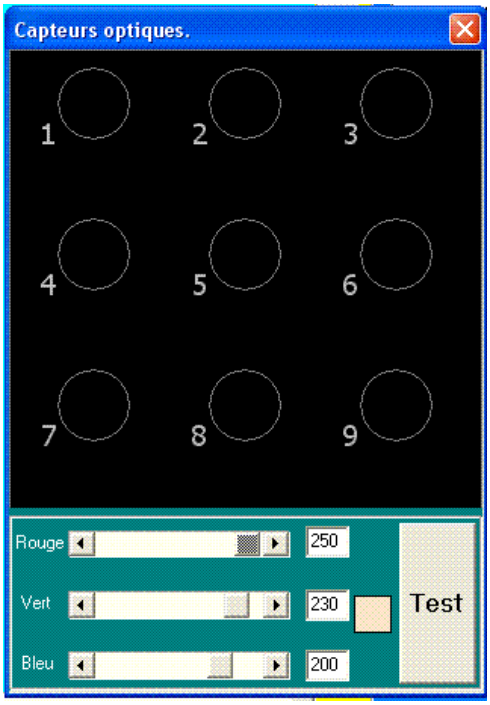
1. Un exemple de matrice 5x5. La taille de la partie décimale des termes de la matrice est limitée à 5 chiffres.

Une option du menu de la fenêtre de l'éditeur matriciel permet d'ouvrir ensuite la fenêtre ci-contre. Un clic sur le bouton OK correspondant au type de matrice et aux paramètres indiqués écrira la matrice (ou le vecteur) dans l'éditeur matriciel.



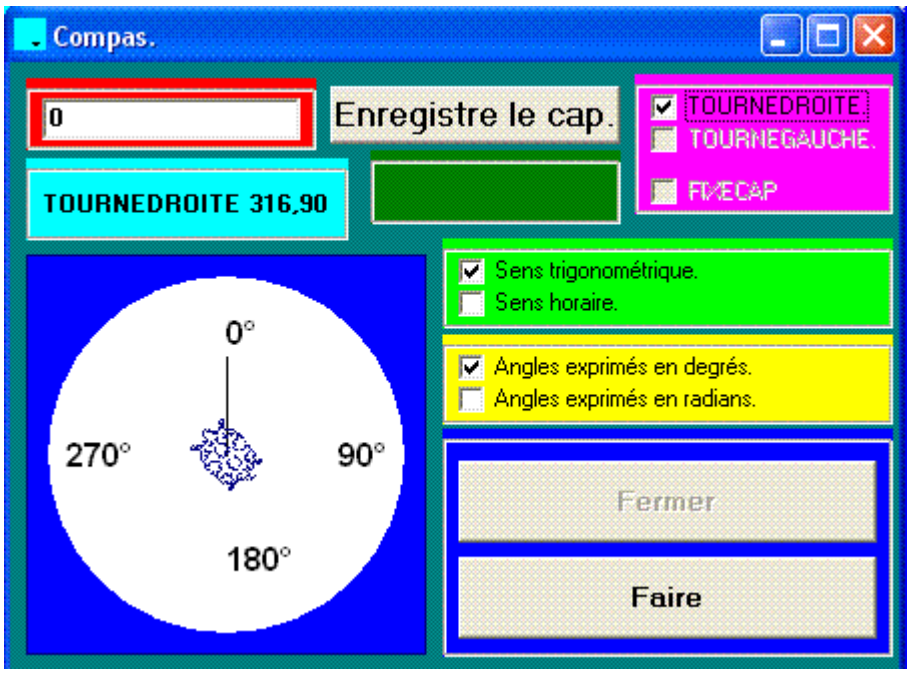
2. Cette fenêtre se découpe en 6 zones, selon le type de matrice souhaité.

I.28. La fenêtre des spots.



A l'origine créée pour émettre des signaux lumineux par l'écran et à destination de capteurs à photo-transistors à construire soi-même dans un usage de robotique, l'existence de cette fenêtre a été maintenue après l'implémentation des primitives de gestion des ports RSC232 et USB . La couleur des signaux est ajustable au moyen de trois curseurs. Elle reste cependant une originalité de Logoplus qui est gérée grâce à trois primitives dédiées : CACHESPOTS, MONTRESPOTS et SPOTS.

I.29. La fenêtre compas.



Invoquée par la primitive COMPAS, cette fenêtre permet de modifier la direction prise par la tortue active (visible ou pas) grâce au cadran mobile réactif aux déplacements de la souris ainsi qu'aux boîtes à cocher qui indiquent soit qu'il s'agit d'une rotation soit d'une direction (cap) imposée, le sens de rotation et l'unité d'angle choisie.

Retour à la table des rubriques >>

II. Les types de données.

Les variables ou données contiennent soit des nombres, des listes, des mots, des couleurs ou des valeurs booléennes (VRAI/ FAUX). En utilisant la primitive DONNE ou son équivalent SOIT, une variable reçoit son contenu. Elle le conservera jusqu'à ce que le programme se termine ou qu'on assigne quelque chose d'autre à cette variable. Vous pouvez utiliser les variables, une fois affectées, tout comme s'il s'agissait de leur contenu. En LOGO, une variable désignée peut contenir successivement par exemple, une liste, un nombre, un mot… au cours du déroulement d'un programme : le trans-typage est ici implicite. Cela vient du fait que le langage LOGO a été conçu pour avoir une syntaxe peu contraignante. Il faut cependant prévenir le jeune programmeur que cette facilité d'écriture recèle un piège redoutable: le risque de fausser des calculs intermédiaires et d'aboutir au final un ou plusieurs résultats erronés. Mieux vaut donc éviter, surtout lorsqu'on débute en programmation, de faire changer trop souvent une variable de type. C'est à la fois la force et la grande faiblesse du langage LOGO.

II.1. Les nombres.

Les nombres sont constamment utilisés dans la vie de tous les jours. Nous avons les nombres appelés entiers naturels : 0, 1, 2, 3, 4, 5, etc. Les nombres négatifs : -1, -2, -3, etc. Enfin les nombres décimaux, ou nombres à virgule. Parfois, comme avec Logoplus, nous les utilisons avec des puissances de 10.
Par exemple : 0,1 ; 3,14 ; 33,3333 ; -5,05 ; -1,0 ; 1E-01

La façon dont ils sont utilisés dans Logoplus, qui calcule aussi bien sur les nombres entiers que sur les nombres décimaux, n'est pas très différente de la façon dont on les utilise dans le langage parlé, ou avec les mathématiques.

n.b. :
- Logoplus considère initialement la virgule comme séparateur décimal, mais l'usage du point est toléré. Dans ce cas, lorsque le point est détecté comme séparateur décimal à un endroit du programme ou d'une consigne, l'affichage des données numériques se fera en utilisant le point comme séparateur décimal. Cependant, si une consigne qui contient des nombres à virgule se réfère à une procédure de l'éditeur qui, elle, utilise le point dans l'expression d'un nombre, l'affichage ultérieur de résultats numériques se fera aussi en utilisant également le point comme séparateur décimal. D'une manière générale, c'est le programme contenu dans l'éditeur qui fixe prioritairement le choix du séparateur décimal selon que dans sa rédaction, il se réfère à l'un des deux ou aux deux indifféremment.
- Les nombres peuvent être utilisés dans des opérateurs mathématiques et dans des opérateurs de comparaison. Ils peuvent également être placés dans des variables.
- Le langage LOGO n'a pas d'autre séparateur que l'espace pour délimiter les consignes. Ne pas hésiter à utiliser les parenthèses dans les expressions calculables pour lever des ambiguïtés dans leur interprétation. Exemple : FIXEXY 50 (-30)
- Une option de calcul sur les nombres longs (qui autorise l'affichage intégral de nombres jusqu'à 4051 chiffres) sur les quatre opérations de base, se trouve dans le menu flottant de la fenêtre de travail. La limite de la notation en virgule flottante qui ne tient compte que les mantisses à 19 chiffres significatifs au plus n'existe plus alors pour ces quatre opérations. Les autres routines mathématiques restent cependant assujetties à la notation en virgule flottante et soumises à la limite des 19 chiffres significatifs de leur mantisse. Le recours à l'une d'entre-elles peut cependant ramener la taille des nombres dans cette limite pour les calculs restants.

Retour au début de la rubrique des types de données >>
Retour à l'index des types de données >>

II.1.a. Notation décimale et hexadécimale.

Si la notation décimale requiert les chiffres de 0 à 9 pour exprimer un nombre, une autre base numérique est souvent utilisée à cet effet, notamment dans la programmation en assembleur et en robotique. C'est la base

16 (ou hexadécimale) et requiert en plus des 10 premiers nombres, ceux de 10 à 15. Pour simplifier leur écriture, ces six nombres supplémentaires sont représentés par les lettres A=10 ; B=11 ; C=12 ; D=13 ; E=14 ; F=15. Ainsi 95 en base 10 s'écrit 5F en base 16 (ou hexadécimale), car (5x16)+15=95. Pour indiquer à LOGO qu'il s'agit d'un nombre exprimé en base 16, on fera précéder son écriture du prédicat $.
Les deux écritures LOGO suivantes sont donc équivalentes :

DONNE "Monnombre 95 /* eq */ **SOIT** "Monnombre 95	**DONNE** "Monnombre $5F /* eq */ **SOIT** "Monnombre $5F

On verra dans les affectations la signification de DONNE .

II.2. Les listes.

Les listes sont des ensembles ou collections de choses délimitées par des crochets ouvrants et fermants. Par exemple :

DONNE "Maliste **[a b c d e f]** est la liste des 5 premières lettres de l'alphabet.

n.b : Il est possible d'utiliser des noms de données à l'intérieur d'une liste, pour rappeler leur contenu (ou valeur) ensuite. Dans ce cas, il est nécessaire de ne pas séparer par un espace le prédicat du nom de cette donnée (:) du reste de ce nom, ce qui aurait à la fois pour conséquence de créer deux items et de rendre le nom de la donnée illisible.

II.2.a. Les listes "inertes".

Les listes dites "inertes" sont des listes qui ne sont pas destinées à contenir des calculs à effectuer, même si elles contiennent des primitives, au contraire des listes "actives" dont le contenu est destiné à être effectué lors du déroulement du programme en tant que partie inhérente de vos algorithmes.

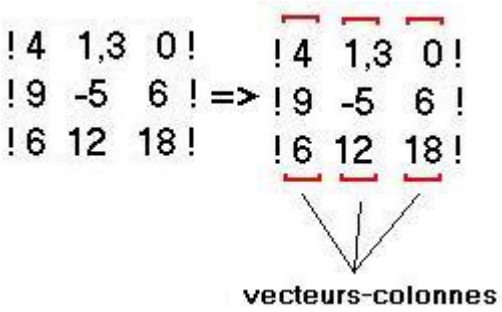
Exemple de liste active :	Exemple de liste inerte :
DONNE "compteur 0 **REPETE 10 [AVANCE :compteur INC :compteur]**	**ECRIS [POUR AV :compteur INC :compteur]** /* Le contenu de la liste est ici considéré comme des mots et non comme des consignes. */

II.2.b. Les listes vectorielles et matricielles.

Ces listes ne doivent contenir que des nombres. Si les listes vectorielles se caractérisent par une succession de nombres, les listes matricielles incluent une ou plusieurs sous-listes qui contiennent un même nombre d'items. Ces sous-listes constituent les vecteurs-colonnes de la matrice.

Exemple de liste vectorielle :	Exemple de liste matricielle :
SOIT "nombre 5,3 **SOIT "vecteur [0 :nombre (-8) 2]**	**SOIT "nbre 6** **SOIT "vect [1,3 (-5) 12]** **SOIT "matrice [[-4 9 (-6)] :vect [0 :nbre 18]]**

Dans ce dernier exemple, "matrice est une liste qui représente une matrice 3 x 3. Ainsi, la liste matricielle **[[4 9 6] [1,3 (-5) 12] [0 6 18]]** sera affichée :



n.b. : De même que pour les nombres, l'utilisation des parenthèses est recommandé pour séparer des items qui représentent des expressions calculables, surtout en cas de valeurs négatives.

Exemple : SEGMENT **[50 (-30)] [-80 75]**

On peut également éviter une erreur de calcul en mémorisant une valeur négative dans une variable.
Exemple : **DONNE** "donnée -8

ECL ITEM [5 :donnée] 2 // donnera -8

Mais attention :

ECL ITEM [5 : donnée] 2 // donnera un autre résultat

II.2.c. Les listes calculables.

Ces listes contiennent une ou plusieurs des primitives ou signes opérandes suivants :

SAUFDERNIERELETTRE SDL SAUFPREMIERELETTRE SPL DERNIERELETTRE DERL PREMIERELETTRE PREML SAUFDERNIER SD SAUFPREMIER SP MODULATION? FIXECANAL_L DETERMINANT DIFFERENCE DIFF ECART_TYPE INTEGRALE SEGMENT3D PUISSANCE VELOCITE? LISNOMBRE VARIANCE QUOTIENT QUOT CARDINAL CARD CAPLUTIN CPL ARRONDIR M_NORME DERNIER DER LUTINY? LUTINX? LISPORT PREMIER PREM SURFACE PRODUIT PROD ARGSECH SOURISX ARGSINH SECANTE COSINUS COS ARGTANH SOURISY ARGCOSH RACINE2 RAC2 RACINE3 RAC3 M_TRACE ARGCSCH ARGCOTH RANDDEC GRAPH2D GRAPH3D HASARD CANAL? ARCCOT BOUCLE ENTIER RADIAN ARCCSC ARCCOS XCOORD RANDOM COMPAS ARCTAN YCOORD ZEROS? NOMBRE ARCSIN SAISIS ARCSEC COSEC DEGRE COTAN ASCII SOMME IMAGE ZMAX? RESTE TOPS? CHOIX SINUS SIN COTH COSH ITEM CSCH MIN? MAX? TANH SINH VCOS SECH VSIN REEL RVB TAN CAP VA LOG ABS EXP PI LN + - * / ^ %

Elles sont susceptibles de contenir un ou plusieurs calculs que l'analyseur de texte décomposera en suite d'instructions.

II.3. Les mots.

Ce sont en fait des mots, composés souvent par des lettres, mais qui peuvent inclure également des chiffres.
Par exemple : **DONNE** "Monmot "a1b2c3d4

Remarque : Si "a1b2c3d4 est la syntaxe correcte pour exprimer un mot en LOGO, "Monmot sera reconnu ensuite par LOGO comme un nom de donnée (chose ou variable). Il devra être alors précédé de ':' .

Retour au début de la rubrique des types de données >>

Retour à l'index des types de données >>

II.4. Les booléens.

Deux valeurs sont possibles : VRAI et FAUX (il n'y a pas de guillemets devant car ce sont des valeurs à part entière). Ces valeurs peuvent être implicitement générées par LOGO par l'utilisation des signes de comparaison et d'inégalités (<, <=, =, >=, >) lors de tests.

II.5. Les couleurs.

Avec Logoplus, les couleurs peuvent être désignées nominalement et sont au nombre de 17 :

BLANCHE BLEUE JAUNE NOIRE ROUGE VERTE TILLEUL EAU FUCHSIA SARCELLE GRISE OLIVE MARRON ARGENT ORANGE ROSE VIOLETTE
On peut les regrouper sous forme d'une liste pour illustrer le paragraphe II.2. sur les listes :
DONNE "Mescouleurs [BLANCHE BLEUE JAUNE NOIRE ROUGE VERTE TILLEUL EAU FUCHSIA SARCELLE GRISE OLIVE MARRON ARGENT ORANGE ROSE VIOLETTE]

Chacune de ces couleurs correspond à un nombre :

```
blanche <-> 16777215
  bleue <-> 16711680
  jaune <-> 65535
  noire <-> 0
  rouge <-> 255
  verte <-> 32768
 tilleul <-> 65280
    eau <-> 16776960
fuchsia <-> 16711935
sarcelle <-> 8421376
  grise <-> 8421504
  olive <-> 32896
 marron <-> 128
 argent <-> 12632256
 orange <-> 8421631
   rose <-> 16744703
violette <-> 8388736
```

Inversement, chaque nombre correspond à une couleur physique, qui n'est pas forcément nominale.
Exemple : COULEURTORTUE 2 fera prendre à la tortue une couleur proche du noir, mais 2 ne correspond pas à une couleur nominale.

n.b. : La primitive RVB permet également de définir une couleur en mélangeant en différentes proportions les trois couleurs fondamentales :

```
RVB [ 0 0 0 ] // noir
RVB [ 0 0 255 ] // bleu
RVB [ 128 64 0 ] // marron
RVB [ 0 255 255 ] // bleu_cyan
RVB [ 0 128 128 ] // vert1
RVB [ 0 255 0 ] // vert2
RVB [ 255 0 255 ] // magenta
RVB [ 128 128 0 ] // olive
RVB [ 255 128 0 ] // orange
RVB [ 128 0 128 ] // pourpre
```

```
RVB [ 255 0 0 ] // rouge
RVB [ 255 255 0 ] // jaune
```

Question : la liste "Mescouleurs est-elle une liste active ou inerte ? Les listes qui suivent RVB ?

Retour au début de la rubrique des types de données »
Retour à l'index des types de données »

II.6. Les lutins.

Les lutins sont des formes graphiques déplaçables désignés en LOGO par des mots. Ils suivent toujours une primitive dédiée à la gestion des lutins. Ils sont limités au nombre de 16 en Logoplus et chacun d'entre eux ne peut porter qu'une seule couleur. Exemple :

```
MONTRELUTIN "voilier /* ou */ MONTRELUTIN ["flèche "multiplie "voilier ]
REVEILLE ["flèche "multiplie "voilier ] /* ou */ REVEILLE "flèche
```

Contrairement aux tortues qui n'ont que trois formes possibles, les lutins peuvent avoir des formes multiples définies dans la grille de la fenêtre d'édition des lutins, ainsi qu'une inertie dans leur déplacement, ce qui les rend particulièrement intéressants dans la simulation d'effets mécaniques et physiques.

III. La syntaxe.

III.1. Les primitives.

333 primitives, sans compter leurs abréviations et les caractères spéciaux (voir catalogue).

Une convention d'écriture : séparer chaque primitive par un espace de chaque côté (sauf en début de ligne dans l'éditeur.)

Une recommandation : Ne pas hésiter à structurer les calculs à l'aide de parenthèses.

III.2. Les affectations.

Une affectation est une opération par laquelle une variable nommée reçoit un contenu (nombre, mot, liste, couleur, booléen…). Cette opération se réalise au moyen de deux primitives équivalentes qui sont : DONNE ou SOIT.

Exemple :

```
DONNE "1234 5
SOIT "mot "logo
```

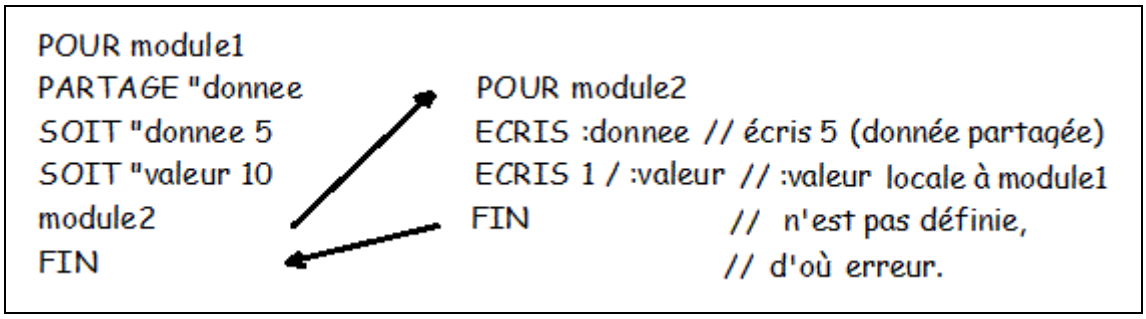
Le recours à l'une de ces deux primitives en dehors d'un bloc de définition procédurale POUR … FIN rend global l'usage d'une variable, qui devient ainsi partagée.

III.3. Les données partagées.

La primitive PARTAGE permet de transmettre la valeur d'une donnée définie dans un bloc à un bloc appelé dans la suite d'un programme LOGO et de la rendre ainsi globale.

```
PARTAGE [ "donnée1 "donnée2 ]
/* ou */ PARTAGE "donnée1 PARTAGE "donnée2
```

Cette primitive, pour bien fonctionner dans l'ensemble de votre programme, doit être placée dans un bloc "appelant", c'est à dire un bloc qui invoque tous les autres blocs qui utilisent vos variables partagées dans la suite de votre programme. Ainsi :



n.b : Une variable déclarée dans l'éditeur de programmation mais à l'extérieur de toute procédure est d'emblée considérée comme **globale**, donc partagée.

III.4. Les structures de contrôle.

III.4.a. TESTE/SIVRAI/SIFAUX (SINON)

Syntaxes :

TESTE condition SIVRAI […] // action 1. SIFAUX […] // action 2.	TESTE condition SIFAUX […] // action 1. SIVRAI […] // action 2.

Le test porte sur la valeur booléenne condition . Selon cette valeur, l'une ou l'autre action sera effectuée. TESTE … SIVRAI/SIFAUX est similaire à SI … SINON des autres langages de programmation. Dans les dernières versions de Logoplus, la primitive SINON peut tenir lieu et place à SIFAUX du standard LOGO. L'une des lignes SIVRAI ou SIFAUX peut être omise.
Les crochets peuvent être également omis s'il n'y a qu'une seule primitive à être effectuée pour chaque suite au test.
n.b : L'usage des parenthèses est encore ici fortement recommandé. Par exemple :

```
TESTE (( ITEM [ 10 20 (-30) 40 50 ]  5 ) >= 30 )
```

Dans cette expression, Les parenthèses ont permis :

1) d'éviter une confusion 20 –30= -10 dans les items de la liste.
2) d'être sûr de comparer les arguments souhaités, ici le cinquième item de la liste (50) à 30.

Retour au début de la rubrique des syntaxes »
Retour à la table des rubriques »

III.4.b. SI

Syntaxe : SI condition […]
Le code entre crochets qui suit la primitive SI ne sera exécuté que seulement si la condition est VRAI (e) . Les crochets peuvent être omis s'il n'y a qu'une seule primitive à être effectuée.

III.4.c. REPETE / BOUCLE

Syntaxe : REPETE nfois […]
REPETE est similaire à TANTQUE (ou TANT QUE). La différence est que REPETE effectue en boucles la consigne ou les consignes entre crochets autant de fois que le nombre ou la donnée nfois indiqué.
La primitive BOUCLE rappelle le nombre de fois que les consignes entre crochets a été effectuées.

Les crochets peuvent être omis s'il n'y a qu'une seule primitive à être effectuée.

III.4.d. TANTQUE (TANT QUE)

Syntaxe : TANTQUE condition […] (ou TANT QUE condition […])
TANTQUE ressemble beaucoup à SI. La différence est que TANTQUE répète le code entre accolades jusqu'à ce que la condition devienne fausse (FAUX).
Les crochets peuvent être omis s'il n'y a qu'une seule primitive à être effectuée.

III.4.e. Les opérateurs booléens.

Tandis que les opérateurs mathématiques travaillent avec des nombres,
les opérateurs booléens travaillent avec des valeurs booléennes (VRAI , FAUX).
Il y a seulement trois opérateurs booléens : ET, OU , NON. Les deux premiers opérateurs ont un autre équivalent sous Logoplus :

ET : LESDEUX ?
OU : UNDESDEUX ?

Retour au début de la rubrique des syntaxes >>
Retour à la table des rubriques >>

III.5. L'écriture des résultats.

L'écriture des résultats peut se faire dans l'écran de la tortue ou dans l'afficheur de texte. On utilisera la primitive DESSINETEXTE dans l'écran de la tortue et les primitives ECRIS (EC) ECRISLIGNE (ECL) dans l'afficheur de textes.

III.6. Les blocs procédure et fonction.

Un bloc en LOGO est un ensemble nommé de consignes compris entre les primitives POUR et FIN. Un nom est donc affecté à ce bloc et invoqué à chaque fois que l'on souhaite faire effectuer les consignes qu'il contient.
Exemple :

```
POUR néant
FIN
```

Peu de chose à dire sur un bloc qui ne contient aucune consigne effectuable en LOGO. Taper néant dans la ligne de commande ne fera rien apparaître d'autre que "Je travaille avec néant" et "J'ai terminé" dans l'afficheur de textes et c'est tout. Nous pouvons à présent insérer une consigne à l'intérieur du bloc, après le nom, ce qui transformera ce bloc en procédure.
Par exemple :

```
POUR procédure
ECRISLIGNE [ Voici une procédure. ]
FIN
```

Après avoir tapé procédure dans la ligne de commande, l'afficheur de textes contiendra la ligne où sera écrit la phrase : Voici une procédure.
C'est l'utilité première de toute procédure informatique : effectuer une ou plusieurs actions, car on aurait pu ajouter d'autres consignes encore.

Il est parfois nécessaire de demander à un bloc de retourner un résultat et un seul, un mot, une liste, un nombre, un booléen. La procédure devient alors une fonction. Deux primitives LOGO sont indispensables pour cela : REPONDRE ou son équivalent RENDS .
Par exemple :

```
POUR fonction
REPONDRE [ Voici une fonction. ]
FIN
```

Cette fonction a pour travail unique (mais ce n'est pas toujours le cas) de restituer la liste :
[Voici une fonction.]

sans plus car LOGO peut découper cette liste en mots ou simplement l'afficher ou d'autres choses mais encore faut-il le lui demander.
Ainsi taper fonction dans la ligne de commande ne rendra … aucun résultat visible.
Il est souvent utile de faire précéder l'invocation à une fonction d'une consigne qui explique à LOGO ce qu'il doit faire du résultat, par exemple :

```
ECL fonction // On écrit la liste [Voici une fonction.]
DONNE "maliste fonction // On affecte à maliste le contenu de la liste [Voici une fonction.]
```

En résumé :

Procédure :	Pas d'action , une ou plusieurs actions.
Fonction :	Un résultat.

Retour au début de la rubrique des syntaxes »
Retour à la table des rubriques »

III.7. La récursivité.

Littéralement, le mot récursivité a pour origine "reprendre sa course". La récursivité signifie qu'un bloc de calculs doit s'appeler et recommencer ses calculs, mais avec des valeurs souvent différentes (mais pas obligatoirement). Le récursivité n'a donc de sens que si ce n'est pas l'utilisateur qui invoque un bloc mais le bloc en lui-même. La question qui vient ensuite est : quand cette course doit-elle s'arrêter ? et c'est la question que se posent tous les puristes de la récursivité informatique avec inévitablement une autre question : quel critère d'arrêt convient-il pour un problème envisagé ?

Exemple :

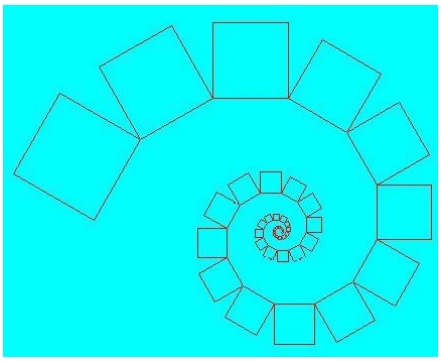
Cas d'une fonction :	Résultat :

`POUR fonction :n`
`TESTE (( ABS :n ) = 0)  // C'est le critère d'arrêt.`
`SIVRAI REPONDRE ABS :n`
`SIFAUX REPONDRE fonction ( ABS ( :n)  -1 ) // Appel !!`
`FIN`
 	
`EFFTXT ECRISLIGNE fonction 5`
`/* Il faut invoquer au moins une fois le bloc`
`  récursif.*/`
	5
	4
	3
	2
	1
	0
	0
	
	LOGO: - J'ai terminé.

Fort bien, mais que se passera t-il alors si on ne détermine pas de critère d'arrêt, lorsqu'on a affaire, par exemple, à une procédure ?
Là, les choses sont moins évidentes :

1) La procédure récursive contient la primitive RETOURNE.
   Cela signifie que la récursivité prendra fin au moment où la primitive RETOURNE sera rencontrée, car les calculs reviendront peu à peu à leur situation antérieure, c'est-à-dire avant l'appel récursif de la procédure.

2) La procédure récursive ne contient pas la primitive RETOURNE.
   a) Si elle ne mémorise pas de résultats au cours de son travail, la récursivité est alors infinie (Logoplus est l'une des versions LOGO qui excellent dans ce domaine, d'où la présence indispensable de son bouton STOP).
   b) Dans le cas où des calculs sont mémorisés (empilés), Logoplus interrompra le programme lorsqu'il n'y aura plus assez de place pour effectuer les calculs en affichant un message explicatif.

Cas d'une procédure :	Résultat :
POUR carré :taille TESTE :taille < 1 // C'est le critère d'arrêt. SIVRAI RETOURNE SIFAUX [REPETE 4 [AVANCE :taille TOURNEGAUCHE 90] /* dessine un carré */ AVANCE :taille TOURNEDROITE 30 carré :taille *0,9 /* recommence les calculs avec une plus petite taille. */] FIN	

(*) Remarquer la primitive RETOURNE qui a été soulignée après le critère d'arrêt. Elle indique que tôt ou tard, la procédure devra rendre **ses** comptes ! (sans jeu de mots)

En résumé :

Une fonction munie d'un critère d'arrêt.	Récursivité finie
Une procédure sans RETOURNE ou une fonction sans critère d'arrêt.	Récursivité infinie ou interruption du programme.

Retour au début de la rubrique des syntaxes >>
Retour à la table des rubriques >>

III.8. Les commentaires.

Un programme LOGO est constitué de primitives qui sont exécutées lorsque le programme est lancé, mais aussi parfois de commentaires. Les commentaires ne sont pas destinés à être exécutés. Logoplus les ignore simplement lorsque votre programme est exécuté. Ils sont là pour aider d'autres programmeurs à mieux comprendre votre travail mais ils peuvent contenir des liens hypertextes, ce qui les rend précieux pour étayer vos explications.

Il existe deux types de commentaires: les commentaires encadrés et les commentaires de fin de ligne.

Les commentaires encadrés sont balisés par les symboles /* et */.
Les commentaires de fin de ligne sont annoncés par la balise // et se terminent naturellement par une fin de ligne (avec un retour-chariot éventuellement).

Par exemple, le petit programme suivant ne fait rien car la seule primitive exécutée est la primitive STOP qui arrête le programme :

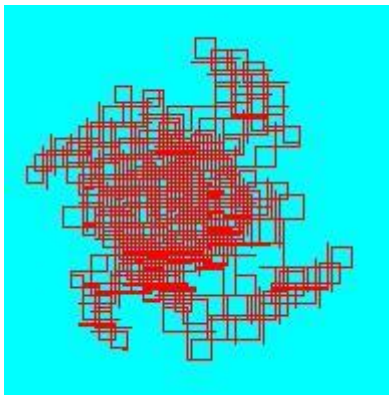
```
/* Ceci est un commentaire encadré. */
 STOP  // ceci est un commentaire de fin de ligne
```

Retour au début de la rubrique des syntaxes »
Retour à la table des rubriques »

IV. Des exemples.

IV.1. TESTE/SIVRAI/SIFAUX (SINON)

```
eff efftxt accélère déroule ct bc
soit "ordre 14 soit "taille 2 trait 1
arbre :ordre :taille

pour arbre :ordre :taille
teste :ordre > 1
sivrai [
          avance :taille tournegauche 90
          dec :ordre
          arbre :ordre :taille
          tournegauche 90 inc [ :taille 2 ]
          avance :taille tournegauche 90
          dec :ordre
          arbre :ordre :taille
          tournegauche 90 avance :taille
        ]
sifaux [ tg 90 av :taille retourne ]
fin
```



IV.2. SI

```
POUR raccourcir : nom
SI :nom = "" RETOURNE
ECRISLIGNE :nom
raccourcir SAUFPREMIERELETTRE :nom
FIN

 efftxt raccourcir "abracadabra
```

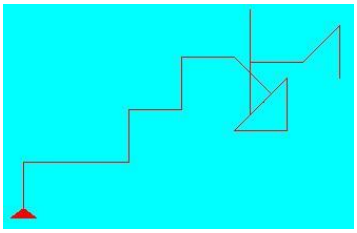
```
abracadabra
bracadabra
racadabra
acadabra
cadabra
adabra
dabra
abra
bra
ra
a
```

Retour à la table des exemples »
Retour à la rubrique des exemples »

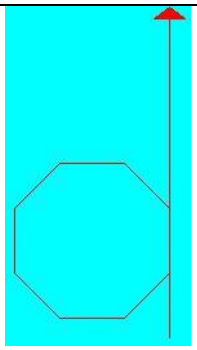
Retour à la table des rubriques »

IV.3. REPETE/BOUCLE

```
EFFACEECRAN ACCELERE
BAISSECRAYON DEROULE
DONNE "caps [ 0 45 90 135 180 225 270 315 ]
REPETE RANDOM 1 8 [
                    AVANCE 50
          FIXECAP ITEM :caps RANDOM 1 8
                    ]
```



```
EFFACEECRAN ACCELERE
BAISSECRAYON DEROULE
DONNE "caps [ 0 45 90 135 180 225 270 315 ]
REPETE RANDOM 1 8 [
                    AVANCE 50
              FIXECAP ITEM :caps BOUCLE
                    ]
```



Retour à la rubrique des exemples »
Retour à la table des exemples »

IV.4. TANTQUE (TANT QUE)

```
POUR énumère :i
EFFACETEXTE
TANT QUE (:i > 0) [
                    ECRISLIGNE :i
                    DONNE "i" :i - 0,5
                  ]
FIN

EFFTXT énumère 10
```

```
10
9,5
9
8,5
8
7,5
7
6,5
6
5,5
5
4,5
4
3,5
3
2,5
2
1,5
1
0,5

LOGO: - J'ai terminé.
```

Retour à la table des exemples »
Retour à la rubrique des exemples »

V. Tableau des premières occurrences de chaque primitive par cycle d'apprentissage.

Ce tableau est donné à titre indicatif au personnel éducateur/enseignant, mais il n'a rien d'impératif. Sa fonction est surtout d'aider chaque utilisateur de Logoplus à mieux se repérer dans le corpus des primitives

et il peut ainsi être modifié ou ajusté selon le niveau du groupe d'enfants/élèves que l'adulte aura à initier au langage LOGO.

Cycle :	**cycle 2 (apprentissages fondamentaux)** pour le langage LOGO : à partir du CE1-CE2	**cycle 3 (consolidation)** CM1-CM2-6ième	**cycle 4 (approfondissements)** cinquième, quatrième, troisième,	**Lycée et plus :**
Première occurrence des primitives essentielles de chaque cycle dans le parcours d'apprentissage	POUR / FIN AVANCE /RECULE BAISSECRAYON / LEVECRAYON CACHETORTUE / MONTRETORTUE COULEURFOND COULEURTORTUE EFFACEECRAN ORIGINE	PI EFFACELIGNE LISCAR LISLISTE LISNOMBRE TANTQUE (TANT QUE) DONNE / SOIT (CM2) DEROULE / ENROULE DATE / TEMPS	LIEU RAMENE SITUATION VERS DIFFERENTS? EGALITE? NEGATIVE? TESTE SI SIVRAI / SIFAUX	ARCCOS ARGCOSH ARCCOT ARGCOTH ARCCSC ARGCSCH ARCSEC ARGSECH ARCSIN ARGSINH

d'un élève :	POINT REPETE / BOUCLE TOURNEDROITE / TOURNEGAUCHE ECRIS ECRISLIGNE EFFACETEXTE Les couleurs : [BLANCHE VERTE JAUNE ROUGE BLEUE NOIRE TILLEUL EAU FUCHSIA SARCELLE GRISE OLIVE MARRON ARGENT ORANGE ROSE VIOLETTE]	6ième : CAP COMPAS FIXECAP FIXEX FIXEXY FIXEY SEGMENT VA XCOORD YCOORD DESSINETEXTE CLIC? SOURISX SOURISY SURECRAN? ET (&) OU (|) NON OPPOSE STOP CARDINAL PREMIER /DERNIER FIXEHASARD HASARD RANDDEC RANDOM ABS ASCII CARACTERE FONDECRAN ACCELERE / RALENTIS ATTENDS TRACE / SILENCE	(SINON) ZERO? OBEIS AUCUNE TOUTES ARRONDIR DEC / INC DIFFERENCE DIV ENTIER COSINUS SINUS TAN MODEDEGRE MODERADIAN DEGRE / RADIAN RACINE2 PRODUIT QUOTIENT MOD (%) / RESTE SOMME PUISSANCE MAX? / MIN? ZEROS? PERMUTATIONS LISTEVARS REPONDRE RETOURNE COPIE PARTAGE PERMUTE VRAI / FAUX INSERE ITEM LISTE / PHRASE	ARCTAN ARGTANH COMPLEXE CONJUG COSH COSEC COTAN COTH CPLXADD CPLXPROD CSCH DIVISEURS EXP GRAPH3D CHAMPVECT FVECT HEXA? IMAGE INTEGRALE LISMAT LN LOG RACINE3 REEL SECANTE SECH SINH TANH TRANSLATE SURFACE POLYSOLVE PLOTPOL RAPPELLE SAUVE SUPPRIME SYSPLOT SYSPLOTPOL Statistiques : BARRES ECART_TYPE HISTO VARIANCE

Cycle :	cycle 2 (apprentissages fondamentaux) pour le langage LOGO : à partir du CE1-CE2	cycle 3 (consolidation) CM1-CM2-6ième	cycle 4 (approfondissements) cinquième, quatrième, troisième,	Lycée et plus :
Première occurrence des primitives essentielles de chaque cycle dans le parcours d'apprentissage d'un élève :			MEMBRE? REMPLACE /SAISIS ROTATION SCINDE SWAP VICEVERSA SAUVEECRAN ARC GRAPH2D GRAPHEUR INITGRAPH PLOT SEGMENT3D ZMAX ZMAX? ZPOINT ZPOINT? ZSEGMENT **ainsi que toutes les primitives de gestion des ports RS232 et USB en vue d'activités robotiques en technologie, gestion d'appareils de mesures simples ...**	**Algèbre linéaire :** DETERMINANT M_ALEA M_DIAG M_DIFF M_IDENT M_INVERSE M_NORME M_NULLE M_SOMME M_SUPPR M_TRACE M_TRANSPOSE M_TRIINF M_TRISUP PRODEXT PRODINT VCOS VDIM VPROD VSIN V_PROPRES V_UNIQUE (**M_** : matrice **V_** : vecteur)

Retour à la table des rubriques >>

VI. Un accès à la robotique.

Avec Logoplus, comme pour quelques autres versions du langage LOGO, il est possible d'entrer dans l'univers créatif de la robotique, à cette différence près qu'avec Logoplus, cet accès n'est pas offert "clef en main" grâce à un robot déjà tout prêt monté muni d'un boîtier de commande soigneusement fermé mais qui ne permet pas d'en apprendre plus pour un novice sur la chaîne techno(logique) mise en jeu. Avec Logoplus, l'utilisateur dispose d'un corpus de primitives agissant sur des cartes de communication qui peuvent à leur tour émettre des signaux soit vers des périphériques (relais, moteurs, autres circuits…) soit au contraire vers l'ordinateur où ces signaux seront analysés et traités par d'autres primitives. Ces cartes sont disponibles chez la société Velleman pour un prix modique et ont l'avantage de pouvoir s'adapter à tous vos projets en électromécanique qui utiliseront le langage LOGO. A retenir qu'il existe également d'autres primitives dans Logoplus qui peuvent accéder directement aux fonctions contenues dans une bibliothèque dynamique (DLL) correspondant à un périphérique de votre choix (imprimante, lecteur DVD, moteurs pas à pas, relais électromagnétiques…), lequel périphérique devient alors possible à piloter en langage LOGO. On voit ainsi que si cet accès à la robotique n'est pas immédiat en Logoplus, il offre ensuite davantage de possibilités dans la conception et la réalisation de projets dans ce domaine.

VI.1. Les cartes de communication Velleman.

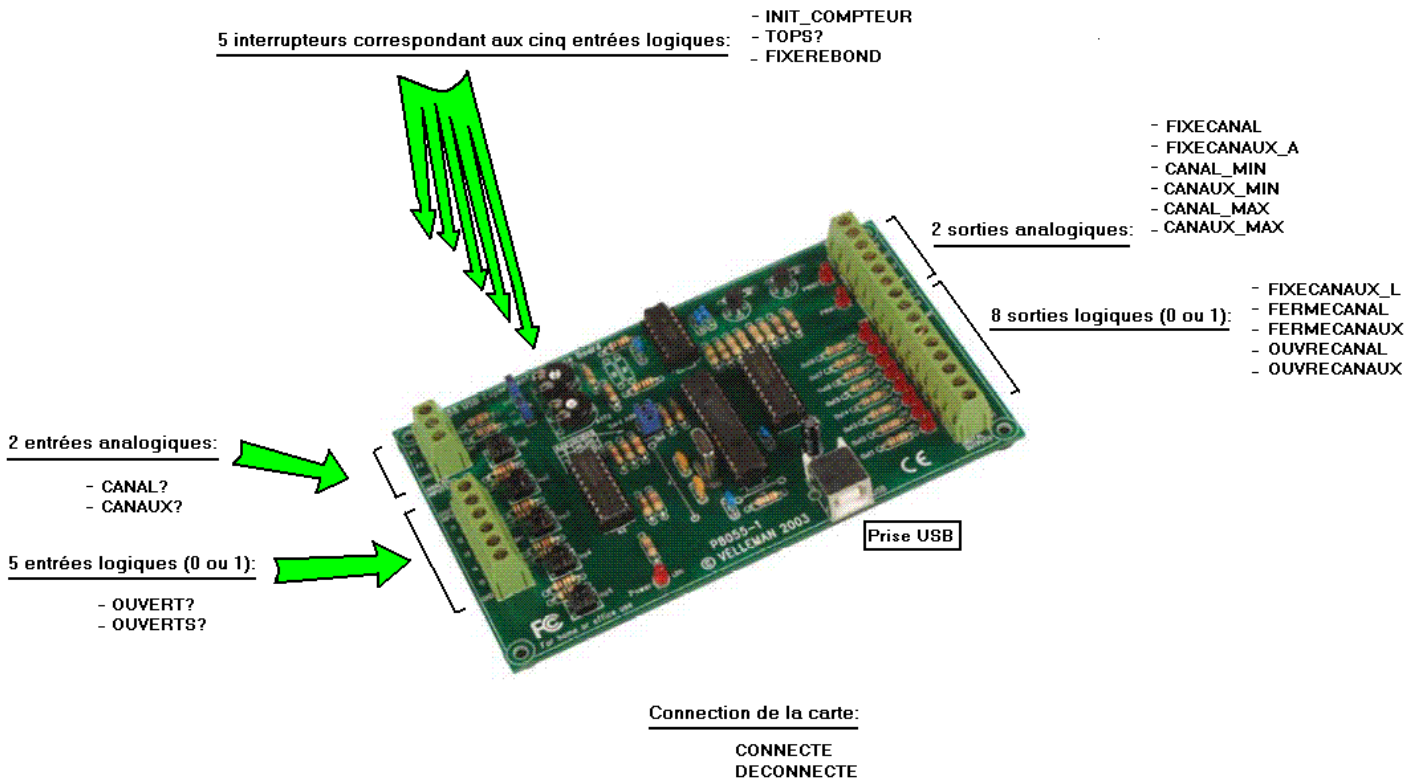


Fig 1. La carte K8055.

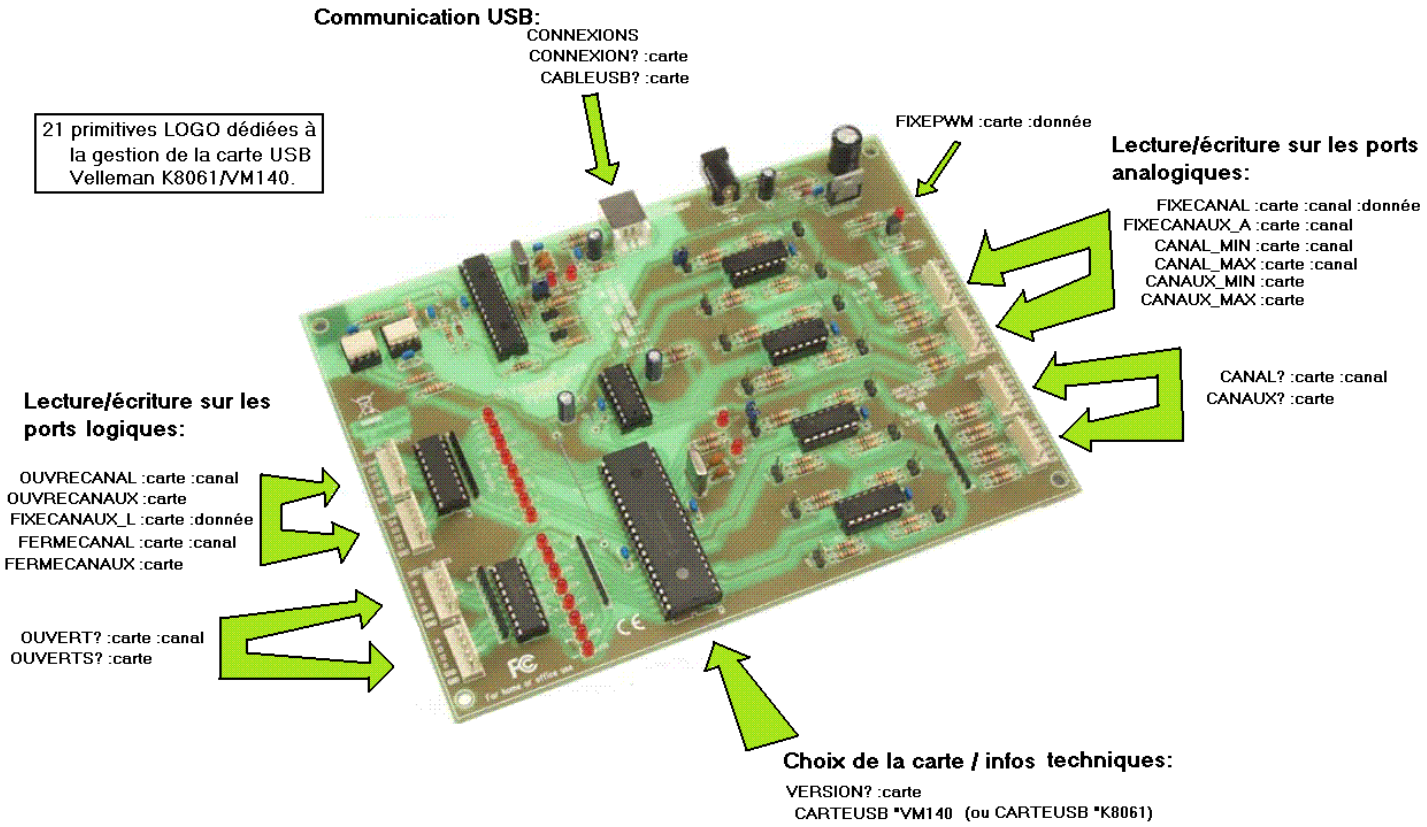
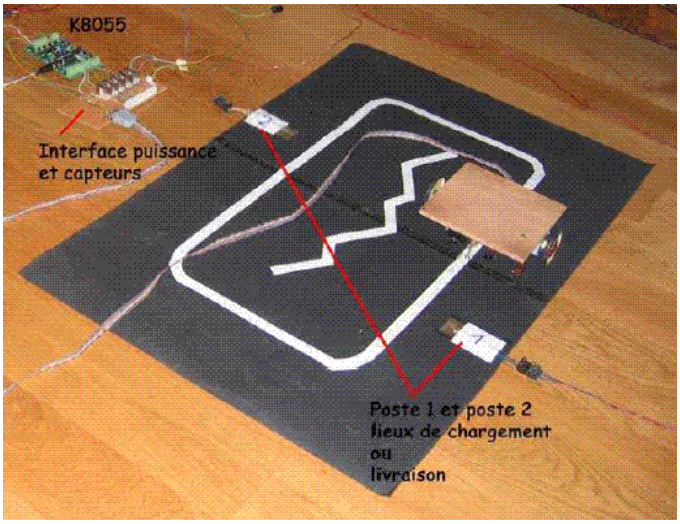


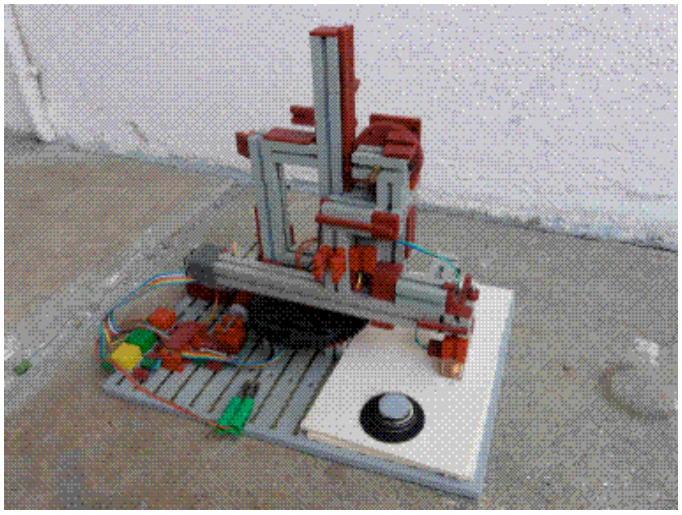
Fig 2. La carte K8061.

Retour à la table des rubriques >>

VI.2. Deux projets réalisés en robotique avec Logoplus.



Le robot-pisteur.


Le robot solutionneur du problème des tours de Hanoï.

Retour à la rubrique Un accès à la robotique.>>

VII. Comparaison des syntaxes entre le langage LOGO et les langages BASIC et Python.

a) Avec le langage BASIC.

Le langage BASIC, bien que fondé sur une syntaxe anglo-saxonne, n'en reste pas moins l'un des premiers langages de programmation les plus faciles à apprendre et universellement connu. Il était donc naturel de construire un tableau de comparaison des syntaxes, qui pourra éventuellement aider le(la) débutant(e) en LOGO à écrire plus facilement ses premiers programmes dans ce langage.

Langage BASIC	Langage LOGO
10 LET ETOILE$ = "" 20 RETURN 30 WAIT 2 40 LET n INT(0.5) 50 LET r SQRT(6) 60 LET longueur LEN "abcdef	DONNE "ETOILE " // rappel : DONNE éq à SOIT RETOURNE ATTENDS 2 DONNE "n ENTIER 0.5 SOIT "r RACINE2 6 DONNE "longueur CARDINAL "abcdef
10 INPUT "Quel est votre nom ?"; NomUtilisateur$ 20 PRINT "Bonjour "; NomUtilisateur$	ECL [Quel est votre nom ?] DONNE "NomUtilisateur LISLISTE ECL PHRASE "Bonjour" :NomUtilisateur //PHRASE éq à PH
10 REM NOMBRE et ETOILE$ sont à définir auparavant. 20 FOR I = 1 TO NOMBRE 30 LET ETOILE$ = ETOILE$ + "*" 40 NEXT I	// :NOMBRE et :ETOILE sont à définir auparavant. REPETE :NOMBRE DONNE "ETOILE MOT :ETOILE "*
10 REM Déclaration des variables.	// Déclaration des variables.

```
20 DIM résu AS STRING
30 DIM tq AS INTEGER
40 REM Initialisation de tq
50 tq = 0
60 REM Efface l'écran
70 CLS
80 REM "TANT QUE" tq est plus petit que 10.
90 WHILE tq < 10
100 PRINT ""; tq
110 REM incrémentation de tq
120 tq = tq + 1
130 REM Recommence la boucle.
140 WEND
150 PRINT "C'est fini."
160 REM termine le programme.
170 END
```

```
DEFINIS "résu  MOT // facultatif en LOGO.
DEFINIS "tq NOMBRE  // facultatif en LOGO.
// Initialisation de tq
DONNE "tq 0
// Efface l'écran
EFFACEECRAN   // Abrév. EFF
// "TANT QUE" tq est plus petit que 10.
TANT QUE :tq < 10 [
                    ECL :tq
                    // incrémentation de tq
                   INC :tq
                   // Recommence la boucle.
                      ]
ECL [ C'est fini. ]
// termine le programme.
```

```
10 Dim matrice = New Integer(3, 2) {{1, 2, 3}, {2, 3, 4}, {3, 4, 5}, {4, 5, 6}}
```

```
DONNE "matrice [[1 2 3] [2 3 4][3 4 5][4 5 6]]
```

```
10 rem Création d'un tableau de 100 éléments.
20 Dim arr(99) as Integer
30 rem Remplissage du tableau.
40 Dim rnd As new Random()
50 For ctr = 0 To arr.GetUpperBound(0)
60 arr(ctr) = rnd.Next()
70 Next
```

```
// Création d'une liste de 100 éléments.
 DONNE "frontière 99
 DONNE "arr []
 // Remplissage de la liste :arr .
 REPETE : frontière
                DONNE "arr PHRASE :arr RANDOM  0 1
```

```
10 CLR DIM TAB(4,3) as Integer
20 FOR R = 1 TO 4
30 PRINT "Question n° " ; R
40 PRINT "1-OUI 2-NON 3-INDECIS
50 PRINT "Quelle est ta réponse ?" ;
60 GET C : IF C < 1 OR C > 3 THEN 60
70 PRINT C : PRINT
80 TAB(R,C) = TAB(R,C) + 1 : REM actualise l'élément
90 NEXT R
100 PRINT
110 PRINT "Veux-tu entrer un autre élément ? " : PRINT
    "Réponse (O/N) " ;
120 GET A$ : IF A$ = "" THEN 120
130 IF A$ = "O" THEN 10
140 IF A$ * "N" THEN 120
150 CLR : PRINT "Les réponses totales sont : " : PRINT
160 PRINT SPC(18) ; "Réponses : "
170 PRINT "QUESTION","OUI" , "NON","INDECIS"
180 PRINT "--------------   ------ ------- --------------
190 FOR R = 1 TO 4
200 PRINT TAB(R,1), TAB(R,2), TAB(R,3)
210 NEXT R
```

```
EFFACETEXTE // Abrév. EFFTXT
DONNE "R 1 DONNE "TAB [ [0 0 0 ] [0 0 0 ] [0 0 0 ] [0 0 0 ]]
DONNE "A "O
TANT QUE ( :A <> "O ) [
TANT QUE ( :R <= 4) [
               ECL PH [Question n°  ] :R
               ECL [ 1-OUI 2-NON 3-INDECIS ]
               ECL [Quelle est ta réponse ? ]
               DONNE "C LISNOMBRE
              TESTE (( :C < 1) OU ( :C > 3))
               SIVRAI  EFFTXT
               SIFAUX [
                          ECL :C
                          SOIT "V SAISIS :TAB [ :R :C ]  +1
                          REMPLACE :TAB :V [ :R :C ]
                          INC :R
                         ]
                         ]
ECL "
ECL [ Veux-tu entrer un autre élément ? ]
ECL [ Réponse (O/N) :] DONNE "A PREMIER LISLISTE
TANTQUE ((:A <> "O) ET (:A <> "N))
                        DONNE "A PREMIER LISLISTE
                                ]
EFFTXT ECL [Les réponses totales sont : ]
```

	ECRISLIGNE [Réponses :] ECL [QUESTION OUI NON INDECIS] ECL [---------- ------ ------- ----------] REPETE 4 ECL PH "/e18 PH SAISIS :TAB [BOUCLE 1] PH SAISIS :TAB [BOUCLE 2] SAISIS :TAB [BOUCLE 3]

a) <u>Avec le langage Python.</u>

Le langage Python semble avoir la préférence des professeurs de lycée, peut-être en raison d'une syntaxe proche du langage C. De par cette similitude qui justifie le tableau suivant, d'autres exemples d'algorithmes simples donnent ligne par ligne la traduction syntaxique dans l'un et l'autre de ces deux langages de programmation.

Langage Python	**Langage LOGO**

VarA=2 # la variable VarA vaut 2
b4=3.1 # la variable b4 vaut 3,1
s=b4+3 # s vaut 6,1
c,d=4.2 , 7 # c vaut 4,2 et d vaut 7
e= " bonjour" # e vaut " bonjour"
E= "Hello!" # E vaut "Hello!"
#E différente de e car minuscules et
# majuscules sont considérées 	
# comme différentes. 	
	soit "A 2
	soit "b4 3.1 // ou 3,1
	soit "s :b4 + 3 // :s vaut 6,1
	soit "c 4.2 soit "d 7 // :c vaut 4,2 et :d vaut 7
	soit "e "bonjour // :e vaut "bonjour
	soit "E "Hello! // :E vaut "Hello!
	// minuscules et majuscules sont considérées comme
	// différentes, sauf dans l'écriture des primitives LOGO.

def f(x) : Return x * x – x + 41	pour f :x rends (:x ^2) - :x +4 // rends équivalent à répondre. fin
valeursx = [x for x in range(20)] # permet d'itérer pour x de 0 à 19.	soit "valeursx listevars 0 19 20 // permet de construire une liste de valeurs de 0 à 19.

def soleq(a,b) :
if a != 0 :
# si a est différent de 0 	
return b / a
 	
def h(x) :
if x<0 :y=2*x+3
elif x<2 : # ici 0<=x<2
y=3–x
else :
# ici x>=2 	
y=x*x–3
return y
	pour soleq :a :b
	si :a <> 0 rends :b / :a
	/* L'exemple en Python à l'origine de la traduction en langage LOGO n'indique pas quoi faire si :a est égal à 0. */
	fin
	
	pour h :x
	teste :x < 0
	sivrai soit "y 2 * :x +3
	sifaux [
	teste :x < 2 // ici 0 <= :x <2
	sivrai soit "y 3 - :x
	sifaux soit "y (:x ^ 2) –3 //ici :x >= 2
	]
	répondre :y // répondre équivalent à rends.
	fin

a = [9, 7, 6, 9]
>>> a[0] # réponse: 9
>>> a[2] # réponse: 6
a = [9, 7, 6, 9]
	soit "a [9 7 6 9] // Les listes LOGO commencent à l'item n°1.
	ecl item :a 1 // réponse: 9
	ecl item :a 3 // réponse: 6
	soit "a [9 7 6 9]

>>> len(a) # réponse: 4
a = [9, 7, 6, 9]
>>> a.sort() # réponse: [6, 7, 9, 9]
a = [9, 7, 6, 9]
>>> a.append(2) # réponse: [9, 7, 6, 9, 2]
a = range(5, 15, 2)
>>> list(a) # réponse: [5, 7, 9, 11, 13]
	ecl card :a // réponse: 4
	soit "a [9 7 6 9]
	ecl trieitem :a [< nombre] // réponse: [6 7 9 9]
	soit "a [9 7 6 9]
	soit "a PH :a 2 // réponse: [9 7 6 9 2]
	soit "a listevars 5 15 5
	ecl :a // réponse: [5 7 9 11 13]

montexte = "Blablablibloubloublou."
>>> len(montexte) # réponse: 22
montexte + " Merci."
# réponse: 'Blablablibloubloublou. Merci.' 	
maliste = ["zéro", "un", "deux", "trois", "quatre", "cinq"]
>>> maliste[-1]
# on demande le dernier élément 	
# réponse: 'cinq' 	
maliste[-2]
# l'avant-dernier 	
# réponse: 'quatre' 	
maliste[-3] # l'avant-avant-dernier (ou antépénultième), et ainsi de suite…
# réponse: 'trois' 	
	soit "montexte "Blablablibloubloublou."
	ecl card :montexte // réponse: 22
	soit "montexte PH :montexte "Merci.
	// réponse: Blablablibloubloublou. Merci.
	soit "maliste [zéro un deux trois quatre cinq]
	ecl dernier :maliste // réponse: cinq
	// ou
	ecl item :maliste card :maliste
	// réponse: cinq
	ecl item :maliste (card :maliste) –1
	// l'avant-dernier
	// réponse: quatre
	ecl item :maliste (card :maliste) –2
	// l'avant-avant-dernier
	// réponse: trois et ainsi de suite…

for i in range(5): # pour i allant de 0 à 4
  print("blabla")
	répète 5 ecl "blabla

from random import randint
 	
d = randint(1, 6)
print(d)
 	
if d == 1:
  print("gagné")
else:
  if d == 5:
  print("gagné")
  else:
  if d == 6:
  print("gagné")
  else:
  print("perdu")
	// Inutile en LOGO : la fonction randint existe en standard
	// sous la désignation random.
	soit "d random 1 6
	écrisligne :d
	
	si :d = 1 ecl "gagné
	sifaux [
	si :d = 5 ecl "gagné
	sifaux [
	si :d = 6 ecl "gagné
	sifaux ecl "perdu
	]
	]

Retour à la table des rubriques >>