LangageGraphique : un outil de création graphique pour les mathématiques

André Boileau
Section didactique, UQAM

Introduction

Il n'y a pas si longtemps, on disait que « la géométrie est l'art de raisonner correctement sur des figures fausses ». Cette affirmation pouvait surprendre le débutant car, en même temps, le curriculum de géométrie insistait sur des constructions exactes utilisant comme seuls outils la règle (non graduée) et le compas. Mais les vieux routiers savaient toute la sagesse contenue dans cette affirmation : une figure exacte pouvait amener les débutants à se satisfaire de ce qu'ils voyaient et qu'ils pouvaient mesurer, alors qu'une figure imprécise incitait à l'argumentation. Cependant ces même experts avaient parfois recours à des figures plus précises, quand leur intuition visuelle et leurs raisonnements ne les conduisaient pas à la solution recherchée.

La situation a bien changé aujourd'hui. Les outils technologiques permettent maintenant des dessins mathématiques qui sont non seulement précis, mais aussi dynamiques. Ainsi, avec un logiciel de géométrie dynamique (tel Cabri-géomètre ou Geometer's Sketchpad), on peut obtenir des figures précises qu'on peut faire varier interactivement. Tant et si bien que les débutants peuvent plus que jamais se satisfaire d'une constatation expérimentale d'une propriété, convaincus par cette apparente capacité de ces logiciels de permettre l'étude précise de « tous les cas ». Aux vieux routiers de faire ressortir que ce genre de certitude ne s'accompagne pas d'un sentiment de compréhension que seule une argumentation satisfaisante peut apporter.

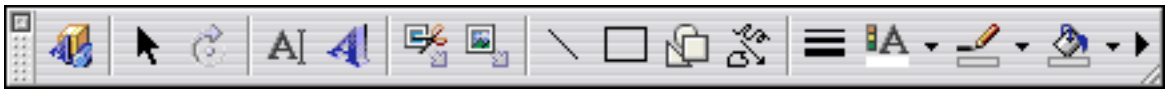
Quoi qu'il en soit, l'éventail de logiciels mathématiques permettant de réaliser des figures mathématiques précises est aujourd'hui d'une grande richesse et d'une non moins grande diversité. Outre les logiciels de géométrie dynamique que nous venons d'évoquer, on peut mentionner les calculatrices graphiques, les chiffriers électroniques (tel Excel), les systèmes de calcul symbolique (tels Derive, Maple et Mathematica), les micro-mondes programmés dans divers environnements (tel Logo), etc. Et ceci sans compter des logiciels qui, sans que leur vocation soit spécifiquement mathématique, permettent d'engendrer des représentations graphiques utiles aux mathématiques, tels certains programmes de dessin vectoriel (comme Illustrator).

Malheureusement, tout le monde n'a pas accès à ces merveilleux outils. Confronté à ce problème en préparant le cours **Progiciels dans l'enseignement des mathématiques** de la concentration *Mathématiques* du *Bacc. En Enseignement Secondaire* de l'UQAM, j'ai voulu fournir à mes étudiants un outil de production de graphiques mathématiques qui serait utilisable partout (ou presque). À tort ou à raison, je me suis dit qu'à peu près tout le monde avait accès à la suite *Office* de *Microsoft*, et j'ai mis au point une extension de *Microsoft Word* (qui peut aussi être adaptée pour *Microsoft Excel*) susceptible de répondre à certains besoins d'un professeur de mathématiques voulant intégrer des graphiques dans ses documents mathématiques. Et même ceux qui ont la chance d'avoir accès à toute une kyrielle de logiciels mathématiques pourraient être agréablement surpris des possibilités de ce nouvel environnement.

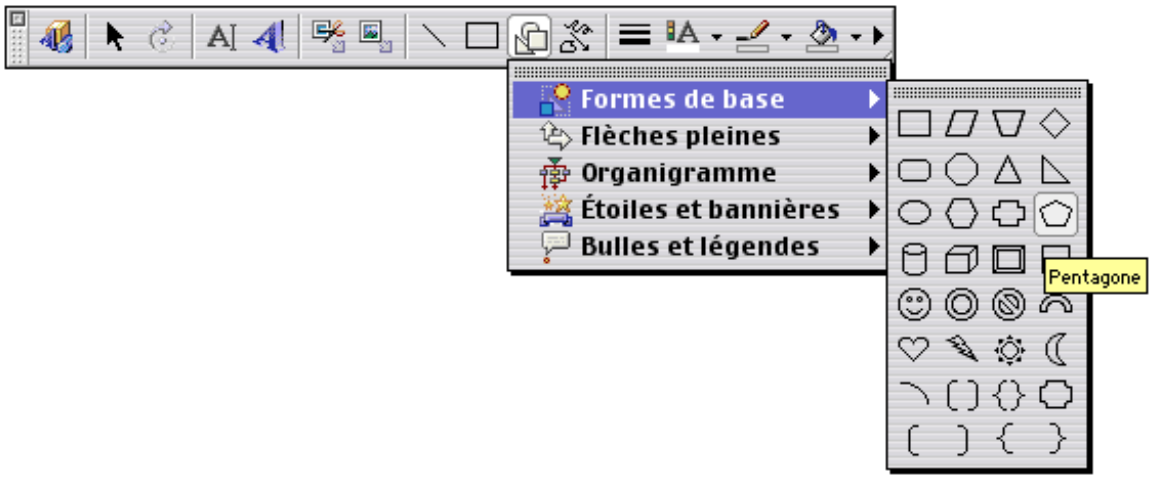
Les outils graphiques dans *Word*

Si vous n'avez jamais eu l'idée d'utiliser *Word* pour produire des dessins, vous serez surpris des nombreuses possibilités mises à votre disposition. Dans ce qui suit, je vais décrire comment utiliser les outils de dessin vectoriel de *Word* (disponibles dans toute la suite *Office*) pour créer certains graphiques utiles en mathématiques. Je ne vise pas une description exhaustive de tous les outils disponibles, je veux seulement illustrer certaines des possibilités.

Pour accéder à la barre des outils de dessin de *Word*, on peut sélectionner le sous-item **Dessin** de l'item **Barres d'outils** du menu **Affichage**. On peut aussi cliquer sur l'icône **Dessin** de la barre d'outils **Standard**, si celle-ci est affichée. Une fois affichée, cette barre d'outils ressemble au dessin ci-dessous[1] :



À première vue, ça peut sembler assez inoffensif. Mais cette barre d'outils recèle des possibilités cachées, comme nous le montre la copie d'écran suivante :



Plaçons-nous dans l'optique de quelqu'un qui veut produire des graphiques précis. Si nous voulons obtenir une forme (un rectangle, par exemple) qui apparaît dans les items de la barre d'outils **Dessins**, nous n'avons qu'à sélectionner (par un clic) l'icône correspondante et qu'à décrire ensuite ***gestuellement*** (par un « glisser » de souris) la position et la dimension de cette forme. Notons en passant que nous pourrons toujours modifier (toujours gestuellement, par un « glisser » de souris) la position et les dimensions de notre forme.

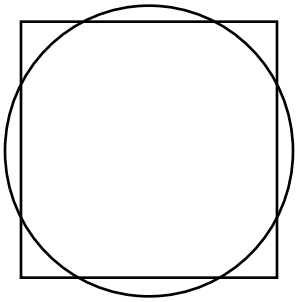
Même si la forme désirée ne correspond à aucun des items de la barre d'outils **Dessins**, tout n'est pas perdu. Si, par exemple, nous voulions tracer un carré, il suffirait d'indiquer à Word que nous désirons tracer un rectangle avec une contrainte : ceci se fait en maintenant enfoncée la touche *Majuscule* pendant qu'on fait glisser la souris. De même, si nous voulions un triangle équilatéral, il suffirait de tracer un triangle isocèle avec contrainte (toujours indiquée par *Majuscule*). Notons, toujours en passant, que cette même touche *Majuscule* nous permet de redimensionner une forme avec la contrainte que les facteurs horizontal et vertical de

[1] La disposition peut varier selon la version de *Word* utilisée, mais aussi selon la forme que vous donnerez à cette palette (grâce à la « poignée » de redimentionnement en bas à droite.

changement d'échelle demeurent identiques (ce qui nous permet d'obtenir une *figure semblable* à la figure initiale).

Il y a une autre façon d'imposer une contrainte à une forme. Supposons que nous voulions obtenir un rectangle dont la base mesure deux fois la hauteur. On commence d'abord par tracer un rectangle quelconque. Puis, tandis qu'il est encore sélectionné, on fait apparaître le menu local correspondant (par un clic du bouton droit avec *Windows*, par un clic avec la touche *Contrôle* enfoncée avec *Macintosh*) et on choisit l'item **Format de la forme...**, suivi de l'onglet **Taille** : on peut alors indiquer ***textuellement*** (via le clavier) la hauteur et la largeur (par exemple 1 cm et 2 cm). Remarquons qu'on pourrait aussi spécifier ici l'angle de rotation à appliquer à cette forme.

Considérons un autre exemple : supposons que nous voulions tracer simultanément un cercle et un carré de même aire. Si c est la mesure du côté du carré et r la mesure du rayon du cercle, on devra avoir $c^2 = \pi r^2$, et donc $c = \sqrt{\pi}\, r$. On peut donc tracer tout d'abord un cercle (une ellipse avec contrainte, via touche Majuscule) et un rectangle quelconques. Puis on fixe (comme au paragraphe précédent) les dimensions du cercle à 1cm ([2]) et celles du carré à $\sqrt{\pi} \approx 1,772453851$ cm. Si on veut obtenir une figure comme ci-contre où nos figures sont centrées et superposées, il faudra en outre



- spécifier qu'on ne veut aucun remplissage pour nos formes (via l'outil « pot de peinture » de la palette **Dessin**)
- donner le même centre à nos deux formes (via les commandes **Aligner le centre**[3] et **Aligner le milieu** appliquées quand les deux formes sont sélectionnées)
- regrouper nos deux formes en un seul objet (via la commande **Grouper**, appliquée quand les deux formes sont sélectionnées)
- redimensionner le tout pour obtenir une figure semblable de la dimension voulue (ne pas oublier la touche *Majuscule*).

Cependant, malgré toute cette versatilité, on se heurte rapidement à des figures qui sont, à toutes fins pratiques, impossibles de réaliser **exactement** dans *Word*. Il peut arriver qu'une forme ne puisse tout simplement pas être obtenue à partir des formes disponibles. Par exemple, si on veut obtenir un polygone régulier à 17 côtés (pour illustrer les polygones constructibles avec la règle et le compas).

Il est aussi très difficile en général de réaliser des figures qui impliquent un agencement précis de plusieurs formes simples. Par exemple, supposons que nous voulions tracer un rectangle avec ses deux diagonales. On peut essayer de tracer tout d'abord un rectangle, puis deux segments en essayant de choisir les sommets du rectangle comme points de départ et d'arrivée, pour terminer en regroupant les trois formes. Mais le choix des extrémités de ces « diagonales » n'est qu'approximatif[4], et on n'obtient donc pas la figure « exacte » recherchée.

---

[2] Pour obtenir un rayon de 1 cm, il faudra spécifier une hauteur et une largeur de 2 cm.

[3] Les commandes **Aligner le centre**, **Aligner le milieu** et **Grouper** sont disponibles à partir de la première icône de la barre d'outils **Dessin**.

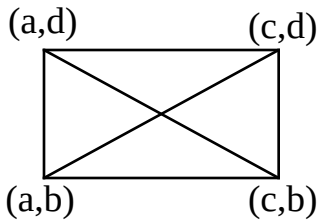
[4] Dans certains cas, il est impossible qu'elles soient exactes. Par exemple si notre rectangle est lui-même fruit d'une autre construction et a ses sommets en des endroits qui ne correspondent pas exactement à des pixels de l'écran.

On peut cependant déterminer ***textuellement*** non seulement les dimensions mais aussi les positions exactes de tous nos éléments graphiques, et ce de la manière suivante : après avoir sélectionné la forme, on fait apparaître le menu local et on choisit l'item **Format de la forme...**, suivi de l'onglet **Disposition** : après un clic sur le bouton **Avancée...**, on peut alors spécifier ***textuellement*** (via le clavier) la position absolue (horizontale et verticale) de la forme sélectionnée.

De cette façon, on peut obtenir une description exacte[5] de chacune des formes constituant la figure désirée, mais il faut avouer que la lourdeur de tout le processus nous fait payer un prix très élevé pour obtenir la précision que nous recherchons. Soulignons encore une fois que cette précision a été rendue possible par une description ***textuelle*** de toutes les dimensions et positions en présence.

On pourrait imaginer une façon de simplifier la démarche de création de figures exactes en cherchant un nouvel équilibre entre les composantes ***gestuelles*** et ***textuelles*** des actions à entreprendre. Supposons, par exemple, qu'on puisse taper *quelque part*

- Segment a , b , c , d
  pour tracer un segment d'extrémités (a,b) et (c,d)
- Rectangle a , b , c , d
  pour tracer un rectangle ayant les points (a,b), (a,d), (c,b) et (c,d) comme sommets
- Cercle a , b , r
  pour tracer un cercle de centre (a,b) et de rayon r.



Voyons comment ceci pourrait nous aider à accomplir plus simplement les tâches précédentes. Pour obtenir un rectangle muni de ses diagonales, il suffirait de taper

- Rectangle a , b , c , d
- Segment a , b , c , d
- Segment a , d , c , b

et pour obtenir notre carré et notre cercle concentriques et de même aire, on devrait taper

- Cercle a , b , r
- Rectangle $a-\frac{1}{2}c\,,\,b-\frac{1}{2}c\,,\,a+\frac{1}{2}c\,,\,b+\frac{1}{2}c$, où $c=\sqrt{\pi}\,r$ est le côté du carré.

Présentation de *LangageGraphique*

« ***LangageGraphique*** » est le nom d'un fichier *Word*[6] qui permet de réaliser ce que nous venons tout juste de décrire. Pour illustrer son fonctionnement, voyons comment l'utiliser dans le cas du carré et du cercle de même aire.
Nous supposerons que nous voulons insérer ce dessin dans un document *Word* sur lequel nous travaillons. Sans fermer ledit document, nous ouvrons donc le fichier ***LangageGraphique***. Si la protection contre les macro-virus dans *Word* a été activée, nous devrons, par un clic sur le bouton approprié, donner la permission d'**Activer les macros** contenues dans le fichier ***LangageGraphique*** : ne vous en faites pas, il n'y a aucun risque.

---

[5] Cette exactitude est cependant limitée par la précision des nombres qui sont entrés textuellement, précision qui ne peut toujours être absolue (exemple : les nombres irrationnels). Mais cette précision est, en général, bien suffisante pour que la figure obtenue soit visuellement exacte.

[6] On peut télécharger ce fichier à partir de la page web suivante
http://www.math.uqam.ca/_boileau/LangageGraphique.html

Une fois le fichier ouvert, nous verrons une page ne contenant que les « boutons » suivants :

Tracer le graphique **Ajouter du texte** **Effacer tout**

Mais avant de les utiliser, nous devons aller écrire *quelque part* les instructions nécessaires. Cet endroit mystérieux est l'éditeur Visual Basic. Mais rassurez-vous, nous n'aurons pas à faire de la programmation : tout a été fait pas nous, et il ne nous reste qu'à ajouter une description mathématique de la figure à tracer.

Nous allons donc sélectionner le sous-item **Visual Basic Editor** de l'item **Macro** du menu **Outil**, et nous obtenons[7] une page contenant un programme. Rendons-nous immédiatement à la toute fin de ce programme, où nous trouverons les lignes suivantes :

```
Sub Graphique()

End Sub
```

C'est entre ces deux lignes que nous devrons taper nos instructions. Dans notre cas, nous obtiendrons ceci :

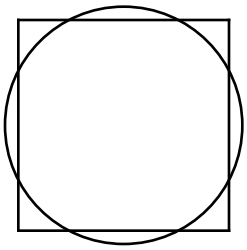
```
Sub Graphique()
   a = 0
   b = 0
   r = 100
   c = r * Sqr(pi)
  Cercle a , b , r
  Rectangle a - c/2 , b - c/2 , a + c/2 , b + c/2
End Sub
```

Notons que nous aurons aussi pu taper directement

```
Sub Graphique()
  Cercle 0 , 0 , 100
  Rectangle  -100*Sqr(pi) / 2,  -100*Sqr(pi) / 2 ,  100*Sqr(pi) / 2,  100*Sqr(pi) / 2
End Sub
```

L'important est de donner des valeurs numériques de façon à ce que le cercle et le rectangle puissent bien être tracés. Notons en passant que l'origine du système d'axes utilisé est située à peu près au milieu d'une page *Word*, et que l'unité de longueur utilisée est le pixel. Mais ceci a relativement peu d'importance puisqu'il nous sera facile de changer ***gestuellement*** la position et les dimensions du graphique ainsi obtenu.

Quand nous avons fini d'entrer nos instructions, nous retournons à notre page *Word* en choisissant l'item **Fermer et retourner à Microsoft Word** du menu **Fichier**. Ne fermez pas inutilement la fenêtre de l'éditeur Visual Basic, car il vous faudrait alors la réouvrir à votre prochaine visite.



Pour obtenir (enfin!) la figure désirée, vous devez maintenant faire un **double-clic** sur le bouton jaune **Tracer le graphique**. Si tout se passe bien, vos efforts seront récompensés par une figure ci-contre, que vous pourrez modifier (exemple : changer les dimensions) avant de la **copier** et de la **coller** dans le document *Word* sur lequel vous travaillez. Notez en passant que votre tâche sera facilitée par le fait

[7] On peut aussi obtenir cette page par un clic sur l'icône **Visual Basic Editor** de la barre d'Outils **Visual Basic**, si celle-ci est affichée. [Et on peut l'afficher en sélectionnant le sous-item **Visual Basic** de l'item **Barre d'outils** du menu **Affichage**.]

que le programme regroupe automatiquement tous les éléments tracés, ce qui facilite tant les redimensionnements que les copies.

Si Visual Basic vous signale une erreur, vous serez amenés à la ligne où l'erreur a été détectée. Vérifiez que vous avez bien tapé l'instruction exactement comme elle apparaît ci-dessus, en m'oubliant pas les espaces, virgules et parenthèses. Corrigez la ou les coquilles, retournez à *Word* et redemandez de **Tracer le graphique**.

Remède pour un petit défaut de Microsoft Word

Il peut arriver qu'essayant de tracer votre premier graphique, Microsoft Word vous donne plutôt un message d'erreur ressemblant à « *Projet ou bibliothèque introuvable* ». Pour corriger la situation, il suffit alors de suivre les étapes ci-dessous :

- Faites un clic sur le bouton « Ok » de la fenêtre affichant le message d'erreur : ceci vous conduira à l'éditeur de Visual Basic pour Applications.
- Sélectionnez d'abord l'item **Réinitialiser** du menu **Exécution**.
- Sélectionnez ensuite l'item **Références**... du menu **Outils** : ceci fera apparaître une fenêtre où il vous faudra décocher tous les items dont la description commence par « MANQUANT ».
  Vous pourrez ensuite fermer cette fenêtre par un clic sur le bouton « Ok ».
- Quittez maintenant l'éditeur VBA (Visual Basic pour Applications) en choisissant l'item **Fermer et retourner à Microsoft Word** du menu **Fichier**. Pour éviter que ce problème ne se produise à nouveau, vous devrez enregistrer les changements à ***LangageGraphique*** que vous venez de faire (item **Enregistrer** du menu **Fichier**).

Un autre exemple d'utilisation de *LangageGraphique*

Après avoir vu comment réaliser le cercle et le carré de même aire, la réalisation d'un rectangle avec ses deux diagonales ne devrait pas poser de problèmes et est laissée en exercice. Nous allons plutôt nous attacher à réaliser le tracé d'un polygone régulier à 17 côtés parce que ce sera l'occasion d'introduire plusieurs autres possibilités de ***LangageGraphique.***

Nous devrons donc tracer 17 segments, dont les extrémités correspondront aux 17 sommets du polygone. Dans le cas d'un polygone inscrit dans un cercle centré en (0,0) et de rayon r, disposé de façon telle qu'un des sommets repose sur l'axe des x positifs, on détermine que les coordonnées des sommets seront

$$(x_k, y_k) = \left(r\cos\left(\frac{2\pi}{17}k\right), r\sin\left(\frac{2\pi}{17}k\right)\right)$$, où k est un entier variant de 0 à 16.

Nous pourrions donc réaliser ce polygone via une série de commandes (une pour chaque segment à tracer) comme suit :

Segment $x_0\ y_0\ x_1\ y_1$
Segment $x_1\ y_1\ x_2\ y_2$
...
Segment $x_{16}\ y_{16}\ x_0\ y_0$

Mais nous pouvons profiter que nous travaillons dans le cadre d'un langage de programmation disposant de commandes pour réaliser des répétitions, et utiliser plutôt les instructions suivantes :

```
Sub Graphique()
    r = 100   '  r est la mesure du rayon du cercle circonscrit au polygone
    n = 17    '  n est le nombre de côtés du polygone
    For k = 0 To n-1
       Segment r*cos(k*2*pi/n), r*sin(k*2*pi/n), r*cos((k+1)*2*pi/n), r*sin((k+1)*2*pi/n)
    Next k
End Sub
```

Nous avons utilisé ici la structure de contrôle

For variable = valeur_initiale To valeur_finale
liste_d'instructions_pouvant_contenir_la_variable
Next variable

qui, successivement,

- exécute la *liste_d'instructions* en donnant à la *variable* la valeur *valeur_initiale*
- exécute la *liste_d'instructions* en donnant à la *variable* la valeur *valeur_initiale* + 1
- exécute la *liste_d'instructions* en donnant à la *variable* la valeur *valeur_initiale* + 2
- ...
- exécute la *liste_d'instructions* en donnant à la *variable* la valeur *valeur_finale* [car *valeur_finale* = *valeur_initiale* +(*valeur_finale* - *valeur_initiale*)].

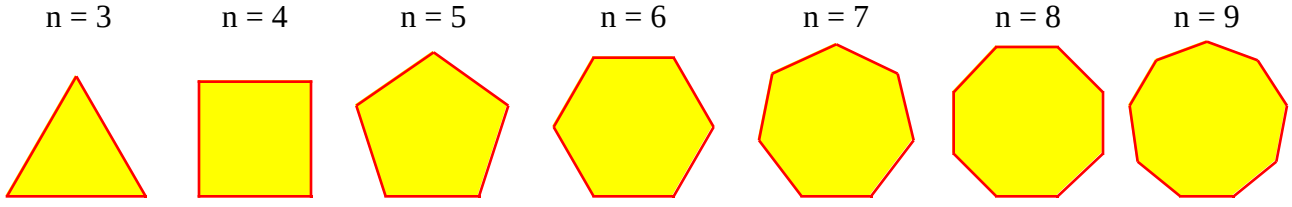
En se rappelant que, on voit donc que l'on obtient des instructions en 5 lignes qui font le même travail que nos 17 instructions initiales. Mais ce n'est pas tout : en changeant la valeur de n, on peut tracer tous les polygones réguliers, quel que soit leur nombre de côtés. Ceci justifie amplement le choix d'utiliser des variables au lieu d'utiliser directement des valeurs, un autre avantage étant de rendre nos instructions plus lisibles.

Notons que, jusqu'à présent, nous avons utilisé le paradigme de la *géométrie analytique* pour tracer nos graphiques. Nous pourrions tout aussi bien utiliser le paradigme de la *géométrie de la tortue* cher aux adeptes de Logo, car ***LangageGraphique*** peut piloter une tortue avec les instructions *Avance*, *Droite*, *FixePosition*, *FixeCap*, *LeveCrayon*, *BaisseCrayon* et *VideEcran*. Dans ce contexte, nous aurions pu utiliser les instructions

```
Sub Graphique()   ' les commentaires sont en vert
    c = 20     ' c est la mesure d'un côté du polygone
    n = 17     ' n est le nombre de côtés du polygone
    For k = 0 To n-1
       Avance c
       Droite 360/n   '  pour la tortue, les angles sont en degrés
    Next k
End Sub
```

et nous aurions obtenu un polygone régulier à 17 côtés dont l'inclinaison et la mesure des côtés auraient été quelque peu différentes. (Exercice : quelle valeur aurait-on dû donner au côté c pour que le polygone soit inscrit dans un cercle de rayon r?)

Nous pouvons maintenant utiliser une des deux versions précédentes, disons celle utilisant le paradigme de la tortue, pour produire la figure suivante



Cette figure a été réalisée à l'aide des instructions suivantes, exécutées 6 fois en donnant successivement à n des valeurs de 3 à 9.

```
Sub Graphique()
    n = 3      ' n est le nombre de côtés du polygone
    r = 30     ' r est le rayon du cercle circonscrit
    c = 2*r*sin(pi/n)     ' c est la mesure d'un côté du polygone
    CouleurCrayon 255,0,0  ' composante rouge au maximum
    CouleurRemplissage 255, 255, 0 ' composantes rouge et verte au maximum
    DebutRemplir   ' la figure fermée formée par les segments suivants sera remplie
    Droite -90   ' pour que la tortue pointe vers la gauche (base horizontale)
    For k = 0 To n-1
       Avance c
       Droite 360/n
    Next k
    FinRemplir  ' remplir la figure amorcée avec DebutRemplir
End Sub
```

Après chaque exécution, on a copié la figure obtenue pour la coller ensuite dans le présent document. Les annotations « n = ... » ont été réalisées en faisant apparaître une zone de texte (via un double-clic sur le bouton jaune **Ajouter du texte**) que l'on complète et l'on dispose ensuite gestuellement.

Notons au passage le tandem d'instructions **DebutRemplir ... FinRemplir** donnant l'instruction de remplir une figure fermée formée de segments, ainsi que les instructions **CouleurCrayon** et **CouleurRemplissage** permettant de spécifier la couleur des traits et des remplissages en utilisant la codification rouge-vert-bleu pour décrire les couleurs.

Les primitives graphiques de *LangageGraphique*

Après ces quelques exemples d'utilisation de ***LangageGraphique***, voici maintenant une description des principales commandes disponibles. Dans la colonne **Procédure**, vous trouverez la forme générale des commandes avec, le cas échéant, un exemple d'utilisation *en italique*. La colonne **Alt.** donne, le cas échéant, la forme abrégée de la commande : par exemple, on peut utiliser **VE** au lieu de **ViderEcran**. Je vous rappelle que :

- le système de coordonnées sous-jacent se trouve approximativement au centre de la feuille
- au départ, la tortue est à l'origine du système de coordonnées et « regarde » vers le haut
- les distances et les coordonnées sont exprimées en pixels (qui peuvent cependant être des nombres décimaux, exprimés avec des *points* et non des *virgules [1.23 et non 1,23]*)
- les angles sont exprimés en degrés. (Attention cependant aux fonctions trigonométriques, qui supposent que leurs arguments sont en radians, **cosD** et **sinD** faisant exception.)

Commandes pour tracer de formes décrites avec des coordonnées

Procédure	Alt.	Commentaires
Segment x1, y1, x2, y2 *Exemple : Segment 0,0,100,50*		Trace un segment joignant deux points dont on donne les coordonnées.
Rectangle x1, y1, x2, y2 *Exemple : Rectangle 0,0,100,50*		Dessine un rectangle (sans remplissage) dont on spécifie les coordonnées d'une diagonale.
RectanglePlein x1, y1, x2, y2 *Exemple : RectanglePlein 0,0,100,50*		Dessine un rectangle rempli avec la couleur de remplissage.

Ellipse x1, y1, x2, y2, Remplir, Angle *Exemple : Ellipse 0,0,100,50,true,45*		Trace l'ellipse déterminée par le rectangle, et la tourne ensuite. Remplir peut prendre les valeurs « true » ou « false ».
Cercle x, y, R *Exemple : Cercle 0,0,100*		Trace le cercle de centre et rayon donnés
Disque x, y, R *Exemple : Disque 0,0,100*		Trace le disque (cercle plein) de centre et rayon donnés
Arc x, y, rayon, depart, arrivee *Exemple : Arc 0,0,100,45,135*		Trace un arc déterminé par un cercle, un angle de départ et un angle d'arrivée (sens +).
Grille x, y, dx, dy, nx, ny *Exemple : Grille 0,0,10,10,20,10*		Dessine une grille à la position (x,y) • nx colonnes de dimension dx • ny lignes de dimension dy

Commandes pour tracer de formes décrites avec des mouvements de la tortue

Procédure	**Alt.**	**Commentaires**
Avance distance *Exemple : Avance 30*	Av	Fait avancer la tortue d'un certain nombre de pas (pixels). Pour reculer, utiliser un nombre de pas négatif.
Droite angle *Exemple : Droite 90*	Dr	Fait tourner la tortue vers la droite d'un certain nombre de degrés. Pour tourner à gauche, utiliser un angle négatif..
FixePosition x, y *Exemple : FixePosition 0,0*	FPos	Fixe la position de la tortue.
FixeCap angle *Exemple : FixeCap45*	FCap	Fixe le cap de la tortue.
LeveCrayon		Lève le crayon de la tortue. Si la tortue se déplace, elle ne laissera pas de trace.
BaisseCrayon		Baissele crayon de la tortue. Si la tortue se déplace, elle laissera une trace.

Commandes pour modifier les caractéristiques des formes

Procédure	**Alt.**	**Commentaires**
ViderEcran	VE	Efface tout le dessin
CouleurCrayon R, V, B *Exemple : CouleurCrayon 0,0,0*	CC	Détermine la couleur du crayon de la tortue. Le paramètres peuvent varier de 0 à 255.
TailleCrayon épaisseur *Exemple : TailleCrayon 2*	TC	Détermine l'épaisseur du crayon de la tortue.
DebutRemplir		Débute un polygone à remplir.
FinRemplir		Fin du polygone à remplir.
CouleurRemplissage R, V, B *Exemple : CouleurRemplissage 255,255,0*	CR	Détermine la couleur des remplissages. Le paramètres peuvent varier de 0 à 255.
Dessous Dessus		L'élément sélectionné est placé *au desssous* ou *au dessus* de tous les autres.
Hasard (min, max) *Exemple : CC Hasard (0,256),0,0*		Fonction retournant un nombre x choisi au hasard tel que $min \leq x < max$
Texte chaine, x, y, Police, Taille, Gras, Italique *Exemple :* *Texte "Bonjour",0,0, "Times",12,true,false*		Dessine une chaîne de caractères à la position et avec les caractéristiques spécifiées.

Commandes pour faire des conversions d'unités

Procédure	Alt.	Commentaires
sinD (x) *Exemple : sinD (45)*		Fonction sinus où l'angle est exprimé en degrés. La fonction sin(x) suppose que x est en radians.
cosD (x) *Exemple : cosD (45)*		Fonction cosinus où l'angle est exprimé en degrés. La fonction cos(x) suppose que x est en radians.
cmApixels (x) *Exemple : Avance cmApixels (1)*		Fonction qui calcule le nombre de pixels dans x centimètres.
pixelsAcm (x) *Exemple : x = pixelsAcm (1)*		Fonction qui calcule le nombre de centimètres correspondant à x pixels.

Commandes pour tracer des courbes de fonctions

Procédure	Alt.	Commentaires
RegionsFonctions xmin, xmax, ymin, ymax *Exemple :RegionsFonctions -2*pi, 2*pi, -2, 2*		Spécifie quelle région du plan mathématique sera utilisée pour visualiser les fonctions. Par défaut, on déclare implicitement RegionsFonctions -10, 10, -10, 10
FenetreFonctions gauche,droite,haut,bas *Exemple :FenetresFonctions 0, 300, 0, 200*		Spécifie quelle région du plan informatique (le document Word) sera utilisée pour représenter les fonctions. Par défaut, on déclare implicitement FenetreFonctions -250, 250, -200, 200
TracerAxes		Noter que les axes tracés se limitent (pour l'instant) à deux segments avec flèches mais sans graduations.
TracerFonction numéro *Exemple TracerFonction 3*		Commande le tracé de la fonction dont le numéro est spécifié.
PasFonctions x *Exemple : PasFonctions 2*		Par défaut, la fonction est calculée à chaque point d'écran. On peut augmenter (x<1) ou diminuer (x>1) le nombre de points utilisés.
PointsRelies PointsNonRelies		Tout programme de tracé de fonctions n'évalue la fonction qu'en un nombre fini de points. On peut spécifier si ces points seront réliés entre eux ou pas via ces commandes.

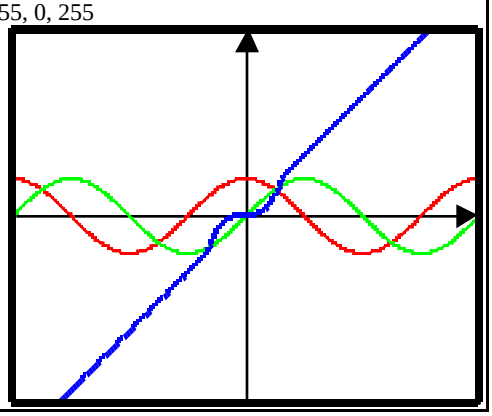
Quelques mots d'explication sont nécessaires pour ce groupe de commandes. On doit tout d'abord spécifier les fonctions que nous voulons tracer. Ceci se fait dans la fonction **F**, qui précède immédiatement la procédure **Graphique** dans **LangageGraphique**. On doit entrer la définition de la *k* ième fonction ($0 < k < 11$) immédiatement après la déclaration **Case *k*** correspondante et, surtout, ne pas modifier le reste de la fonction. Dans l'exemple ci-dessous, on peut voir (écrit en *italique*) qu'on a défini les fonctions 1 à 4. Notez en passant la quatrième fonction, dont la définition utilise la structure conditionnelle

If ... Then ... Else ... End If.

Puis nous entrons, comme précédemment, dans la procédure **Graphique**, les commandes que nous souhaitons voir exécuter. Notons ici que l'ordre des commandes est important : par exemple, on ne peut tracer correctement les axes tant que la région et la fenêtre utilisées n'est pas connue. De même, on commande le tracé de la fonction 4 après celui de la fonction 2 : en cas d'intersection, la couleur de la fonction 4 prévaudra donc. Remarquons que cette lourdeur de notation s'accompagne d'une certaine versatilité : si le cœur nous en dit, on peut tracer, sur un même graphique, la même fonction représentée dans des régions différentes (et donc possiblement avec des échelles différentes).

```
Function F (n, x)
   F = "ND"
   On Error GoTo Erreur
   Select Case n
      Case 1
         F = Cos(x)
      Case 2
         F = Sin(x)
      Case 3
         F = Sqr(x) ' Fonction racine carrée
      Case 4
         If Abs(x) < 1 Then
            F = x ^ 3
         Else:
            F = x
         End If
      Case 5

      Case 6

      Case 7

      Case 8

      Case 9

      Case 10

      Case Else

   End Select
Erreur:
End Function
```

```
Sub Graphique()
   RegionFonctions -2 * pi, 2 * pi, -5, 5
   ' FenetreFonctions -250, 250, -200, 200 déclarée implicitement
   TracerAxes ' Après les commandes RegionFonctions
              '  et FenetreFonctions (ici implicite)
   CouleurCrayon 255, 0, 0
   TracerFonction 1
   CouleurCrayon 0, 255, 0
   TracerFonction 2
   CouleurCrayon 0, 0, 255
   TailleCrayon 2
   TracerFonction 4
   PointsNonRelies
   PasFonctions 10
   TailleCrayon 3
   CouleurCrayon 255, 0, 255
   TracerFonction 3
End Sub
```



En guise de conclusion

Disons-le clairement : ***LangageGraphique*** n'est pas un environnement graphique qui est toujours optimal. Si vous avez la chance d'avoir accès à des logiciels tels *Cabri-géomètre*, *Maple*, *POV-Ray*, ou d'autres logiciels mathématiques avec possibilités graphiques, il arrivera souvent que vous pourrez obtenir *plus simplement* de merveilleux graphiques en choisissant l'outil le plus approprié.

Rappelons simplement les caractéristiques qui font la force de ***LangageGraphique*** et qui peuvent nous guider sur les circonstances où il est avantageux de l'employer.

1. Il est *gratuit*, si vous disposez déjà de *Microsoft Word.*
   En fait, il est placé dans le domaine public, code source compris. Ceci permet éventuellement de l'*adapter à ses besoins*, que ce soit en le modifiant à sa guise ou en lui ajoutant des possibilités qui ne s'y trouvent pas actuellement.
2. Il produit des *dessins vectoriels.*
   Rappelons que, contrairement aux dessins matriciels qui sont essentiellement composés d'une grille rectangulaire de points de couleurs, les dessins *vectoriels*

« retiennent » la description des objets dont ils sont composés. Ceci permet de les modifier de façon précise, que ce soit pour les redimensionner, leur appliquer une transformation, ou les rendre à une résolution différente (lors d'une impression, par exemple).
Notons aussi que, bien que plusieurs logiciels tels *Cabri-géomètre* et *Maple* produisent eux-aussi des dessins vectoriels, ceux-ci subissent parfois certaines « pertes » lors d'un copier-coller dans *Word*. Ce n'est pas le cas avec ***LangageGraphique*** puisqu'on demeure toujours dans *Word*.

3. ***LangageGraphique*** combine une approche à la fois ***gestuelle*** et ***textuelle*** dans la production de graphiques.
Ceci n'est pas fréquent. *Cabri-géomètre*, par exemple, est presque exclusivement gestuel (la calculatrice étant une exception notable). De son côté, *Maple* est presque exclusivement textuel (avec, comme exception intéressante, la possibilité de changer gestuellement de point de vue pour observer un graphique à trois dimensions).
LangageGraphique, de son côté, permet de produire *textuellement* des graphiques pour ensuite leur apporter *gestuellement* des changements qui peuvent être considérables (transformations, groupements et dissociations, alignements divers, ajouts d'éléments, etc.).
4. ***LangageGraphique*** permet de produire des graphiques via une *programmation.*
C'est aussi vrai de *Maple*, mais cette caractéristique manque cruellement dans *Cabri-géomètre*, qui ne permet ni les constructions conditionnelles générales, ni les constructions itérées automatiquement. Nous admettons cependant que c'est une possibilité assez intimidante pour plusieurs.

Terminons cette brève présentation par quelques graphiques produits via ***LangageGraphique***.