

Manuel utilisateur de DR. GEO



Hilaire Fernandes
OFSET – <http://www.ofset.org>
DSI-SEM – <http://www.ge.ch/dip/ecole>

4 décembre 2011



<http://www.ge.ch/sem/cc/by-nc-sa>

Licence : by-nc-sa

Ce document est publié par le DIP Genève sous licence Creative Commons - utilisation et adaptation autorisée sous conditions.

Auteur : Hilaire Fernandes

Relecteurs : François Audirac, Odile Benassy, Gérard Blanchet, Marc Leygnac, Yves Ouvrard, Thierry Pasquier, Jean Peyratout, Guy Veyssière.

\$Id: drgeo.tex 203 2011-11-17 15:21:53Z hfernandes \$

Table des matières

1	Introduction	7
1.1	Avant-propos	7
1.2	Présentation	8
1.3	DR. GEO sur le web	9
2	Fonctions de base	11
2.1	Outils de construction	11
2.1.1	Outils de point	12
2.1.2	Outils de ligne	13
2.1.3	Outils de transformation	14
2.1.4	Outils numériques et texte	15
2.1.5	Outil macro-construction	16
2.1.6	Autres outils	16
2.2	Autres fonctions	21
2.2.1	Déplacer la figure	21
2.2.2	Grossissement de la figure	21
2.2.3	Déplacer un objet	21
2.2.4	Afficher une grille	22
3	Fichiers et documents	23
3.1	Renommer une figure	23
3.2	Enregistrement d'une construction	23
3.3	Enregistrement d'une session	23
3.4	Enregistrer une macro-construction	24
3.5	Ouvrir un fichier	24
4	Fonctionnalités avancées	25
4.1	Macro-construction	25
4.1.1	Création d'une macro-construction	26
4.1.2	Jouer une macro-construction	28
4.2	Script Smalltalk DR. GEO	30
4.2.1	Script par l'exemple	30
4.2.2	Méthodes de référence pour les scripts DR. GEO	37
4.3	Figure Smalltalk de DR. GEO	41
4.3.1	Quelques exemples	41
4.3.2	Méthodes de référence pour les Figures Smalltalk DR. GEO	42
4.3.3	Galerie d'exemples	52
5	Applications didactiques	55

6	Astuces diverses	57
6.1	Programmation	57
6.1.1	Espace de travail	57
6.1.2	Débogueur	60
6.1.3	Inspecteur	60

Table des figures

1.1	La navigateur de code source de DR. GEO	7
1.2	Écran de bienvenue de DR. GEO	8
2.1	Catégories d'outils de DR. GEO et leurs descriptions	11
2.2	Menu et sous-menus pour éditer le style d'un objet point	17
2.3	Menu et sous-menus pour éditer le style d'un objet ligne	18
2.4	Menu et sous-menu pour éditer le style des valeurs	18
2.5	Éditer les coordonnées d'un point libre	18
2.6	Éditer les coordonnées d'un point libre sur une ligne	19
2.7	Éditer une valeur libre	19
2.8	Éditer un script	19
2.9	Éditer un texte	20
2.10	Appuyer sur [Shift] pour muter un point	21
3.1	La boîte de dialogue de session DR. GEO	24
4.1	La figure initiale	26
4.2	La figure avec la construction finale	26
4.3	Première page introductive de la boîte de dialogue de l'assistant pour construire une macro-construction	27
4.4	La seconde page, les trois points sont sélectionnés	27
4.5	La troisième page, le cercle et son centre sont sélectionnés	27
4.6	La quatrième page, le nom et la description de la macro-construction	27
4.7	L'utilisateur sélectionne les paramètres d'entrée dans la figure	28
4.8	Une figure avec trois points	28
4.9	La figure finale avec le cercle et son centre	29
4.10	L'éditeur de script	31
4.11	Mon premier script	32
4.12	Choisir un script	32
4.13	Courbe et tangente en un point	35
4.14	Triangle de Sierpinski	53
6.1	Votre espace de travail avec le code source collé et son menu contextuel	58
6.2	Résultat de l'exécution du code source : intégrale de la fonction sur $[-1; 1]$	58
6.3	Le profileur DR. GEO	59
6.4	Le débogueur DR. GEO	59
6.5	L'inspecteur sur la variable <code>sommets</code>	60

Chapitre 1

Introduction

1.1 Avant-propos

DR. GEO II est un logiciel libre¹ multiplate-formes de géométrie interactive. Il est une réécriture complète de DR. GEO 1.1 en Smalltalk. Pharo, implémentation libre de ce langage et environnement de développement performant – <http://pharo-project.org> –, a été utilisé pour ce faire. DR. GEO 1.1 était écrit en C++ et intégrait un interpréteur Scheme pour la rédaction de scripts et de figures programmées. DR. GEO II permet également l'intégration de scripts dans les figures géométriques ainsi que l'écriture de figures interactives entièrement décrites avec un langage de programmation.

Le choix d'une réécriture en Smalltalk fut motivé par les qualités dynamiques uniques de ce langage ; celui-ci nous permet en effet de pousser extrêmement loin nos investigations sur les dimensions interactives entre l'utilisateur et le logiciel. Ainsi DR. GEO n'est pas seulement un logiciel convivial de géométrie interactive mais aussi, tel que distribué, un environnement complet de programmation dans lequel le logiciel peut être étudié, modifié et amélioré. Pour s'en convaincre, l'utilisateur est invité à cliquer sur le fond de l'environnement DR. GEO – en dehors de toute fenêtre – puis à presser les touches `CTRL-B`. Le navigateur de classes affiché permet de modifier le code source de DR. GEO alors que celui-ci est en fonctionnement. Cet accès au code source du logiciel, pour l'étudier, le modifier et le redistribuer est complètement ancré dans l'esprit du logiciel libre pour une approche non verrouillée à une informatique autre que de béatitude. Loin de nous l'idée de prétendre que DR. GEO permet de rendre les esprits plus alertes, néanmoins il y contribue assurément.

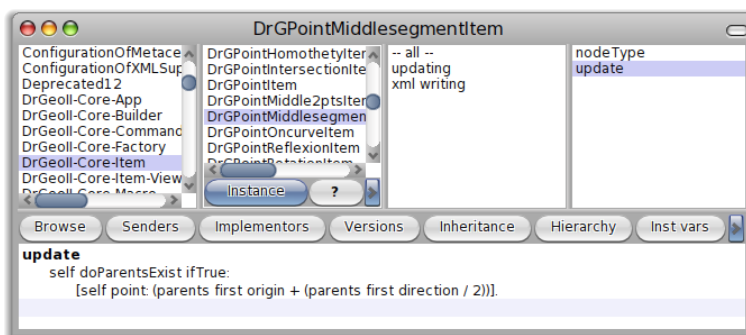


FIGURE 1.1 – La navigateur de code source de DR. GEO

Avec ce même esprit, les systèmes de figures programmées et de scripts – présentés dans

1. Un logiciel est libre lorsque son code source peut être étudié, modifié et redistribué.

les sections sur les outils avancés – sont adossés à un outillage évolué de mise au point du code : navigateur, débogueur, inspecteur d'objet.

Dans la suite du document, nous nommerons indifféremment le logiciel DR. GEO II ou DR. GEO.

1.2 Présentation

DR. GEO permet de créer des figures géométriques et de les manipuler interactivement en respectant leurs contraintes géométriques. Il offre également la possibilité d'introduire graduellement la programmation. Il est ainsi utilisable dans des situations d'enseignement allant du niveau primaire au niveau supérieur.

L'interface utilisateur de DR. GEO a été conçue pour allier dans un ensemble harmonieux à la fois simplicité d'utilisation, ergonomie et fonctionnalités avancées.

Ainsi l'interface de DR. GEO, sous une apparence de très grande simplicité, permet au néophyte de se familiariser très rapidement avec les fonctions de base du logiciel. Puis, au cours de sa progression, l'utilisateur découvrira des aspects plus avancés de l'interface et du fonctionnement de DR. GEO : multiplicité des modalités de construction d'objet², macro-construction, enregistrement multiple, scriptabilité, Figure Smalltalk de DR. GEO, héritage de Smalltalk dans DR. GEO. Ces fonctionnalités avancées génèrent peu de surcharge sur l'interface, c'est pour cela que DR. GEO est très agréablement utilisé en enseignement primaire, cependant il est également très intéressant pour le lycée.

Dans les sections suivantes, les outils de base seront exposés. Ensuite les fonctionnalités avancées seront présentées en détail.

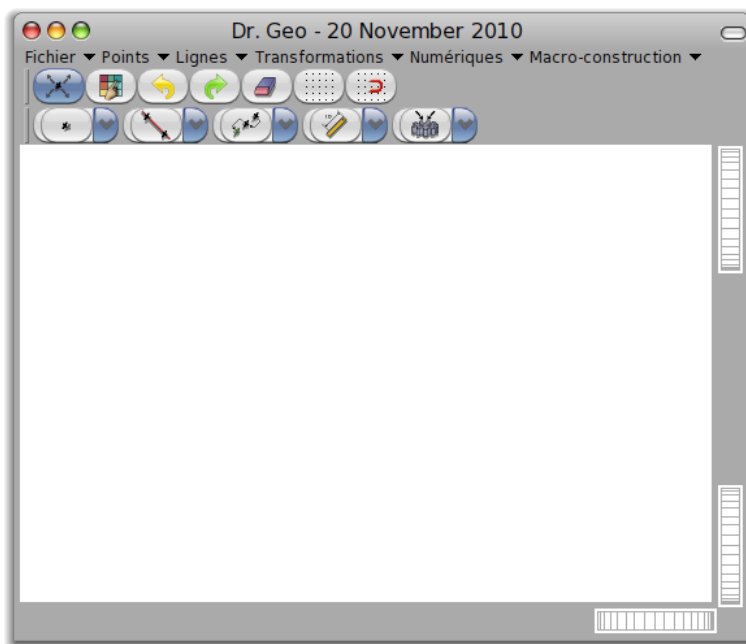


FIGURE 1.2 – Écran de bienvenue de DR. GEO

2. Il s'agit de pouvoir, à partir d'une même commande, créer un type d'objet selon des modalités différentes. Par exemple à partir de la commande construction de cercle, l'utilisateur peut créer un cercle défini par son centre et un point, ou bien une longueur, etc. Bien sûr cette commande n'est représentée que par une seule icône, il incombe à DR. GEO d'anticiper sur la construction de l'utilisateur. L'effet immédiat est donc une diminution de la charge cognitive de l'interface sur l'utilisateur, tout en proposant un nombre important de modes de construction.

L'agencement de l'interface est comme suit :

1. la *barre de menu* caractéristique avec **Fichier** , **Points** , **Lignes** , **Transformations** , **Numériques** et **Macro-construction** ;
2. une *barre d'outils d'édition* pour attraper un objet, modifier son style, défaire/refaire des actions d'édition, effacer un objet, afficher la grille et aimanter les éditions à celle-ci ;
3. la *barre d'outils* regroupant sous la forme d'icônes les fonctions de construction présentes dans la barre de menu ;
4. dans le coin droit en bas, l'utilisateur a à sa disposition deux molettes pour déplacer horizontalement et verticalement la figure ;
5. la seconde molette à droite permet de changer l'échelle de la figure.

Pour créer une nouvelle figure géométrique, l'utilisateur va dans le menu **Fichier**->**Nouveau** . Pour chaque nouvelle figure, une fenêtre distincte est proposée avec ses propres barres de menus et d'icônes.

1.3 Dr. Geo sur le web

DR. GEO dispose de son propre espace web sur le site d'OFSET à l'adresse : <http://www.offset.org/drgeo>.

Sur cet espace, l'utilisateur trouvera les informations suivantes :

- comment obtenir DR. GEO ;
- la documentation sur le logiciel ;
- des indications pour s'impliquer dans le projet DR. GEO ;
- des références sur des exploitations pédagogiques du logiciel.

Chapitre 2

Fonctions de base

Ce chapitre décrit les outils utilisés pour construire une figure géométrique.

2.1 Outils de construction

Ces outils sont partagés en six groupes accessibles à partir de la seconde barre d'icônes de DR. GEO.

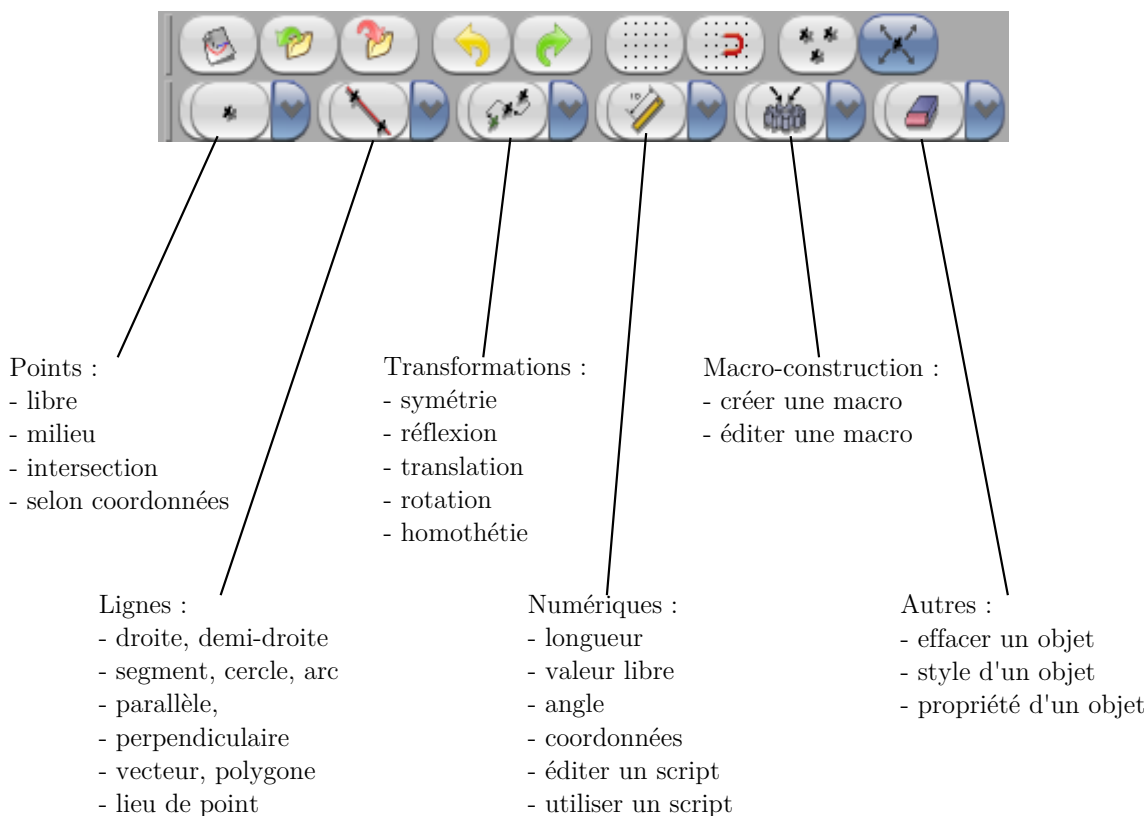


FIGURE 2.1 – Catégories d'outils de DR. GEO et leurs descriptions

Lorsque l'utilisateur clique sur une des flèches associées aux icônes de la barre d'outils, une barre verticale d'icônes s'affiche immédiatement. Celle-ci regroupe des fonctions d'une même famille. Lorsqu'un outil est choisi dans une barre verticale, celui-ci prend la place du

bouton alors visible dans la barre horizontale d'outils. L'utilisateur peut ainsi naviguer plus facilement entre plusieurs outils.

De la gauche vers la droite, nous avons accès aux barres d'icônes verticales pour construire des points et des lignes, utiliser des transformations, calculer des valeurs, gérer les macro-constructions.

Ces fonctions se retrouvent à l'identique dans le menu en haut de chaque fenêtre de DR. GEO.

2.1.1 Outils de point

Point libre



Crée un point libre dans le plan ou sur un objet unidimensionnel (segment, demi-droite, droite, arc de cercle, cercle, lieu) :

1. Dans le premier cas, le point créé peut être déplacé n'importe où dans le plan de la figure. Pour le construire l'utilisateur clique simplement sur le fond.
2. Dans le deuxième cas, le point n'est libre que dans l'objet unidimensionnel (ligne) où il a été créé ; il est collé sur l'objet. Pour construire ce type de point, l'utilisateur clique sur une ligne (c.-à-d. droite, demi-droite, segment, cercle, arc de cercle, etc.).

Pour créer un **point d'intersection** entre deux lignes (c.-à-d. droite, demi-droite, segment, arc de cercle, cercle), il suffit de cliquer à l'intersection de celles-ci, DR. GEO l'indique d'ailleurs par une bulle d'aide *Intersection*.

Comment placer un point avec des coordonnées données ? Une possibilité est de placer deux valeurs libres dans la figure – outils Numériques, section 2.1.4, p. 15 – puis de construire le point ayant pour coordonnées ces deux valeurs – outil Point défini par ses coordonnées, section 2.1.1, p. 12. Cette possibilité a un avantage sur la précédente, le point ainsi construit ne peut pas être déplacé directement à la souris, le point est en quelque sorte bloqué dans sa position.

Milieu



Crée le milieu de deux points ou d'un segment :

1. Dans le premier cas, l'utilisateur sélectionne deux points.
2. Dans le deuxième cas, l'utilisateur sélectionne simplement un segment.

Intersection



Crée la ou les intersection(s) de deux lignes (i.e. droite, demi-droite, segment, arc de cercle, cercle). L'utilisateur doit sélectionner deux lignes. Lorsqu'une des lignes choisie est un arc de cercle ou un cercle alors deux points d'intersection sont créés.

Point défini par ses coordonnées



Crée un point défini par ses coordonnées. L'utilisateur doit soit sélectionner un script retournant un couple de coordonnées – section 4.2.1, p. 35 – soit sélectionner deux nombres : le premier correspond à l'abscisse, le second à l'ordonnée.

Comment placer un point contraint par ses coordonnées ? Cette fonction est largement utilisée lorsque nous souhaitons par exemple construire le lieu d'un point. Cette construction suppose au préalable l'existence de deux valeurs – section 2.1.4, p. 15 – le point est ensuite construit en sélectionnant ces deux valeurs.

2.1.2 Outils de ligne

Droite



Crée une droite définie par deux points. L'utilisateur sélectionne deux points.

Droite parallèle



Crée une ligne parallèle à une direction et passant par un point. L'utilisateur sélectionne un point et une direction (c.-à-d. une droite, une demi-droite, un segment ou un vecteur).

Droite perpendiculaire



Crée une droite perpendiculaire à une direction et passant par un point. L'utilisateur sélectionne un point et une direction (c.-à-d. une droite, une demi-droite, un segment ou un vecteur).

Demi-droite



Crée une demi-droite définie par deux points. L'utilisateur sélectionne deux points, le premier est l'origine, le second appartient à la demi-droite.

Segment



Crée un segment donné par deux points.

Vecteur



Crée un vecteur donné par deux points. L'utilisateur sélectionne deux points, le premier est l'origine, le second est l'extrémité.

Une fois que le vecteur est créé, celui-ci peut être déplacé indépendamment des deux points. Ceci reste vrai pour un vecteur construit par une transformation – section 2.1.2, p. 14.

Cercle



Crée un cercle. L'utilisateur peut créer un cercle à partir de différentes sélections :

1. le centre et un point du cercle ;
2. le centre et un nombre (le rayon du cercle) ;

- le cercle et un segment dont la longueur est le rayon du cercle.

Arc de cercle



Crée un arc de cercle défini par trois points. Le premier est l'origine de l'arc, le troisième est l'extrémité, le second est un point de l'arc.

Lieu d'un point



Crée un lieu défini par deux points. L'utilisateur sélectionne deux points, l'un des deux est un point sur une ligne, l'autre est un point sous contraintes du premier (c.-à-d. quand l'un bouge, l'autre fait de même).

Polygone



Crée un polygone défini par n points. L'utilisateur sélectionne $n+1$ points limitant le polygone. Le premier et le dernier sont un seul et même point ce qui indique à DR. GEO que la sélection est terminée. L'objet polygone n'est pas un objet comme les autres lignes, il n'est pas possible de placer un point dessus ou de construire une intersection entre un polygone et une autre ligne. En revanche il est possible de construire l'image d'un polygone par une transformation géométrique.

2.1.3 Outils de transformation

Symétrie axiale



Crée l'image d'un objet par une symétrie axiale. L'utilisateur sélectionne l'objet à transformer et l'axe de symétrie (une droite). Quand l'utilisateur veut construire l'image d'une droite, la première droite sélectionnée est la droite à transformer.

Symétrie centrale



Crée l'image d'un objet par une symétrie centrale. L'utilisateur sélectionne l'objet à transformer et le centre de symétrie (un point). Quand l'utilisateur veut construire l'image d'un point, le premier point sélectionné est le point à transformer.

Translation



Crée l'image d'un objet par une translation. Quand l'utilisateur veut construire l'image d'un vecteur, le premier vecteur sélectionné est le vecteur à translater.

Rotation



Crée l'image d'un objet par une rotation. L'utilisateur sélectionne l'objet à transformer, le centre et l'angle de la rotation. Quand l'utilisateur veut créer l'image d'un point, le premier point sélectionné est le point à transformer.

L'angle de la rotation peut être défini de plusieurs façons :

- **valeur numérique** : l'angle est alors exprimé en radians (Exemples de valeurs numériques : valeur libre, distance entre deux points, longueur d'un segment, une coordonnée, une valeur retournée par un script Smalltalk DR. GEO, etc.) ;
- **la mesure d'un angle géométrique formé par trois points** : sa mesure est alors exprimée en degrés. Attention, dans ce cas la mesure appartient seulement à l'intervalle $[0; 180]$;
- **la mesure d'un angle orienté de deux vecteurs** : sa mesure est exprimée en degrés et couvre l'intervalle $]-180; 180]$.

Homothétie



Crée l'image d'un objet par une homothétie. L'utilisateur sélectionne l'objet à transformer, le centre et le facteur (c.-à-d. un nombre). Quand l'utilisateur veut créer l'image d'un point, le premier point sélectionné est le point à transformer.

2.1.4 Outils numériques et texte

Distance, longueur & nombre



Crée une valeur numérique. La valeur numérique, selon la sélection de l'utilisateur, peut être calculée ou bien saisie :

1. pour deux points c'est la distance entre ces deux points ;
2. pour un segment c'est la longueur de ce segment ;
3. pour un vecteur c'est la norme de ce vecteur ;
4. pour un cercle c'est le périmètre de ce cercle ;
5. pour un arc de cercle c'est la longueur de cet arc ;
6. pour une droite c'est la pente de cette droite ;
7. pour une droite et un point c'est la distance entre ce point et la droite ;
8. un **clic souris directement sur le fond de la figure** permet à l'utilisateur d'entrer une nouvelle valeur (c.-à-d. une valeur libre).

Cette dernière possibilité est très intéressante dans certaines situations. Elle permet de fixer une longueur, le rayon d'un cercle, la mesure d'angle (en radians) ou les coordonnées d'un point. La valeur est ensuite utilisée à partir des outils spécifiques de construction de cercle, de rotation et de point défini par ses coordonnées.

Angle



Calcule la mesure d'un angle défini par trois points ou deux vecteurs. Dans le premier cas, l'angle est considéré comme non orienté (c.-à-d. angle géométrique dont la mesure est dans l'intervalle $[0; 180]$). Dans le second cas, l'angle est orienté et sa mesure est dans l'intervalle $]-180; 180]$.

Coordonnées



Crée les coordonnées (abscisse et ordonnée) d'un point ou d'un vecteur.

Script Smalltalk Dr. Geo



Crée un script Smalltalk DR. GEO. Le script reçoit 0 ou n références d'objets en entrée. Il retourne une instance d'un objet dont une représentation sous la forme d'une chaîne de caractères est affichée dans la figure, à l'emplacement désigné par l'utilisateur. Un script peut être utilisé pour ses effets de bord ou pour sa valeur de retour. Les scripts Smalltalk DR. GEO sont couverts en détails dans le chapitre des fonctionnalités avancées 4 et exactement à la section script 4.2, p. 30.

Texte libre



Ajoute un bloc de texte dans la figure. Cliquer à l'emplacement souhaité, puis éditer directement le texte. Si nécessaire, cliquer à nouveau sur le texte pour afficher le curseur de saisie. Valider ensuite par la touche clavier .

Pour éditer à nouveau le texte, choisir l'outil de modification des propriétés des objets, section 2.1.6, p. 18.

2.1.5 Outil macro-construction

Créer une macro-construction



Extrait une séquence de construction d'une figure et la transforme en macro-construction.

Jouer une macro-construction



Joue (c.-à-d. "lance") une macro-construction. La macro-construction peut être nouvelle ou chargée depuis un fichier.

Les macro-constructions sont présentées en détail dans la section 4.1, p. 25.

2.1.6 Autres outils

Supprimer un objet




Un objet d'une figure peut être supprimé en activant ce menu. Éventuellement, l'utilisateur peut annuler la suppression à l'aide de la fonction d'annulation à partir de la barre d'icônes ou du menu de l'application. Par défaut le nombre d'annulations possibles est de 10 mais l'utilisateur peut ajuster cette valeur depuis la boîte de dialogue des préférences.

Changer l'aspect d'un objet



Chaque objet géométrique a ses propres attributs de style comme la couleur, l'épaisseur, l'étiquette, la taille et la forme. De plus, il est possible de cacher temporairement un objet sans le supprimer. Par exemple, il peut être utile de cacher des constructions intermédiaires sans les supprimer. Tous ces attributs sont ajustables depuis le menu contextuel qui s'affiche lorsque l'utilisateur clique sur un objet de la figure. Pour cela il faut d'abord se mettre dans

le mode d'édition de style en cliquant sur le bouton  de la barre d'outils.

Style de point. Le menu contextuel de style d'un point concerne tous les types de points. Il est possible d'ajuster la couleur, la forme, la taille, le nom et la visibilité. Les points libres dans le plan ou sur une ligne peuvent également être collés à leur emplacement, dans le menu une entrée supplémentaire permet de basculer entre ces deux états.

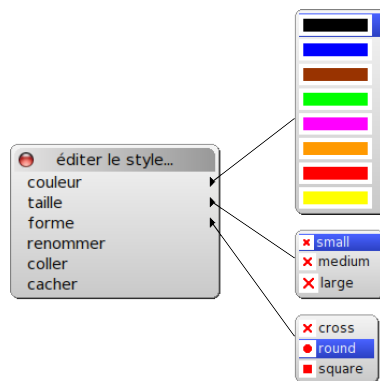


FIGURE 2.2 – Menu et sous-menus pour éditer le style d'un objet point

Style de ligne. Le menu contextuel de style d'une ligne concerne les droites, les demi-droites, les segments, les vecteurs, les cercles, les arcs de cercle, les lieux de points et les polygones. Il est possible d'ajuster la couleur, l'épaisseur, le style de trait, d'adjoindre des flèches pour les segments et d'éditer le nom et la visibilité.

Par ailleurs, pour les segments, deux sous-menus supplémentaires **flèche** et **marque** offre la possibilité d'adjoindre des flèches et des marques sur les segments.

Style de valeur. Le menu contextuel de style d'une valeur concerne toutes les sortes de valeurs : saisie par l'utilisateur, correspondant à une mesure ou calculée par un script Smalltalk DR. GEO. Il permet de modifier la couleur d'affichage, d'éditer la valeur – pour les valeurs librement saisies par l'utilisateur – d'éditer le nom et la visibilité de la valeur. Les valeurs peuvent également être verrouillées à leur emplacement.

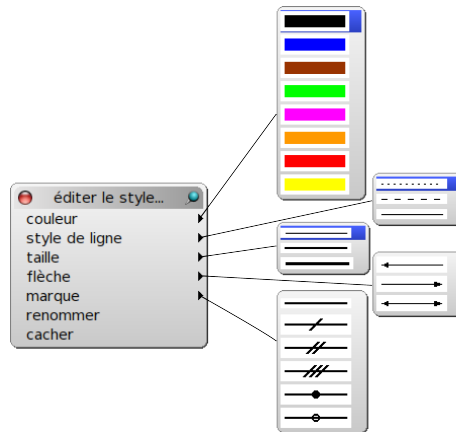


FIGURE 2.3 – Menu et sous-menus pour éditer le style d'un objet ligne



FIGURE 2.4 – Menu et sous-menu pour éditer le style des valeurs

Propriété d'un objet



Les propriétés de certains objets sont modifiables numériquement. Sont concernés les points libres dans le plan ou sur une courbe, les valeurs numériques libres et les scripts. Pour ce faire, après avoir sélectionné cet outil, choisir un de ces objets ; une boîte de dialogue s'affichera alors :

Point libre. En choisissant un point libre, une boîte de dialogue permet de modifier son abscisse et son ordonnée.

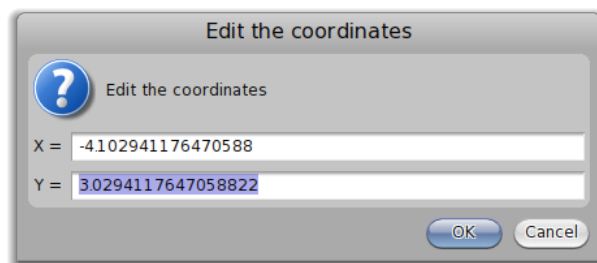


FIGURE 2.5 – Éditer les coordonnées d'un point libre

Point libre sur une courbe. En choisissant un point libre sur une courbe, une boîte de dialogue permet de modifier son abscisse curviligne. Cette dernière est normalisée sur

l'intervalle $[0; 1]$.

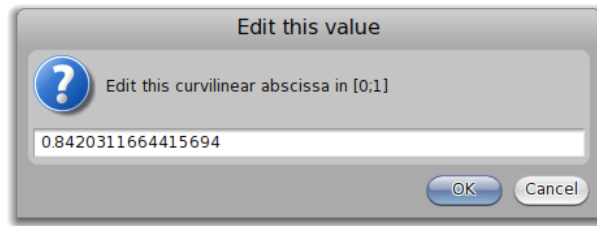


FIGURE 2.6 – Éditer les coordonnées d'un point libre sur une ligne

Valeur libre. En choisissant une valeur libre, une boîte de dialogue permet d'éditer sa valeur.

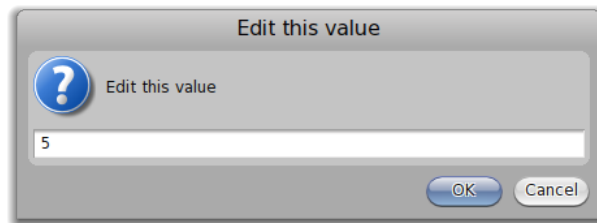


FIGURE 2.7 – Éditer une valeur libre

Script. En choisissant un script, un navigateur de code permet de l'étudier et de le modifier. Pour sauvegarder toute modification apportée au script, utiliser la combinaison de touche `CTRL-S` ou l'entrée *Do-it* dans le menu contextuel au-dessus du script (clic droit de la souris pour l'afficher).

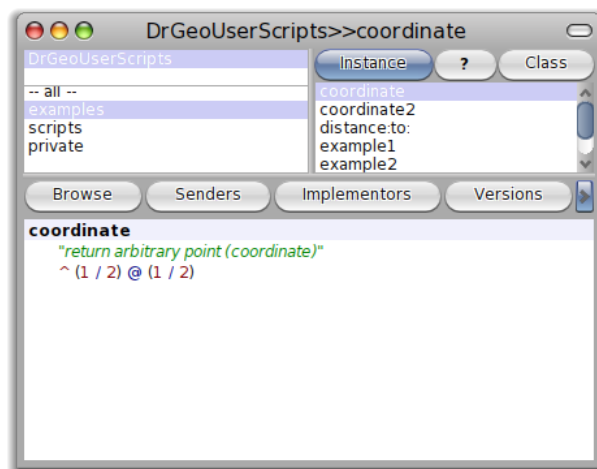


FIGURE 2.8 – Éditer un script

Texte. En choisissant un texte, une boîte de dialogue permet de l'éditer. Dans celle-ci, le texte peut être mise en forme sur plusieurs lignes à l'aide de retour chariot – touche clavier ENTRÉE.

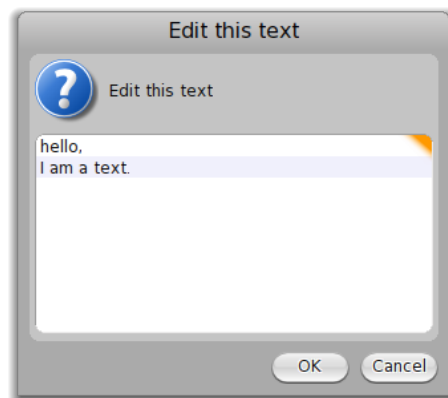


FIGURE 2.9 – Éditer un texte

2.2 Autres fonctions

2.2.1 Déplacer la figure

La figure peut être déplacée à l'aide des molettes dans le coin en bas à droite de la figure ou bien directement avec le bouton droit de la souris.

Dans ce mode, il est également possible de changer la nature d'un point d'une des natures suivantes :

- point libre dans le plan
- point libre sur une ligne
- point d'intersection

vers un point d'une des natures suivantes :

- point libre dans le plan
- point libre sur une ligne
- point d'intersection

Par exemple transformer un point libre dans le plan en un point d'intersection. Il existe toutefois une contrainte : il n'est pas possible de muter un point vers une ligne (libre ou intersection) plus "jeune" que le point. Plus jeune signifie que la ligne a été créée après le point.

La touche SHIFT enfoncée en même temps qu'un point est attrapé et déplacé indique toujours par des bulles d'information les points qui peuvent être mutés et les destinations (plan, ligne, intersection de lignes) possibles.

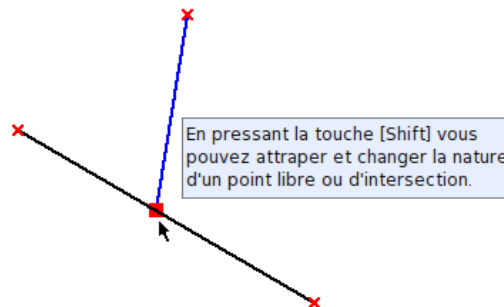


FIGURE 2.10 – Appuyer sur [Shift] pour muter un point

2.2.2 Grossissement de la figure


L'échelle de la figure est modifiable à l'aide de la molette en haut à droite de la fenêtre. La molette de la souris offre cette même fonction ; presser simultanément la touche SHIFT du clavier augmente la vitesse de grossissement.


2.2.3 Déplacer un objet



Un objet peut être déplacé par glisser-déposer. La figure est alors redessinée en respectant ses propriétés. Quasiment tous les objets géométriques peuvent être déplacés. Si nécessaire, DR. GEO déplace les points libres associés. Par exemple, quand l'utilisateur déplace une droite définie par deux points, DR. GEO déplace les deux points simultanément.

2.2.4 Afficher une grille

Il est possible d'afficher ou cacher une grille unitaire dans toute figure de DR. GEO, la commande est accessible depuis le bouton  de la boîte d'outils. La grille est unitaire, chaque subdivision représente une unité. Enfin, lors de la sauvegarde d'une figure, l'état de la grille est également sauvegardé (affichée ou non affichée).

Lors de la création ou du déplacement de points, il est possible de coller ceux-ci à la grille – ou au mieux selon la contrainte du point. Pour cela il suffit d'activer la fonction grille aimantée avec le bouton  de la boîte d'outils.

Chapitre 3

Fichiers et documents

Les constructions peuvent être enregistrées de deux manières. Une construction par fichier ou un ensemble de constructions par fichier (c.-à-d. une session DR. GEO). Nous vous rappelons que les documents contenant des figures doivent être sauvegardés avec une extension **.fgeo** et ceux contenant uniquement des macro-constructions avec l'extension **.mgeo**. Pour les documents contenant des figures et des macro-constructions, utiliser l'extension **.fgeo**.

Ceci n'est qu'indicatif mais suivre ces règles permet de s'y retrouver plus facilement dans ses fichiers.

3.1 Renommer une figure

Depuis le menu **Fichier->Renommer**, l'utilisateur change le nom et le titre de la fenêtre de la figure active. Cette fonction est particulièrement utile lorsque l'utilisateur sauvegarde un ensemble de figures dans un unique fichier de session.

3.2 Enregistrement d'une construction

À partir du menu **Fichier->Enregistrer**, la figure de la vue active est enregistrée dans un fichier.

(!) DR. GEO peut travailler avec plusieurs figures en même temps, chaque figure ayant sa propre fenêtre. L'utilisateur peut passer d'une figure à l'autre en utilisant la barre des tâches en bas de l'environnement DR. GEO.

Avec le menu **Fichier->Enregistrer sous...**, l'utilisateur peut changer le nom du document enregistré.

3.3 Enregistrement d'une session

Une session est un ensemble de données de DR. GEO que l'utilisateur enregistre d'un seul coup dans un fichier. Cela permet de placer un ensemble de données (figures et macro-constructions) dans un seul fichier, afin de faciliter leurs réutilisations.

À partir du menu **Fichier->Enregistrement multiple**, l'utilisateur accède à la boîte de dialogue de sauvegarde d'une session.

Dans cette boîte de dialogue, la liste de toutes les données actives est présentée dans une colonne. Chaque ligne est préfixée du tag **Fig1** ou **Macro** selon son type, la seconde partie de la ligne contient le nom de la donnée.

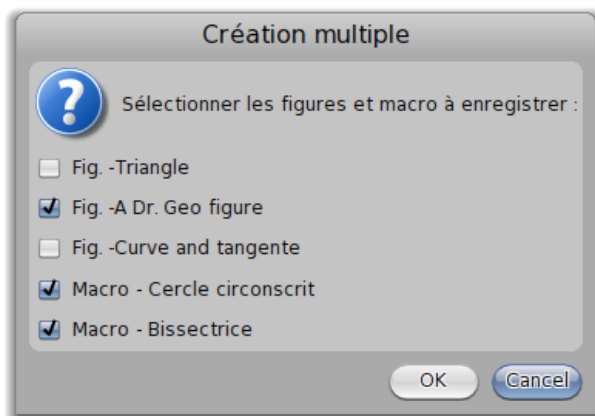


FIGURE 3.1 – La boîte de dialogue de session DR. GEO

(!) Actuellement, une session peut contenir deux types de données : figure interactive 2D et macro-construction.

L'utilisateur sélectionne les données à enregistrer en cochant leur case puis valide avec le bouton `OK`.

(!) Le menu `Fichier->Enregistrement multiple` est le seul moyen d'enregistrer une macro-construction dans un fichier.

3.4 Enregistrer une macro-construction

Pour enregistrer une ou des macro-constructions dans un fichier, procéder comme pour l'enregistrement d'une session – enregistrement multiple. Depuis la boîte de dialogue de sauvegarde d'une session, sélectionner la ou les macro-constructions à sauvegarder, puis sauver dans un fichier d'extension `.mgeo`. C'est tout !

Il est ainsi possible de constituer des bibliothèques de macro-constructions, une par fichier ou plusieurs regroupées selon un même thème dans un seul fichier.

3.5 Ouvrir un fichier

Que l'utilisateur ait sauvegardé une seule figure ou une session avec différents types de documents, la procédure pour l'ouverture est la même par le menu `Fichier->Ouvrir`. Si la session ouverte contient des macro-constructions, celles-ci sont directement disponibles depuis l'outil pour jouer des macro-constructions. Les macro-constructions sont alors disponibles depuis toutes les fenêtres ouvertes de DR. GEO, actuelles ou futures.

Chapitre 4

Fonctionnalités avancées

Dans ce chapitre, nous présentons les fonctionnalités qui permettent d'étendre les possibilités de DR. GEO ou de l'adapter à une situation pédagogique donnée.

La première est la *macro-construction*. Elle permet d'extraire une construction logique pour la placer dans un enregistrement. Cet enregistrement peut ensuite être répété autant de fois que souhaité, sauvegardé dans un fichier et ouvert ultérieurement dans une autre figure.

Les scripts Smalltalk DR. GEO représentent une autre fonctionnalité pour étendre DR. GEO. Ces scripts sont de véritables items de figure, comme les items géométriques. En entrée, ils reçoivent une ou plusieurs références d'items géométriques et ils retournent une valeur placée dans la figure. Ce sont en fait des fonctions¹ greffées dans une figure, elles sont évaluées à chaque mise à jour de la figure (c.-à-d. lorsque la figure a besoin d'être redessinée). Les scripts Smalltalk DR. GEO sont utiles pour la valeur qu'ils retournent ou leur effet de bord, cela dépend de ce que l'utilisateur souhaite réaliser.

En extension des scripts Smalltalk, DR. GEO propose d'aller encore plus loin avec les figures Smalltalk de DR. GEO. Cette fois il s'agit de décrire une figure géométrique complètement sous la forme d'un code source écrit dans le langage Smalltalk. La force de cette approche est de permettre une construction fonctionnelle² des figures et non plus simplement déclarative comme c'est le cas avec l'interface graphique.

4.1 Macro-construction

Une macro-construction ressemble un peu à une procédure qui reçoit des items d'une figure en entrée et qui retourne un ou plusieurs items de figure, construits par la macro-construction. Une macro est construite à partir d'un modèle défini par l'utilisateur. Cela signifie que l'utilisateur doit réaliser la séquence de construction une première fois dans une figure puis demander à DR. GEO de l'enregistrer dans une macro-construction. La macro-construction peut ensuite être sauvegardée dans un fichier au même titre qu'une figure.

Pour enregistrer une séquence de construction, DR. GEO doit connaître les items initiaux de la séquence ainsi que les items finaux. Évidemment, les items finaux ne doivent dépendre *que* des items initiaux³, sinon DR. GEO ne sera pas capable de déduire les items finaux à partir des items initiaux.

Ainsi, DR. GEO déduit la logique de la séquence de construction et l'enregistre dans une macro-construction. L'utilisateur peut alors répéter cette séquence en jouant la macro-construction, elle demande seulement des items initiaux (du même type) de la figure et construit les items résultants.

1. Ou procédures pour les amateurs de Pascal.

2. Par exemple sous forme récursive

3. Cette contrainte a depuis été assouplie et permet d'aller encore plus loin avec les macro-constructions.

(!) Les items de figure intermédiaires et invisibles sont aussi construits par la macro-construction. Ils sont nécessaires pour construire les items résultants.

Pour illustrer la fonctionnalité macro-construction, nous prenons l'exemple de la construction du cercle passant par trois points.

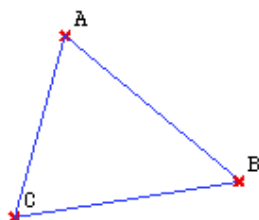


FIGURE 4.1 – La figure initiale

Avant la création de la macro-construction, l'utilisateur doit construire la figure finale, elle est considérée comme modèle par la macro-construction.

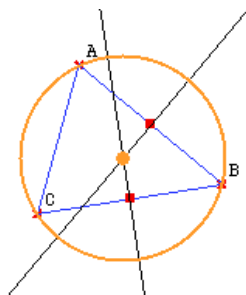



FIGURE 4.2 – La figure avec la construction finale

4.1.1 Création d'une macro-construction

À cette étape, la séquence de construction est faite. L'utilisateur doit maintenant avertir DR. GEO qu'il veut une macro-construction à partir de cette séquence. Il doit appeler la

fonction **Construire une macro** à partir de la barre d'icônes  ou depuis le menu **Macro-construction>Construire une macro** de la fenêtre de la figure.

Depuis la boîte de dialogue qui s'affiche alors, l'utilisateur sélectionne les paramètres d'entrée et de sortie, le nom et la description de la macro-construction.

La seconde page de la boîte de dialogue sert à sélectionner les paramètres d'entrée. Dans notre exemple, ce sont les trois points initiaux. L'utilisateur a juste besoin d'aller jusqu'à cette seconde page et il sélectionne les trois points A, B et C de la figure. Les points sélectionnés clignotent alors et leur nom s'affiche dans la page de la boîte de dialogue.

À partir de la troisième page, l'utilisateur choisit les paramètres de sortie. Dans notre exemple, nous voulons le cercle et son centre comme résultat de la macro-construction. L'utilisateur clique alors sur ces deux objets dans la figure, lorsque sélectionnés ces derniers clignotent et leur nom apparaît dans la boîte de dialogue.

Dans la quatrième page, l'utilisateur entre le nom et la description de la macro-construction. Ces informations sont affichées lorsque l'utilisateur exécute une macro-construction. Ceci permet de distinguer les macro-constructions entre elles.

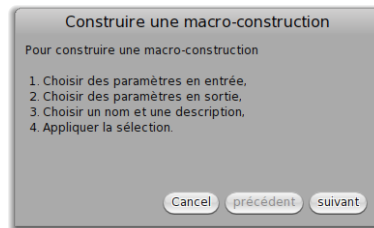


FIGURE 4.3 – Première page introductive de la boîte de dialogue de l'assistant pour construire une macro-construction

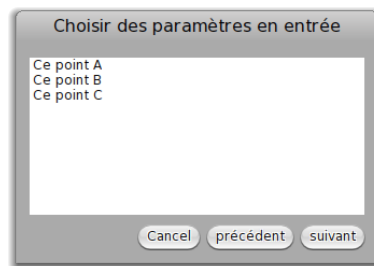


FIGURE 4.4 – La seconde page, les trois points sont sélectionnés

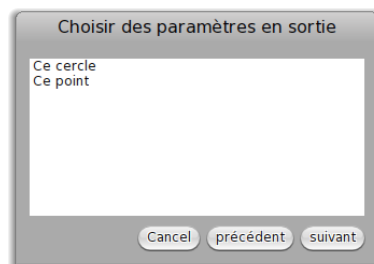


FIGURE 4.5 – La troisième page, le cercle et son centre sont sélectionnés

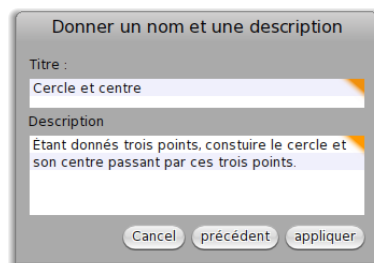


FIGURE 4.6 – La quatrième page, le nom et la description de la macro-construction


L'utilisateur valide ensuite la définition de la macro-construction en appuyant sur le bouton **Appliquer**. Il peut aussi revenir aux étapes précédentes pour ajuster les paramètres de la macro-construction.

-
- (!) Si la sélection des paramètres d'entrée et de sortie ne correspond pas (DR. GEO ne peut pas extraire la logique de la construction), la macro-construction ne peut pas être créée. Dans ce cas, l'utilisateur doit reconsidérer la sélection des paramètres d'entrée et de sortie. Il peut revenir à la seconde ou la troisième page de la boîte de dialogue de l'assistant pour ajuster ses choix.
-

À cette étape, la macro-construction est créée et enregistrée dans DR. GEO. Dans la prochaine section, nous verrons comment l'utiliser.

4.1.2 Jouer une macro-construction

À l'aide de la boîte de dialogue

Pour exécuter une macro-construction, l'utilisateur clique sur le bouton  de la barre d'outils ou choisit le menu `Macro-construction>Jouer une macro` de la fenêtre de la figure. Une boîte de dialogue décrivant la procédure s'affiche alors.

À partir de celle-ci, l'utilisateur sélectionne la macro-construction. Dans la seconde page, il sélectionne la macro-construction dans la liste en haut de la boîte de dialogue. Une fois la macro sélectionnée, il peut directement choisir les paramètres d'entrée dans la figure. Dès que tous les paramètres d'entrée sont sélectionnés, la macro-construction est exécutée et les paramètres finaux sont affichés.

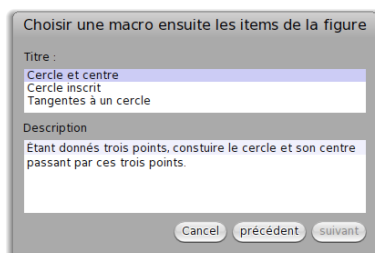


FIGURE 4.7 – L'utilisateur sélectionne les paramètres d'entrée dans la figure

Dans notre exemple, la macro-construction nécessite trois paramètres d'entrée (trois points) et elle construit un point et un cercle. Pour exécuter notre macro-construction, il faut une figure avec au moins trois points.



FIGURE 4.8 – Une figure avec trois points

Une fois que notre macro-construction est appliquée à ces trois points, nous avons le cercle souhaité et son centre.

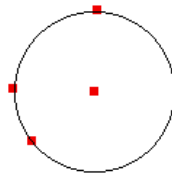


FIGURE 4.9 – La figure finale avec le cercle et son centre

4.2 Script Smalltalk Dr. Geo

DR. GEO est une application dynamique écrite en Smalltalk. Cela signifie qu'il est possible de modifier DR. GEO alors qu'il est en cours de fonctionnement. Nous avons exploité cette possibilité pour définir dans DR. GEO des items de figure qui sont en fait des scripts Smalltalk – des bouts de codes – pour étendre dynamiquement, à l'infini, les possibilités de DR. GEO. Mais qu'est-ce que Smalltalk ?

Smalltalk est un langage de programmation orienté objet, réflexif et dynamiquement typé. Il fut l'un des premiers langages de programmation à disposer d'un environnement de développement intégré complètement graphique. Il a été créé en 1972. Il est inspiré par Lisp et Simula. Il a été conçu par Alan Kay, Dan Ingals, Ted Kaehler, Adele Goldberg au Palo Alto Research Center de Xerox. Le langage a été formalisé en tant que Smalltalk-80 et est depuis utilisé par un grand nombre de personnes. Smalltalk est toujours activement développé.

Smalltalk a été d'une grande influence dans le développement de nombreux langages de programmation, dont : Objective-C, Actor, Java et Ruby.

Un grand nombre des innovations de l'ingénierie logicielle des années 1990 viennent de la communauté des programmeurs Smalltalk, tels que les Design Patterns (appliquées au logiciel), l'Extreme Programming (XP) et le refactoring. Ward Cunningham, l'inventeur du concept du Wiki, est également un programmeur Smalltalk.⁴

Cet extrait de la préface du livre *Pharo By Example*⁵ – <http://pharobyexample.org> – décrit précisément la plate forme Smalltalk utilisée pour DR. GEO :

Pharo est une implémentation moderne, libre et complète de l'environnement et langage de programmation Smalltalk.

Pharo s'attache à offrir une plate forme robuste et stable pour du développement professionnel en langages et environnement dynamiques.

DR. GEO exploite l'environnement Smalltalk pour proposer, d'une part, un environnement convivial d'écriture de scripts et, d'autre part, pour donner accès à l'interface des items géométriques ou numériques constitutifs d'une figure. L'interface est en fait l'ensemble des méthodes d'instance – fonctions de classe – de ces items.

Ainsi l'utilisateur peut écrire des scripts pour manipuler les items des figures ; et puisque ces scripts sont des items de figure au même titre que d'autres, ils n'ont pas besoin d'être dans un fichier séparé, ils sont enregistrés dans le fichier de la figure.

L'autre grande force des script est de s'appuyer sur l'environnement de développement de Pharo Smalltalk ; l'utilisateur bénéficie ainsi d'outils évolués pour mettre au point ses scripts : navigateur de classes, inspecteur, débogueur, etc. L'utilisateur souhaitant exploiter au mieux la puissance des scripts est donc invité à étudier le livre *Pharo By Example* – <http://pharobyexample.org>, il y apprendra le langage Smalltalk et son environnement.

4.2.1 Script par l'exemple

Les scripts ont deux facettes :

- l'écriture du script lui même ;
- l'utilisation du script dans la figure, un même script est utilisable plusieurs fois, avec des paramètres différents.

L'outil pour créer ou éditer un script est disponible depuis le menu Numériques>Éditer un script , la fonction est aussi disponible depuis la barre d'outils.

4. Article Smalltalk de Wikipédia en français (<http://fr.wikipedia.org/wiki/Smalltalk>). Page consultée le : 2 janvier 2011 15 :47 UTC. Contenu soumis à la licence CC-BY-SA 3.0 (<http://creativecommons.org/licenses/by-sa/3.0/deed.fr>).

5. Une traduction en français est en cours de finalisation.

L'outil pour utiliser un script est disponible depuis le menu **Numériques>Utiliser un script**, la fonction est aussi disponible depuis la barre d'outils.

Un script peut recevoir de 0 à n paramètres d'entrée. Après avoir choisi un script à insérer dans la figure, il suffit de choisir, dans la construction, les paramètres d'entrée puis de cliquer quelque part sur le fond de la figure pour y placer le résultat du script.

Dans la suite nous vous proposons de travailler sur quelques exemples de scripts, leurs fonctionnalités et leur puissance seront plus facilement analysées. Les scripts, comme les macro-constructions, donnent une dimension particulière à DR. GEO, ils permettent – chacun avec un positionnement différent⁶ – d'aller là où les auteurs du logiciel ne sont pas allés ou ne souhaitent pas aller.

Il est aussi important de comprendre que la plupart des fonctionnalités de Pharo Smalltalk sont disponibles depuis les scripts. C'est particulièrement vrai pour ses bibliothèques de fonctions⁷, nous allons bien sûr les utiliser intensément.

Script sans paramètre d'entrée

Premier exemple : La procédure pour créer un script sans paramètre d'entrée est la suivante :

1. Édition du script

- (a) Choisir **Numériques>Éditer un script** dans le menu. L'éditeur de script s'affiche alors :

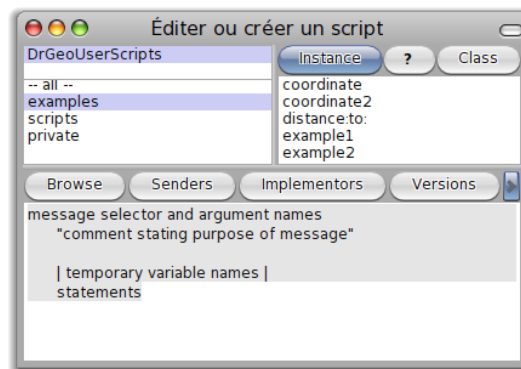


FIGURE 4.10 – L'éditeur de script

L'éditeur de script comprend trois parties :

- En haut à gauche les catégories de scripts, il s'agit simplement de les y ranger : **examples**, **private** et **scripts**. C'est dans cette dernière catégorie que l'utilisateur doit placer ses scripts.
 - En haut à droite les scripts de la catégorie sélectionnée sont présentés à l'utilisateur. En cliquant sur un de ceux-ci, son code source est affiché dans l'éditeur de texte en bas.
 - En bas sur toute la largeur, la zone texte avec le code source du script sélectionné. C'est ici que nous créons ou modifions les scripts. Pour valider une modification, il suffit de presser les touches **CTRL-S**.
- (b) Dans l'éditeur de script, cliquer sur la catégorie **scripts** jusqu'à ce que la zone texte en bas contienne le modèle de script tel qu'affiché dans la figure 4.10.

6. Les macro-constructions ont une approche géométrique tandis que les scripts ont une approche numérique mais aussi et surtout nous pouvons les utiliser dans un esprit de bidouillage ("hacking" en anglais).

7. En particulier, les fonctions mathématiques.

(c) Saisir le code source ci-dessous et comme dans la figure 4.11 :

```
MonPremierScript
"Le commentaire de mon premier script"
^ 'hello !'.
```

Sauvegarder le script par la combinaison de touches `CTRL-S`, DR. GEO vous demandera vos nom et prénom, vous pouvez entrer vos initiales, c'est sans grande importance pour le moment. La premier ligne du script, `MonPremierScript` désigne le nom du script, la suite est le code source du script. La première ligne de celui-ci, entre guillemets, est un commentaire de ce que fait le script, nous conseillons de bien le documenter car c'est utile pour la suite.

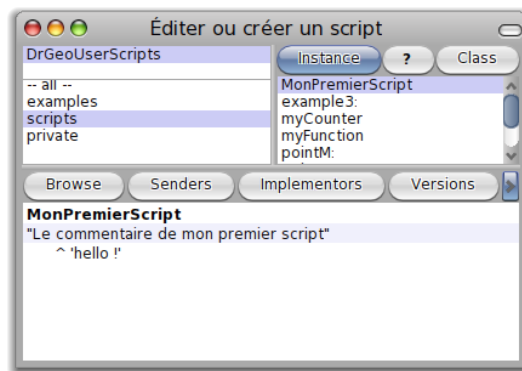


FIGURE 4.11 – Mon premier script

L'éditeur de script peut maintenant être fermé.

2. Utilisation du script dans la figure

Choisir `Numériques>Utiliser un script` dans le menu. Dans la boîte de dialogue qui s'affiche alors, choisir le script `MonPremierScript` que nous avons créé précédemment. Noter qu'à chaque fois qu'un script est choisi, son commentaire descriptif est affiché en bas de la boîte de dialogue.

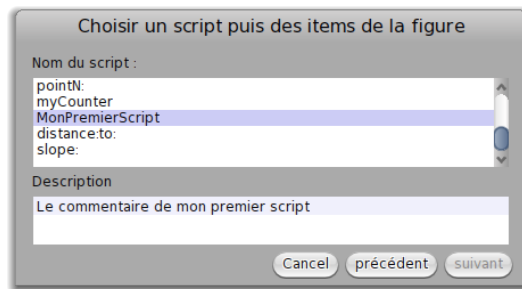


FIGURE 4.12 – Choisir un script

Une fois le script sélectionné, cliquer dans la figure à l'emplacement souhaité, le script y sera inséré; dans cet exemple le script ne fait que retourner le message `<< hello! >>`. La valeur retournée par un script est celle affichée dans la figure.

Dans les exemples qui suivent, nous donnons simplement le code source des scripts Smalltalk, il suffira de reprendre les étapes expliquées précédemment pour créer le script correspondant et l'utiliser dans une figure, nous ne revenons pas sur ces étapes.

Un générateur de nombres aléatoires et autres : Si vous souhaitez un générateur de nombres entiers aléatoires entre 0 et 10, rien de plus simple, c'est ce que fait le script suivant :

```
aléatoire
"Je retourne un nombre continuellement modifié aléatoirement"
  ^ 10 atRandom.
```

À chaque mise à jour de la figure, il génère un nombre aléatoire entier dans l'intervalle $[0; 10]$.

Si vous préférez un nombre flottant dans l'intervalle $[0; 1]$, utilisez ce script :

```
aléatoire2
"Je retourne un nombre décimal aléatoire entre 0 et 1"
  ^ 100 atRandom / 100.
```

-
- (!) Quelques précisions :
- La valeur retournée par le script est le résultat de l'expression après le symbole `^` ;
 - La valeur retournée peut être de n'importe quel type⁸, DR. GEO en affiche une représentation sous forme de chaîne de caractères ;
 - Si l'on souhaite retourner la valeur d'une variable, il suffit de mettre son nom après le symbole `^` .
-

Calculer des valeurs usuelles : Pour calculer une valeur approchée de π :

```
pi
  ^ Float pi
```

ou de e :

```
e
  ^ Float e
```

Les valeurs retournées par ces scripts sont ensuite utilisables comme toutes les autres valeurs numériques que peut générer DR. GEO. Pour toutes ces petites choses les scripts sont donc vos amis. Mais ils peuvent faire bien plus de choses intéressantes lorsqu'ils reçoivent des paramètres en entrée.

En effet ici les scripts n'avaient aucun argument, il n'y avait donc pas lieu de sélectionner des items de la figure lors de l'insertion des scripts dans la construction. Bien sûr leur intérêt réside dans les traitements numériques qu'ils permettent sur des items et la restitution de ce résultat dans la figure, sous la forme d'un objet qui lui même peut-être utilisé par d'autres scripts. Dans les sections suivantes nous montrerons de tels enchaînements de scripts.

Script avec un paramètre d'entrée

La procédure pour créer un script avec un paramètre d'entrée est sensiblement la même :

1. Lors de l'édition du script (`Numériques>Éditer un script`), l'argument s'intercale dans le nom du script.

Par exemple pour calculer la distance à l'origine d'un point, nous écrivons le script suivant :

```
distanceToOrigin: item
"Retourne la distance à l'origine d'un point"
  ^ item point dist: 0@0
```

Quelques explications :

- `item` est l’argument de notre script, cela doit être un point. C’est en fait une instance de la classe `DrGPointItem`⁹.
 - `DrGPointItem` a une méthode d’instance `#point` qui retourne ses coordonnées. Ainsi de l’argument nous pouvons extraire des informations, ici ses coordonnées et faire un calcul à partir de celles-ci.
 - `0@0` est une instance de la classe `Point` de coordonnées (0,0).
 - `#dist:` est un message à mot clé¹⁰ de la classe `Point` qui attend comme unique argument une instance d’un autre point, elle calcule la distance entre ces deux instances. Elle peut se comprendre comme : “distance entre `item` point et (0,0)”. Les messages à mot clé sont une spécificité de Smalltalk : les arguments sont intercalés dans le nom du message.
2. Lors de l’utilisation du script (`Numériques>Utiliser un script`), DR. GEO attend que l’utilisateur clique sur un point, puis sur un emplacement de la figure.
Attention, si un autre objet qu’un point est choisi, DR. GEO génère une erreur, il suffit de fermer la fenêtre du débogueur et de sélectionner à nouveau le script pour recommencer.

Selon le type d’objet en référence, diverses méthodes sont disponibles ; qui pour obtenir sa valeur, qui pour obtenir ses coordonnées, etc. Le répertoire des méthodes est disponible depuis la section Méthodes de référence des scripts 4.2.2, p. 37.

Script avec deux paramètres d’entrée :

Supposons que nous souhaitions calculer la distance entre deux points, nous créons alors un script avec deux paramètres d’entrée, ils sont intercalés dans le nom du script. Celui-ci peut s’appeler `distance:to:;` chaque “ : ” désigne l’emplacement d’un paramètre. Ainsi dans l’éditeur de script nous écrivons le code source suivant :

```
distance: item1 to: item2
"Calcule la distance entre deux points"
  ^ item1 point dist: item2 point
```

`item1` et `item2` sont les deux noms de nos paramètres, nous sommes libres de leur nommage. Pour utiliser ce script, procéder comment dans les exemples précédents : choisir deux points de la construction et un emplacement de la figure pour y placer le résultat du script.

Exemple détaillé de figure avec plusieurs scripts

Dans la section suivante, nous présentons une figure plus complexe intégrant un enchaînement de scripts pour la construction d’une portion de courbe représentative d’une fonction et la tangente en un point mobile de cette portion de courbe.

La figure finale est disponible dans le dossier `exemples` de DR. GEO, elle s’appelle `Curve and slope.fgeo`.

Dans une nouvelle figure, nous commençons par construire un segment horizontal, nous y plaçons un point libre appelé “Move me!”. Ce point servira de base à la construction de la courbe comme lieu d’un point.

Définir un fonction : Comme un script est capable de retourner n’importe quel type d’objet, le premier de notre construction définira simplement la fonction utilisée. Pour ce faire nous utilisons des objets Smalltalk de type bloc de code – fonction anonyme en Lisp. Nous nommons ce script `myFunction`, sans paramètre :

9. Pour découvrir le protocole de cette classe, écrire son nom n’importe où dans l’environnement DR. GEO, le sélectionner à la souris puis presser les touches `CTRL-B`, un navigateur de classes s’affiche alors sur cette classe, il permet de naviguer et d’étudier son code source.

10. Comme le script ici son nom se termine par “ : ”

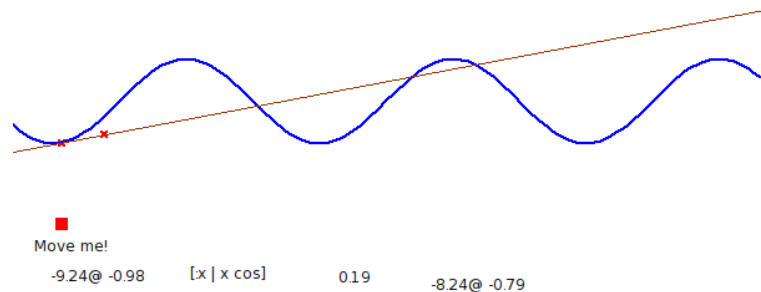


FIGURE 4.13 – Courbe et tangente en un point

```
myFunction
"La définition de notre fonction"
^ [:x | x cos]
```

Ensuite nous le plaçons dans la figure¹¹. Ainsi le bloc de code retourné par `myFunction` attend un argument `:x` et retourne le cosinus de celui-ci. Nous verrons dans la suite comment manipuler ce script.

Image d’une valeur par une fonction : Maintenant nous calculons les coordonnées d’un point appartenant à la courbe. Nous utilisons notre point “Move me!” et notre fonction. Le script aura comme unique argument le point; comme nous allons le voir nous n’avons pas besoin du script `myFunction` en argument :


```
pointM: item
"Given a point calculate the y value from myFunction and the x valule of the point"
^ item point x @(self myFunction value: item point x)
```


Le point retourné par ce script a la même abscisse que le point de départ, son ordonnée est l’image par la fonction.

Noter :

- l’accès direct au script de la fonction : `self myFunction`;
- le passage de l’argument à la fonction doit se comprendre comme `myFunction(item point x)`.

Maintenant utilisons ce script `pointM` avec comme argument le point “Move me!”; le résultat du script est de la forme `1.2@0.5`, cela représente un couple de coordonnées.

Avec l’outil point , créons un point ayant ses coordonnées contraintes par le résultat de ce script.

La fonction lieu d’un point  donne la courbe en sélectionnant nos deux points.

Pente en un point d’une fonction : Pour ce faire, nous calculons une valeur approchée de la pente :

¹¹. Il est important de le référencer dans la figure afin qu’il soit inclus dans la description de celle-ci lors d’une opération de sauvegarde sur fichier.

$$p = \frac{f(x + 0.001) - f(x)}{0.001}$$

Cela se traduit par un script `slope:` avec comment argument le point où calculer une approximation de la pente :

```
slope: item
"Compute the slope at the given x point position for myFunction"
| f x |
  f := self myFunction.
  x := item point x.
  ^ (f value: x + 0.001) - (f value: x) / 0.001
```

Nous plaçons ensuite ce script dans la figure.

Noter :

- La déclaration de variables temporaires `| f x |`. Comme déjà expliqué les variables ne sont pas typées, pas de soucis de ce côté là.
- La référence du bloc de code dans une variable `f := self myFunction`. Le symbole pour assigner une valeur à une variable est “:=”.
- Les parenthèses! Smalltalk ne connaît pas la priorité des opérateurs, en fait ils n’existent pas dans ce langage. Le lecteur est invité à étudier la section sur les messages Smalltalk du livre *Pharo By Example*.

Calculer et afficher la tangente à une courbe : Pour construire notre tangente nous avons besoin de deux points : un sur la courbe – nous l’avons déjà – et un deuxième. Pour ce dernier nous utiliserons le calcul précédent de notre pente. Écrivons alors un script retournant les coordonnées d’un point sur la tangente :

```
pointN: item
"Given an initial point, calculate the coordinates of a point on the tangente"
| p x |
  p := self slope: item.
  x := item point x + 1.
  ^ x @ (item point y + p)
```

En utilisant le protocole de la classe `Point`, ce script s’écrit aussi en une ligne :

```
pointN: item
"Given an initial point, calculate the coordinates of a point on the tangente"
  ^ item point + (1@ (self slope: item))
```

Utilisons ce script avec comme argument le point de notre courbe. Nous obtenons un deuxième couple de coordonnées. Avec celles-ci construisons un point, la tangente est la droite définie par ce point et celui de la courbe.

En déplaçant le point “Move me!”, la tangente est recalculée. Tout aussi intéressant : modifier le script `myFunction` actualise correctement l’ensemble de notre construction. Quelques exemples de modifications :

```
^ [:x | x * x / 10]
^ [:x | x cos + (10 * x) sin]
^ [:x | (x * 5) cos + x abs]
```

4.2.2 Méthodes de référence pour les scripts Dr. Geo

Un argument passé à un script est toujours une référence vers une instance de classe de la hiérarchie `DrGMathItem`, elle est le modèle de base pour représenter tout objet d'une figure. Ainsi pour connaître les messages compris par un argument d'un script, il convient d'examiner la hiérarchie de `DrGMathItem`. Celle-ci comprend plus de 80 classes, mais du fait des héritages, seules quelques unes sont intéressantes pour les usages courants des scripts :

- `DrGMathItem`
- `DrGpointItem`
- `DrGDirectionItem` concerne segment, droite, demi-droite, vecteur
- `DrGArcItem`
- `DrGCircleItem`
- `DrGLocus2ptsItem`
- `DrGPolygonNptsItem`
- `DrGValueItem`

Pour explorer ces classes, ouvrir un espace de travail – `CTRL-K` après un clic sur le fond de l'environnement – y saisir et sélectionner le nom de la classe puis ouvrir le navigateur de classe par `CTRL-B`.

Les sections suivantes contiennent la description de quelques messages pouvant être utiles. Elles sont présentées par classe.

Item Math

C'est le protocole de la classe `DrGMathItem`, il concerne tout argument, c'est à dire que les messages exposés ci-dessous s'appliquent à tous les types d'objets passés en argument à un script.

```
<string> item safeName
```

`item` : item mathématique

Retourne : une chaîne de caractères représentant le nom de l'item

Exemple :

```
nom := point1 safeName.
```

```
^ nom asUppercase.
```

```
<boolean> item exist
```

`item` : item mathématique

Retourne : un booléen indiquant si l'item est dans un état permettant son existence

Exemple :

```
line exist ifTrue: [ position := line origin ].
```

```
<collection> item parents
```

`item` : item mathématique

Retourne : une collection des items parents de item

Exemple :

```
point1 := segment parents first.
```

```
item move: vector
```

Déplace item dans une direction donnée, tout en tenant compte de ses contraintes.

`item` : item mathématique

`vector` : un vecteur (x,y) représentant le déplacement

Exemple :

```
circle move: 2@1.
```

```
<point> item closestPointTo: aPoint
```

item : item mathématique

aPoint : un couple de coordonnées

Retourne : un couple de coordonnées du point de “item” le plus proche de “aPoint”.

Exemple :

```
position := item closestPointTo: 2@1.
```

Point

```
<point> item point
```

item : référence d’un point

Retourne : les coordonnées de ce point

Exemple :

```
abscissa := pointItem point x.
```

```
item point: aPoint
```

item : référence d’un point

aPoint : couple de coordonnées

Action : modifie les coordonnées de “item”

Exemple :

```
pointItem point: 5@2.
```

```
<float> item abscissa
```

item : référence d’un point libre sur une ligne

Retourne : abscisse curviligne de ce point, elle appartient à l’intervalle [0; 1]

Exemple :

```
a := pointItem abscissa.
```

```
item abscissa: a
```

item : référence d’un point libre sur une ligne

a : nombre décimal compris dans [0; 1]

Action : modifie l’abscisse curviligne d’un point libre sur une ligne

Exemple :

```
pointItem abscissa: 0.5.
```

```
point moveAt: aPoint
```

point : un item représentant un point géométrique

aPoint : un couple de coordonnées (x,y) où déplacer “point”

Action : déplace “point” à la position donnée

Exemple :

```
point moveAt: 2@1.
```

Ligne droite ou courbe

```
<float> curve abscissaOf: aPoint
```

curve : une ligne droite ou non

aPoint : un point (x,y)

Retourne : un nombre dans $[0; 1]$ abscisse curviligne de “aPoint” sur “curve”

Exemple :

```
a := curve abscissaOf: 2@1.
```

<point> curve pointAt: anAbscissa

curve : une ligne droite ou non

anAbscissa : un nombre dans $[0; 1]$

Retourne : un point sur “curve” d’abscisse curviligne “anAbscissa”

Exemple :

```
myPoint := curve pointAt: 0.5.
```

<boolean> curve contains: aPoint

curve : une ligne droite ou non

aPoint : un point (x, y)

Retourne : un booléen indiquant si “aPoint” est sur “curve”

Exemple :

```
(curve contains: 0@1) ifTrue: [^ 'Yes!'].
```

Droite, Demi-droite, Segment, Vecteur

<point> item origin

item : référence vers un objet de type droite, demi-droite, segment ou vecteur

Retourne : un point origine de cet item

Exemple :

```
item origin.
```

<vector> item direction

item : référence vers un objet de type droite, demi-droite, segment ou vecteur

Retourne : un vecteur (x, y) indiquant la direction

Exemple :

```
v := item direction.
```

```
slope := v y / v x.
```

<vector> item normal

item : référence vers un objet de type droite, demi-droite, segment ou vecteur

Retourne : un vecteur unitaire normal à la direction de “item”

Exemple :

```
n := item normal.
```

Cercle, Arc de cercle, polygone

<point> item center

item : référence vers un cercle ou un arc de cercle

Retourne : point contenant les coordonnées du centre du cercle ou de l’arc de cercle

Exemple :

```
c := item center.
```

`<float> item radius`

item : référence vers un cercle ou un arc de cercle

Retourne : rayon du cercle ou de l'arc de cercle

Exemple :

`r := item radius`

`<float> item length`

item : référence vers un arc de cercle ou un polygone

Retourne : longueur de l'arc de cercle ou du polygone

Exemple :

`l := arc length`

Valeur

`<float> item valueItem`

item : référence vers un item de type valeur

Retourne : valeur de cet item

Exemple :

`n1 := item2 valueItem.`

`n2 := item2 valueItem.`

`n1 + n2.`

`item valueItem: v`

item : référence vers un item de type valeur libre

v : valeur décimale

Action : modifie la valeur d'un item de type valeur libre

Exemple :

`item valueItem: 5.2.`

`item position: aPoint`

item : référence vers un item de type valeur

aPoint : point (x, y)

Action : déplace à l'écran la position d'un item valeur

Exemple :

`item position: 0.502.`

Angle

`<integer> angle degreeAngle`

angle : référence vers un angle, orienté ou géométrique

Retourne : une mesure de cet angle en degrés

Exemple :

`angle1 := a1 degreeAngle.`

`<float> angle radianAngle`

angle : référence vers un angle, orienté ou géométrique

Retourne : une mesure de cet angle en radian

Exemple :

`angle1 := a1 radianAngle.`

4.3 Figure Smalltalk de Dr. Geo

Les *Figures Smalltalk de DR. GEO* – (FSD) – sont des figures écrites en langage Smalltalk. Il ne s’agit donc plus de construire une figure à l’aide de l’interface graphique de DR. GEO mais plutôt de décrire une figure dans le langage Smalltalk. Nous avons apporté le plus grand soin afin que la syntaxe utilisée soit facile et légère.

4.3.1 Quelques exemples

En lui-même Smalltalk est un langage de très haut niveau. Lorsqu’une figure est définie dans ce langage, nous disposons également de toute sa puissance pour par exemple définir récursivement telle partie de la figure, ou bien pour placer aléatoirement certains objets de telle sorte qu’à chaque ouverture de la figure, celle-ci est légèrement différente. Bref, les FSD sont libérées du carcan de l’interface graphique tout en étant renforcées par le langage Smalltalk.

Une FSD est un code source Smalltalk à exécuter dans un espace de travail – *workspace*. C’est une fenêtre texte depuis laquelle du code Smalltalk est écrit et exécuté. Pour ouvrir un tel espace, faire `CTRL-K` lorsque aucune fenêtre n’est sélectionnée. On peut aussi y coller du texte par `CTRL-V`.

Nous allons étudier plusieurs exemples, chacun d’eux sera écrit dans un workspace et exécuté en sélectionnant le code puis la séquence de touches `CTRL-D` pour *Do-it!*

Commençons par étudier un exemple simple de FSD :

```
DrGeoCanvas new
```

C’est la plus petite FSD que nous puissions définir. Lors de son exécution, celle-ci va simplement créer une nouvelle figure vide. Le canevas DR. GEO affiché est simplifié puisqu’il ne comporte ni barre de menu, ni barre d’outils. En revanche un menu contextuel s’affiche par un clic bouton droit ou gauche sur le fond du canevas ; il donne accès aux outils pour construire et éditer la figure. du canevas.

Pour déplacer ce canevas, l’attraper sur les bords en bas ou à droite ; pour le fermer, presser le bouton du milieu de la souris et utiliser l’icone fermer en haut à gauche.

Abordons un deuxième exemple :

```
| c item |
c := DrGeoCanvas new.
item := c point: 1.2 @ -2.
item name: 'A'.
```

Cette FSD définit une figure avec un point libre *A* de coordonnées initiales (1,2 ; -2). Les objets sont ajoutés à la figure à partir du canevas, ici `point:` crée un point libre à partir de deux coordonnées. Le résultat est un objet `point`, qu’il est possible de modifier, ici il est renommé `'A'`.

Poursuivons avec un troisième exemple :

```
| c triangle hasard m n p |

triangle := [:p1 :p2 :p3 |
c segment: p1 to: p2.
c segment: p2 to: p3.
c segment: p3 to: p1].

hasard := [5 - 10 atRandom].

c := DrGeoCanvas new.
```

```

m := c point: hasard value @ 0.
n := c point: 5 @ 0.
p := c point:  hasard value @ 3.
triangle value: m value: n value: p.

```

Cet exemple est particulièrement intéressant, il nous montre trois choses importantes :

1. L'introduction d'une construction de plus haut niveau, non prévue au départ par DR. GEO. Ici nous avons défini le bloc de code `triangle` qui, à partir de trois points, construit le triangle passant par ces trois points. Nous pouvons comparer ceci avec les macro-constructions mais avec un degré de liberté beaucoup plus important.
2. La définition d'un bloc de code associé, ici nous avons défini `hasard` qui retourne un nombre entier compris entre -5 et 5. Nous utilisons ce bloc pour placer au hasard certains points de notre figure, ainsi à chaque exécution la figure est légèrement différente.
3. L'affectation du résultat d'une construction à une variable n'est pas obligatoire, nous l'utilisons lorsque nous souhaitons garder une référence de l'objet créé. Par exemple dans le bloc de code `triangle`, nous ne gardons pas de référence des segments créés, en revanche lorsque nous définissons nos trois points nous avons besoin de garder une référence dans des variables temporaires. Ainsi, lors de l'utilisation du bloc de code `triangle`, nous passons en paramètre les variables `m`, `n` et `p`.

Pour clore cette section, voici un dernier exemple :

```

| c a b d |

c := DrGeoCanvas new.
a := c point: 1@0.
b := c point: 5@0.
d := c line: a to: b.
a color: Color yellow;
  round;
  large.
b hide.
d dashed.

```

Deux points et une droite sont créés. Ensuite des commandes sont utilisées pour modifier l'aspect des objets, voire pour en cacher.

Nous avons terminé notre petite visite guidée des *Figures Smalltalk Dr. Geo*. Dans les sections suivantes nous exposons l'ensemble des commandes disponibles pour définir des FSD.

4.3.2 Méthodes de référence pour les Figures Smalltalk Dr. Geo

La définition d'objets dans un document FSD se fait dans un canevas DR. GEO. Les objets ainsi créés sont modifiables comme nous le verrons par la suite.

Cependant, avant toute définition d'objets d'une figure, cette dernière doit être créée avec la commande `DrGeoCanvas new`.

Commandes générales

```
<canvas> DrGeoCanvas new
```

Retourne : Retourne une référence vers un canevas et affiche celui-ci. Cette référence est nécessaire pour créer des objets dans ce canevas, il est donc important de la placer dans une variable.

Exemple :

```
| canvas |
canvas := DrGeoCanvas new.
```

```
canvas update
```

Action : Mise à jour du canevas après modification d'attributs de quelques items. La plupart du temps ce n'est pas nécessaire.

Point.

```
<point> canvas point: aPoint
```

aPoint : un couple de coordonnées (x,y)

Retourne : référence d'un point libre du plan de coordonnées initiales aPoint

Exemple :

```
canvas point: 5@2.
```

```
<point> canvas pointX: v1 Y: v2
```

v1 : un objet valeur

v2 : une objet valeur

Retourne : référence d'un point contraint par ses coordonnées

Exemple :

```
canvas pointX: (canvas freeValue: 2) hide Y: (canvas freeValue: 5) hide.
```

```
<point> canvas pointOnCurve: curve at: abscissa)
```

curve : référence d'une ligne (droite, demi-droite, segment, etc.)

abscissa : abscisse curviligne du point libre, la valeur appartient à l'intervalle [0; 1]

Retourne : référence d'un point libre sur une ligne

Exemple :

```
myPoint := canvas pointOnCurve: s1 at: 0.5.
```

```
<point> canvas middleOf: p1 and: p2
```

p1 : référence d'un point ou d'un couple de coordonnées

p2 : référence d'un point ou d'un couple de coordonnées

Retourne : référence du milieu des deux points

Exemple :

```
| a i |
```

```
a := canvas point: 1@1.
```

```
i := canvas middleOf: a and: 4@4.
```

```
<point> canvas middleOf: s
```

s : référence d'un segment

Retourne : référence du milieu du segment

Exemple :

```
canvas middleOf: s.
```

```
<point> canvas intersectionOf: l1 and: l2
```

l1 : référence d'une ligne

l2 : référence d'une ligne

Retourne : référence du point d'intersection des deux lignes

Exemple :

```
canvas intersectionOf: droite and: segment
```

```
<point> canvas altIntersectionOf: l1 and: l2
```

l1 : référence d'une ligne

l2 : référence d'une ligne

Retourne : référence de l'autre point d'intersection des deux lignes, lorsqu'il existe

Exemple :

```
canvas altIntersectionOf: droite and: circle.
```

```
<point> canvas point: bloc parent: item
```

bloc : bloc de code retournant un point

item : référence d'un item géométrique

Retourne : référence d'un point dont les coordonnées sont calculées avec le bloc de code ayant comme argument item

Exemple :

```
| figure s mobile c block |
figure := DrGeoCanvas new.
s:=figure
  segment: (figure point: -5@0)
  to: (figure point: 5@0).
mobile := figure pointOnCurve: s at: 0.1.
block := [:mathItem | |x|
  x := mathItem point x.
  x @ (x * x * x / 25 - x)].
c := figure point: block parent: mobile.
figure locusOf: c when: mobile.
```

```
<point> canvas point: bloc parents: itemCollection
```

bloc : bloc de code retournant un point

itemCollection : une collection d'items géométriques

Retourne : référence d'un point dont les coordonnées sont calculées avec le bloc de code ayant comme argument itemCollection

Exemple :

```
| figure a b d m p |
figure:=DrGeoCanvas new.
a:=figure point: (-2)@1.
b:=figure point: 3@3.
d:=figure line: a to: b.
d color: Color blue.
m:=figure point: 1@(-1).
p:= figure
  point: [:parents | parents first closestPointTo: parents second point]
  parents: {d . m}.
```

Droite.

```
<line> canvas line: p1 to: p2
```

p1 : référence d'un point ou d'un couple de coordonnées

p2 : référence d'un point ou d'un couple de coordonnées

Retourne : référence d'une droite passant par deux points

Exemple :

```
| p1 |
```

```
p1 := canvas point: 0@0.
```

```
canvas line: p1 to: 1@2.
```

```
<line> canvas parallel: d at: p
```

p : référence d'un point ou d'un couple de coordonnées

d : référence d'une direction (droite, segment, vecteur, ...)

Retourne : référence d'une droite parallèle à la direction de d et passant par p

Exemple :

```
| a |
```

```
a := canvas point: 1@5.
```

```
canvas parallel: d at: a.
```

```
<line> canvas perpendicular: d at: p
```

p : référence d'un point ou d'un couple de coordonnées

d : référence d'une direction (droite, segment, vecteur, ...)

Retourne : référence d'une droite perpendiculaire à la direction de d et passant par p

Exemple :

```
canvas perpendicular: d at: 1@5.
```

Demi-droite.

```
<ray> canvas ray: o to: p
```

o : référence d'un point ou d'un couple de coordonnées, origine de la demi-droite

p : référence d'un point ou d'un couple de coordonnées, point de la demi-droite

Retourne : référence d'une demi-droite définie par son origine et un point

Exemple :

```
| a |
```

```
a := canvas point: 1@5.
```

```
canvas ray: 0@0 to: a.
```

Segment.

```
<segment> canvas segment: p1 to: p2
```

p1 : référence d'un point ou d'un couple de coordonnées

p2 : référence d'un point ou d'un couple de coordonnées

Retourne : référence d'un segment défini par ses extrémités

Exemple :

```
| a |
```

```
a := canvas point: 5@5.
```

```
canvas segment: 10@10 to: a.
```

Cercle.

```
<circle> canvas circleCenter: c to: p
```

c : référence d'un point ou d'un couple de coordonnées, centre du cercle

p : référence d'un point ou d'un couple de coordonnées, point du cercle

Retourne : référence d'un cercle défini par son centre et un point

Exemple :

```
| a |
```

```
a := canvas point: 1@5.
```

```
canvas circleCenter: a to: 10@4.
```

```
<circle> canvas circleCenter: c radius: r
```

c : référence d'un point ou d'un couple de coordonnées, centre du cercle

r : référence d'un item numérique ou d'une valeur numérique, rayon du cercle

Retourne : référence d'un cercle défini par son centre et son rayon

Exemple :

```
| a r |
```

```
a := canvas point: 1@5.
```

```
r := canvas freeValue: 4.
```

```
canvas circleCenter: a radius: r.
```

```
canvas circleCenter: 4@4 radius: 5
```

Arc de cercle.

```
<arc> canvas arc: p1 to: p2 to:p3
```

p1 : référence d'un point ou d'un couple de coordonnées, 1^{ère} extrémité de l'arc

p2 : référence d'un point ou d'un couple de coordonnées de l'arc

p3 : référence d'un point ou d'un couple de coordonnées, 2^{ème} extrémité de l'arc

Retourne : référence d'un arc de cercle défini par ses extrémités et un point

Exemple :

```
| a b |
```

```
a := canvas point: 1@5.
```

```
b := canvas point: 0@5.
```

```
canvas arc: a to: b to: -1 @ -2.
```

Polygone.

```
<polygon> canvas polygon: collection
```

collection : une liste de références de points ou de couples de coordonnées; sommets du polygone

Retourne : référence d'un polygone défini par ses sommets

Exemple :

```
| b |
```

```
b := canvas point: 1@3.
```

```
canvas polygon: {1@2. b. 0@0. d.}.
```

Les transformations géométriques.

Les transformations géométriques permettent la construction des transformés d'objets. Elles s'appliquent à des références d'objets de type point, segment, droite, demi-droite, vecteur, cercle, arc de cercle et polygone.

```
<transformed item> canvas rotate: item center: centre angle: angle
```

item : point, segment, droite, demi-droite, vecteur, cercle, arc-cercle, polygone

objet : référence de l'objet à transformer

centre : référence d'un point ou d'un couple de coordonnées, centre de la rotation

angle : référence d'un item valeur ou d'une valeur, angle de la rotation

Retourne : référence de l'objet transformé

Exemple :

```
| c k l |
```

```
c := canvas point: 5@5.
```

```
k := 3.1415.
```

```
l := canvas line: 0@0 to: 5@5.
```

```
canvas rotate: l center: c angle: k.
```

```
canvas rotate: l center: 0@0 angle: Float pi / 3.
```

```
<transformed item> canvas scale: item center: centre factor: k
```

item : point, segment, droite, demi-droite, vecteur, cercle, arc-cercle, polygone

objet : référence de l'objet à transformer

centre : référence d'un point ou d'un couple de coordonnées, centre de l'homothétie

k : référence d'un item valeur ou d'une valeur, facteur de l'homothétie

Retourne : référence de l'objet transformé

Exemple :

```
| c k l |
```

```
c := canvas point: 5@5.
```

```
k := -3.
```

```
l := canvas line: 0@0 to: 5@5.
```

```
canvas scale: l center: c angle: k.
```

```
canvas scale: l center: 0@0 angle: 5.
```

```
<transformed item> canvas symmetry: item center: centre
```

item : point, segment, droite, demi-droite, vecteur, cercle, arc-cercle, polygone

centre : référence d'un point ou d'un couple de coordonnées, centre de la symétrie

Retourne : référence de l'objet transformé

Exemple :

```
| a |
```

```
a := canvas point: 4@2.
```

```
canvas symmetry: a center: 0@0.
```

```
<transformed item> reflect: item axe: axe)
```

item : point, segment, droite, demi-droite, vecteur, cercle, arc-cercle, polygone

axe : référence d'une droite, axe de la réflexion

Retourne : référence de l'objet transformé

Exemple :

```
canvas reflect: polygon axe: d1.
```

```
<transformed item> canvas translate: item vector: vecteur
```

item : point, segment, droite, demi-droite, vecteur, cercle, arc-cercle, polygone

vecteur : référence d'un vecteur ou d'un couple de coordonnées

Retourne : référence de l'objet transformé

Exemple :

| u a |

u := canvas vector: (canvas point: 1@1) to: (canvas point: 3@2).

a := canvas translate: (canvas point: 2@1) vector: u.

Exemple :

| u a |

a := canvas translate: (canvas point: 2@1) vector: 2@1.

Lieu d'un point.

`<locus> canvas locusOf: c when: m`

m : référence d'un point mobile sur une ligne

c : référence d'un point fixe dépendant du point m

Retourne : référence d'un lieu

Exemple :

canvas locusOf: p when: mobile.

Vecteur.

`<vector> canvas vector: o to: e`

o : référence d'un point ou d'un couple de coordonnées, origine du vecteur

e : référence d'un point ou d'un couple de coordonnées, extrémité du vecteur

Retourne : référence d'un vecteur

Exemple :

| b |

b := canvas point: 0@5.

canvas vector: b to: -1 @ -2.

`<vector> canvas vector: p`

p : référence d'un point ou d'un couple de coordonnées, coordonnées du vecteur

Retourne : référence d'un vecteur

Exemple :

| p |

p := canvas point: 5@5.

canvas vector: p.

canvas vector: -5 @ -5

Nombre.

`<point> pointItem coordinates`

pointItem : un point géométrique

Retourne : coordonnées (statiques) de pointItem

Exemple :

| c p |

p := canvas pointOnCurve: segment at: 0.5.

c := p coordinates.
 c x

`<value> canvas freeValue: v`

v : la valeur initiale du nombre
Retourne : référence d'un nombre libre
Exemple :
 canvas freeValue: (-1 arcCos).

`<value> canvas lengthOf: segment | circle | arc | vector)`

segment|circle|arc|vector : référence d'un segment, cercle, arc ou vecteur
Retourne : référence d'un nombre, longueur de l'item
Exemple :
 canvas lengthOf: v1.

`<value> canvas distance: line | point to: point2`

line|point : référence d'une ligne ou d'un point
 point2 : référence d'un point
Retourne : référence d'un nombre, distance entre deux points ou un point et une droite
Exemple :
 canvas distance: l1 to: a.

`<value> canvas slopeOf: line`

line : référence d'une droite
Retourne : référence d'un nombre, pente de la droite
Exemple :
 | p |
 p := canvas slopeOf: d.

Angle.

`<value> canvas angle: a to: b to: c`

a : référence d'un point
 b : référence d'un point, sommet de l'angle
 c : référence d'un point
Retourne : référence d'un angle géométrique \widehat{abc}
Exemple :
 canvas angle: a to: b to: c

`<value> canvas angle: v1 to: v2`

v1 : référence d'un vecteur
 v2 : référence d'un vecteur
Retourne : référence d'un angle orienté formé par les deux vecteurs
Exemple :
 | v1 v2 a |
 v1 := canvas vector: a to: b.
 v2 := canvas vector: a to: c.
 a := canvas angle: v1 to: v2.

Modification d'attributs d'objets

Pour modifier les attributs d'un objet déjà créé, nous utilisons un système de messages envoyés directement à l'objet. La modification des attributs se fait donc toujours à posteriori.

`item color: aColor`

item : référence d'un objet

aColor : une instance de `Color`, voir ses méthodes de classe pour des définitions existantes : `Color black`, `Color red`, `Color blue`, `Color orange`, `Color yellow`,...

Action : modifie la couleur d'un item

Exemple :

`pointA color: Color green.`

`item name: aString`

aString : une chaîne de caractères

Action : renomme un item

Exemple :

`segment name: '[AB]'.`

`item hide`

Action : masque un item

`item show`

Action : montre un item

`line small`

line : référence d'une ligne (droite, demi-droite, cercle, lieu, etc.)

Action : donne une épaisseur fine à une ligne

Exemple :

`circle small.`

`line normal`

line : référence d'une ligne (droite, demi-droite, cercle, lieu, etc.)

Action : donne une épaisseur normale à une ligne

Exemple :

`arc normal.`

`line large`

line : référence d'une ligne (droite, demi-droite, cercle, lieu, etc.)

Action : donne une épaisseur large à une ligne

Exemple :

`polygon large.`

`line plain`

line : référence d'une ligne (droite, demi-droite, cercle, lieu, etc.)

Action : donne un style ligne continue

Exemple :

`polygon plain.`

line dashed

line : référence d'une ligne (droite, demi-droite, cercle, lieu, etc.)

Action : donne un style en tirets à une ligne

Exemple :

polygon dashed.

line dotted

line : référence d'une ligne (droite, demi-droite, cercle, lieu, etc.)

Action : donne un style en pointillés à une ligne

Exemple :

arc dotted.

point cross

point : référence d'un point

Action : donne une forme en croix à un point

Exemple :

a cross.

point round

point : référence d'un point

Action : donne une forme en rond à un point

Exemple :

a round.

point square

point : référence d'un point

Action : donne une forme carrée à un point

Exemple :

a square.

point small

point : référence d'un point

Action : donne une petite taille à un point

Exemple :

a small.

point large

point : référence d'un point

Action : donne une taille large à un point

Exemple :

a large.

item moveTo: point

item : référence d'un point ou d'une valeur

point : couple de coordonnées

Action : déplace l'item à la position donnée, pour peu que cela ait un sens

Exemple :

| a |

```
a := canvas point: 0@0.
a moveTo: 5@5.
canvas update
```

4.3.3 Galerie d'exemples

Pour illustrer l'utilisation des Figures Smalltalk DR. GEO, nous vous proposons une petite série d'exemples. Ceux-ci vous montrent leurs importantes possibilités et nous espérons qu'ils seront également une source d'inspiration. Pour chacun de ces exemples, nous donnons le code source Smalltalk de la figure puis son résultat. Le code source doit être copié dans un workspace de l'environnement de DR. GEO puis exécuté.

Animer une figure

Ces exemples s'appuient sur la gestion du temps et celle des processus programmés en Smalltalk.

Un premier exemple simple pour comprendre le principe :

```
| figure p pause |
figure:=DrGeoCanvas new.
p := figure point: 0@0.
pause := Delay forSeconds: 0.2.
[100 timesRepeat: [
p mathItem moveTo: (p mathItem point + (0.1@0)).
figure update.
pause wait]
] fork
```

Un deuxième exemple avec une figure plus élaborée :

```
| figure s r u pause |
figure:=DrGeoCanvas new.
s:=figure segment: (figure point: 0@(-1)) to: (figure point: 4@(-1)).
r:=figure pointOnCurve: s at: 0.8.
s:=figure segment: (figure point: 0@0) to: (figure point: 0@1).
u:=figure pointOnCurve: s at: 0.7.
u round small.
u color: Color blue.
1 to: 100 do: [:n|
    u:=figure point: [:parents| |y t|
        y:=parents first point y.
        t:=parents second point x.
        (n/5)@t*y*(1-y)]
        parents: {u.r}.
        u round small.
        u color: Color blue.
    ].
pause := Delay forSeconds: 0.1.
[0 to: 1 by: 0.05 do:[:x |
r mathItem setCurveAbscissa: x.
figure update.
pause wait]
] fork
```

Triangle de Sierpinski

Cet exemple s'appuie largement sur un bloc de code récursif.

```
| triangle c |
triangle := [:s1 :s2 :s3 :n |
  c segment: s1 to: s2;
  segment: s2 to: s3;
  segment: s3 to: s1.
n > 0 ifTrue:
  [triangle
   value: s1
   value: (c middleOf: s1 and: s2) hide
   value: (c middleOf: s1 and: s3) hide
   value: n-1.
  triangle
   value: (c middleOf: s1 and: s2) hide
   value: s2
   value: (c middleOf: s2 and: s3) hide
   value: n-1.
  triangle
   value: (c middleOf: s1 and: s3) hide
   value: (c middleOf: s2 and: s3) hide
   value: s3
   value: n-1.]].

c := DrGeoCanvas new.
triangle
value: (c point: 0 @ 3)
value: (c point: 4 @ -3)
value: (c point: -4 @ -3)
value: 3.
```

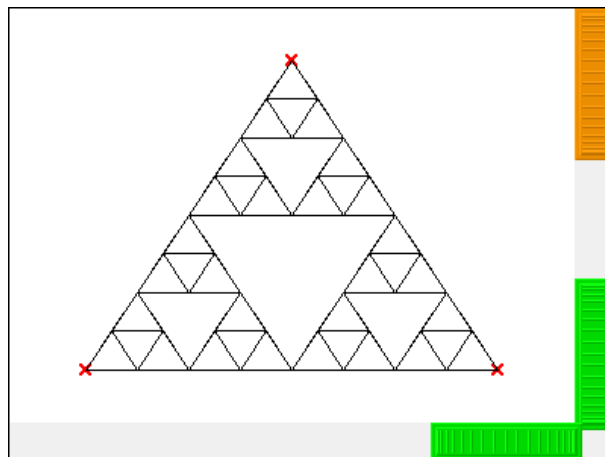


FIGURE 4.14 – Triangle de Sierpinski

Chapitre 5

Applications didactiques

Ce chapitre est une aide pour étudier DR. GEO à partir d'exemples. Contrairement aux chapitres précédents, l'approche est plus concrète par rapport à des situations précises. Le contenu de ce chapitre s'est constitué lors de diverses activités pédagogiques.

Chapitre 6

Astuces diverses

Du fait de son intégration dans l'environnement Pharo Smalltalk, DR. GEO recèle quelques génies, nombres de ceux-ci sont cachés à la vue de l'utilisateur. Ce n'est pas tant le souhait d'en restreindre l'accès mais plutôt le souci d'alléger la charge cognitive avec une interface simple. Comme nous allons le voir dans les sections suivantes, ces génies s'invoquent par raccourcis clavier, commandes de menu ou codes Smalltalk.

6.1 Programmation

Dans cette section nous présentons quelques outils à utiliser lors de l'écriture de scripts ou de figures programmées : l'espace de travail, le débogueur, l'inspecteur, etc.

6.1.1 Espace de travail

Pour l'afficher, cliquer sur le fond de l'environnement DR. GEO puis faire `CTRL-K`¹.

Un espace de travail ressemble à s'y méprendre à un éditeur de texte. Mais c'est en fait une console d'édition de code Smalltalk : pour écrire, compiler et exécuter du code source, il est bien sûr possible d'y coller un code copié ailleurs. Après l'invocation d'un espace de travail², y coller le code source de la figure programmée ci-dessous³ :

```
| figure fonction p integrale sommets |  
  
fonction:= [ :x | x*x ].  
sommets:=OrderedCollection new.  
figure:=DrGeoCanvas new.  
p:=figure point: -1@0.  
p hide.  
sommets add: p.  
  
-1 to: 1 by: 0.1 do: [ :x |  
p:=figure point: x@(fonction value: x).  
sommets add: p hide].  
  
p:=figure point: 1@0.  
sommets add: p hide.
```

1. Selon votre système d'exploitation, remplacer `CTRL` par `ALT`

2. Contrairement aux génies, vous pouvez invoquer plusieurs espaces de travail.

3. Pour coller un texte, essayer avec le raccourci clavier `CTRL-V` ou depuis le menu contextuel de l'espace de travail (clic droit de souris).

```
integrale:=figure polygon: sommets.
integrale color: Color blue.
```

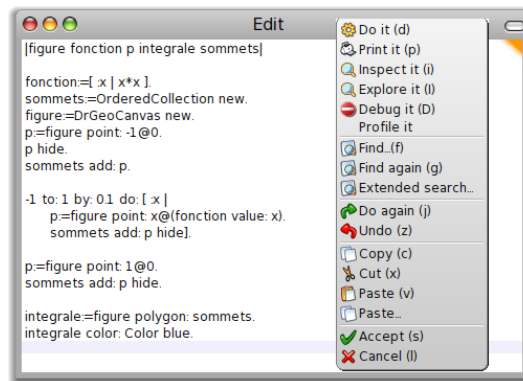


FIGURE 6.1 – Votre espace de travail avec le code source collé et son menu contextuel

Pour compiler et exécuter ce code source, il suffit de le sélectionner à la souris et d’invoquer *Do it(d)* dans le menu contextuel. Ces deux opérations se font également au clavier par un `CTRL-A` suivi d’un `CTRL-D`. Vous obtenez alors immédiatement le résultat de cette figure programmée, sous la forme d’une figure interactive dans un canevas DR. GEO.



FIGURE 6.2 – Résultat de l’exécution du code source : intégrale de la fonction sur $[-1; 1]$

Lors de l’exécution d’un code source complexe, lancer celui-ci avec l’option de profilage permet de trouver les goulots d’étranglement. Pour ce faire, dans le menu contextuel invoquer la commande *Profile it*. Le code source est exécuté, le canevas DR. GEO affiché et en plus la fenêtre du profileur informe l’utilisateur sur le temps d’exécution du code et des méthodes invoquées. C’est un outil remarquable pour naviguer dans l’arbre d’exécution du code et afficher les méthodes posant problème.

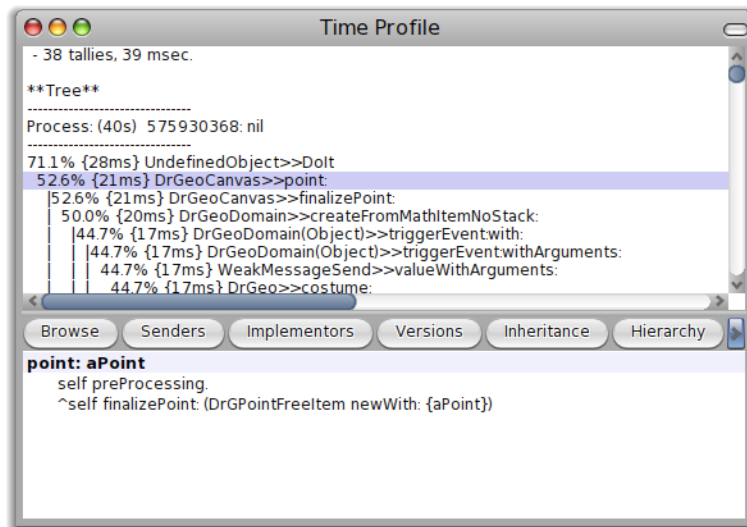


FIGURE 6.3 – Le profileur DR. GEO

Dernier raffinement : l'exécution en mode pas à pas du code source. Cela se fait en l'exécutant avec le débogueur, dans le menu contextuel choisir la commande *Debug it*. Le débogueur est invoqué sur la première ligne, le code s'exécute pas à pas avec le bouton *Over*. Dans la partie basse à droite, les variables locales et leur contenu est consultable et modifiable. Les autres boutons permettent d'autres raffinements dans l'exécution pas à pas, à vous de les explorer !

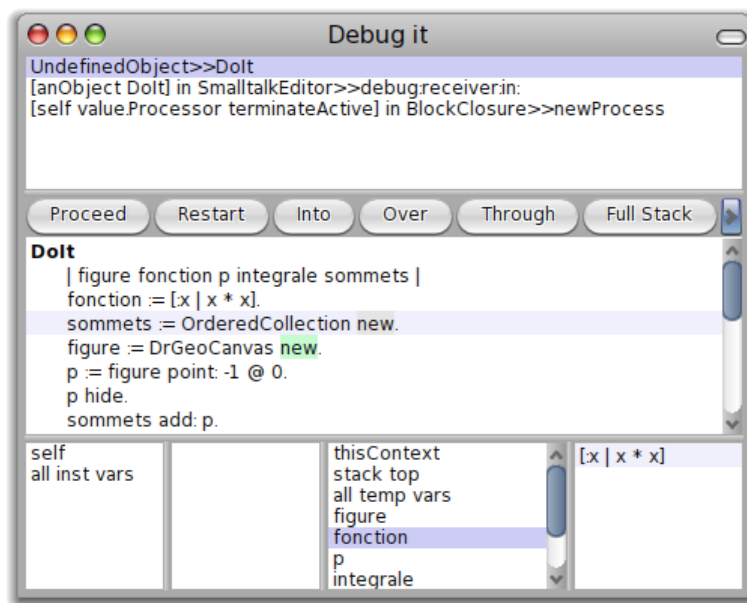


FIGURE 6.4 – Le débogueur DR. GEO

6.1.2 Débogueur

Comme montré dans une section précédente, le débogueur permet l'exécution en mode pas à pas. Il s'invoque à n'importe quel moment avec le raccourci clavier `[ALT-.]`.

En outre, le débogueur s'enclenche également par programmation, directement dans le code source en ajoutant une ligne `self halt`. Dans notre exemple précédent, nous pouvons modifier le code source comme suit :

```
...
p:=figure point: -1@0.
p hide.
sommets add: p.

self halt.

-1 to: 1 by: 0.1 do: [ :x |
p:=figure point: x@(fonction value: x).
sommets add: p hide].
...
```

6.1.3 Inspecteur

Avec l'inspecteur, l'utilisateur consulte les attributs d'une instance ou bien le contenu d'une variable.

Dans notre exemple, supposons que nous souhaitons voir le contenu de la collection `sommets`. Dans ce cas rien de plus simple, nous ajoutons une ligne de code où nous envoyons le message `inspect` à `sommets`, l'emplacement où se fait cette invocation n'est pas très important car nous n'avons ni point d'arrêt, ni exécution en mode pas à pas :

```
...
p:=figure point: -1@0.
p hide.
sommets add: p.

sommets inspect.

-1 to: 1 by: 0.1 do: [ :x |
p:=figure point: x@(fonction value: x).
sommets add: p hide].
...
```

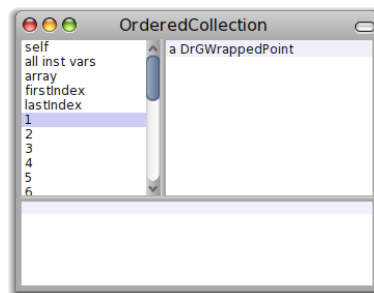


FIGURE 6.5 – L'inspecteur sur la variable `sommets`

Index

- Angle
 - Géométrie, 15
 - Orienté, 15
- Arc de cercle, 14
 - Longueur, 15
- Autres outils, 16
- Cercle, 13
 - Périmètre, 15
- Débogueur, 59
- Demi-droite, 13
- Droite, 13
 - Distance, 15
 - Parallèle, 13
 - Pente, 15
 - Perpendiculaire, 13
- Editer
 - Propriété, 18
 - Style, 16
- Espace de travail, 57
 - Coller du code, 57, 58
 - Exécution pas à pas du code, 59
 - Profilier du code, 58
- Figure
 - Déplacer, 21
 - Enregistrer, 23
 - Session, 23
 - Grossissement, 21
 - Nouvelle, 9
 - Ouvrir, 24
 - Renommer, 23
- Figure Smalltalk, 40
 - Création d'objets
 - Angle, 49
 - Arc de cercle, 46
 - Attributs des objets, 49
 - Cercle, 45
 - Demi-droite, 45
 - Droite, 44
 - Lieu, 48
 - Nombre, distance, coordonnées, 48
 - Point, 43
 - Polygone, 46
 - Segment, 45
 - Transformations géométriques, 46
 - Vecteur, 48
 - Exécuter, 41
 - Exemples, 41, 52
 - Animation, 52
 - Triangle de Sierpinski, 52
- FSD, 40
- Grille
 - Afficher, 21
 - Aimantée, 22
- Inspecteur, 60
- Lieu géométrique, 14
 - Script, 35
- Macro-construction, 16
 - Créer, 26
 - Enregistrer, 23, 24
 - Introduction, 25
 - Jouer, 28
 - Boîte de dialogue, 28
 - Ouvrir, 24
- Nombre, 15
 - Valeur libre, 15
- Objet
 - Déplacer, 21
 - Masquer, 16
 - Propriété, 18
 - Renommer, 16
 - Style, 16
 - Supprimer, 16
- Point
 - Coller, 17
 - Coordonnées, 15
 - Défini par coordonnées, 12
 - Intersection, 12
 - Libre, 12
 - Milieu, 12
 - Muter, 21
 - Renommer, 17
- Point libre

- Propriété, 18
 - sur ligne
 - Propriété, 18
- Polygone, 14
- Renommer
 - Objet, 16
- Script, 16, 25
 - valeur aléatoire, 32
 - API
 - abscissa:, 38
 - abscissaOf:, 38
 - abscissa, 38
 - center, 39
 - closestPointTo:, 38
 - contains:, 39
 - degreeAngle, 40
 - direction, 39
 - exist, 37
 - length, 40
 - move:, 37
 - moveAt:, 38
 - normal, 39
 - origin, 39
 - parents, 37
 - point:, 38
 - pointAt:, 39
 - point, 38
 - position:, 40
 - radianAngle, 40
 - radius, 39
 - safeName, 37
 - valueItem:, 40
 - valueItem, 40
 - Coordonnées
 - Point, 35
 - Courbe
 - Tangente, 34
 - Exemples, 30
 - Complexe, 34
 - Simple, 31
 - Tangente à une courbe, 36
 - Fonction, 34
 - Image, 35
 - Introduction, 30
 - Paramètres
 - 0, 31
 - 1, 33
 - 2, 34
 - Propriété, 19
- Segment, 13
 - Longueur, 15
 - Marquer, 17
- Session
 - Enregistrer, 23
 - Ouvrir, 24
- Style, 16
 - Ligne, 17
 - Point, 17
 - Polygone, 17
 - Script, 17
 - Segment
 - Marquer, 17
 - Valeur, 17
- Texte, 16
 - Propriété, 19
- Transformation, 14
 - Figure Smalltalk, 46
 - Homothétie, 15
 - Rotation, 14
 - Symétrie
 - Axiale, 14
 - Centrale, 14
 - Translation, 14
- Valeur
 - Coller, 17
- Valeur libre
 - Propriété, 19
- Vecteur, 13
 - Coordonnées, 15
 - Norme, 15