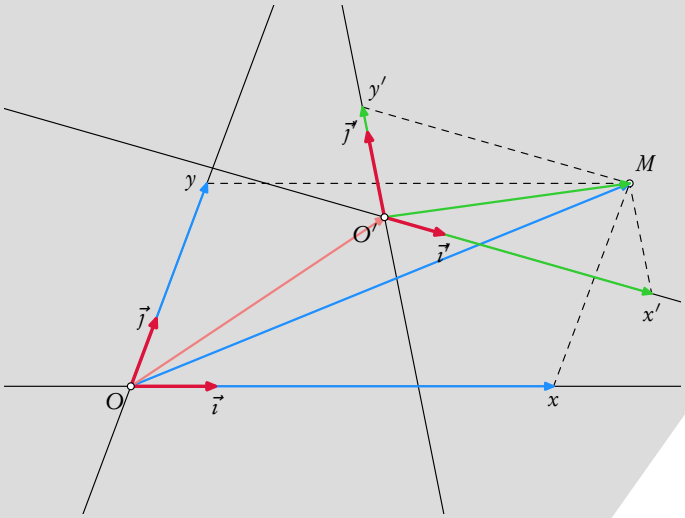
mp-gdd

Paquet METAPOST pour des figures de géométrie plane



Contributeurs
Maxime CHUPIN
Jean-Michel SARLAT

Version 0.1 du 25 septembre 2011
http://melusine.eu.org/syracuse/G/mp-gdd/

1 Introduction

mpgdd est un ensemble d'outils pour la géométrie plane avec METAPOST. Cet ensemble se compose de plusieurs fichiers :

1. gdd.mp : c'est le fichier principal, il contient les structures et fonctions générales.
2. gdd-arc.mp : contient tout ce qui concerne les arcs de cercles.
3. gdd-c2d.mp : contient tout ce qui concerne les courbes du second degré.
4. gdd-fct.mp : contient quelques fonctions mathématiques usuelles.
5. gdd-lbl.mp : contient les fonctions relatives aux labels.
6. gdd-plt.mp : contient des fonctions facilitant la représentation de fonctions mathématiques.
7. gdd-rep.mp contient différents outils pour le tracé de figure dans un repère.
8. gdd-tra.mp contient les fonction permettant de gérer la transparence (code emprunté à Anthony PHAN).

Nous allons, dans la suite, décrire plus en détails chacune de ces fonctions. Il est à noté aussi que certaines fonctions s'appuient sur l'extension graph.mp présent dans toutes les bonnes distributions TEX.

2 Objectif

mpgdd a été écrit avec le but de proposer des macros METAPOST permettant de réaliser une figure de géométrie *en collant* d'assez près à une description impérative :

Soit A le point de coordonnées (2,3).
Soit B le point de coordonnées (4,5).
Trace la droite (A,B).
....

Dans ce cadre, les objets géométriques sont le plus souvent nommés (A, B, etc.) ou désignés par leur nature et leurs attributs (droite (A,B), etc.). Pour ne pas avoir à dépasser ce mode de description, en particulier pour éviter d'avoir à déclarer le *type* de ces objets, le choix a été fait de les identifier par un *index* [1] dans des tables qui en précisent les caractéristiques.

Note – À ce jour, l'objectif n'est pas atteint, le développement est loin d'être achevé ; il est encore nécessaire de faire appel à des commandes METAPOST ou à des *macros intermédiaires* pour décrire une figure. Cela évoluera sans doute avec le temps, le temps de trouver une syntaxe satisfaisante...

3 Principe général de fonctionnement

mpgdd utilise des tables comme structure principale. Chaque objet est numéroté via le compteur gdd0, son type [2] est stocké dans la table gddT[] à la place gddT[gdd0]. Les propriétés des objets sont définies dans, là encore, des tables de type numeric qui sont gddA[], gddB[], ..., gddF[].

Par exemple, pour un Point (type mpgdd), la première coordonnée se trouve dans gddA[] et la seconde dans gddB[] (les autres table ne sont pas utilisées pour un tel objet).

Il y a deux tables particulières gddP[] qui est du type path et gddS[] qui est du type string. Nous verrons plus tard quelle est leur utilité.

Bien entendu, lors d'une utilisation classique de mpgdd, l'appel à toutes ces tables n'est pas chose courante. Les fonctions que nous allons décrire dans la suite de ce documents permettent de ne pas avoir recours trop précisément à cette machinerie.

4 Les types

On peut, avec mpgdd, construire plusieurs types d'objets. Rappelons le, tout est *objet* dont le nombre est enregistré dans la variable gdd0. Le type d'objet, lui, est stocké dans la table gddT[], et les tables gddA[] à gddF[] contiennent les propriétés des objets.

Nous allons ici décrire chaque type de l'extension mpgdd ainsi que leurs propriétés respectives.

1. Le type numeric, qui est le type par défaut dans METAPOST, ne demande pas de déclaration préalable.
2. Les types sont propres à mpgdd et seront décrits plus tard.

Le type `Point` Ce type correspond au point de l'espace euclidien. Pour être plus clair voici la fonction principale pour créer un tel objet :

```
vardef Point(expr a,b) =
  gddT[incr gdd0] = "point";
  gddA[gdd0] = a; gddB[gdd0] = b; gdd0
enddef;
```

Cette fonction «retourne» le compteur `gdd0` et crée dans la table de type une entrée `point` et les attributs (coordonnées) correspondants `a` et `b` dans les tables `gddA` et `gddB`.

Avec un tel type de fonctionnement, la plupart des manipulations se fait sur des `numerics`. En effet, pour déclarer un `point`, il suffit d'écrire

```
A = Point(2,3);
```

`A` prend alors la valeur courante de `gdd0`. C'est l'identifiant du point.

Le type `Vecteur` Ce type correspond aux vecteurs définis à l'aide de deux points de l'espace euclidien. La fonction créatrice d'un tel objet est celle-ci

```
vardef Vecteur(expr a,b) =
  save n; n = incr gdd0;
  gddT[n] = "vecteur"; gddA[n] = PointImp(a); gddB[n] = PointImp(b); n
enddef;
```

Cette fonction a la même architecture que celle correspondante au `point` : elle retourne la valeur courante de `gdd0` après incrémentation, puis affecte le type `vecteur` à l'entrée correspondante dans la table `gddT`. Par contre, les entrées des tables `gddA` et `gddB` sont différentes. En effet, on stock ici les identifiants des points formant le vecteur. La fonction `PointImp` s'assure que l'entrée correspond bien à un identifiant de point et non pas une `pair` METAPOST. Elle sera décrite plus en détail ultérieurement.

5 Fonctions générales

5.1 Relatives aux `Points`

5.2 Un peu de géométrie