

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 noter aussi que certaines fonctions s'appuient sur l'extension `graph.mp` présent dans toutes les bonnes distributions \TeX .

2 Principe général de fonctionnement

mpgdd utilise des tables comme structure principale. Chaque objet est numéroté via le compteur `gdd0`, son type¹ 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.

3 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.

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 :

```
1 vardef Point(expr a,b) =
2   gddT[incr gdd0] = "point";
3   gddA[gdd0] = a; gddB[gdd0] = b; gdd0
4 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 `numeric`s. En effet, pour déclarer un `point`, il suffit d'écrire

```
1 A = point(2,3);
```

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

1. Les types sont propres à `mpgdd` et seront décrit plus tard.

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

```
1 vardef Vecteur(expr a,b) =  
2   save n; n = incr gdd0;  
3   gddT[n] = "vecteur"; gddA[n] = PointImp(a); gddB[n] = PointImp(b); n  
4 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 paire `METAPOST`. Elle sera décrite plus en détail ultérieurement.

4 Fonctions générales

4.1 Relatives aux Points

4.2 Un peu de géométrie