3.4 Animation

Modifiez le "grapheur", en ajoutant une variable z. Vous créerez des animations en faisant varier la valeur de z, et en visualisant les points x, y, z tels que $f(x, y, z) = 0$. Prévoir des boucles pour faire varier z : par exemple avec la syntaxe "for z = 0 to 10 step 0.01 do". Ce qui est visualisé est donc une coupe d'une surface par un plan.

3.5 Courbes

Modifiez le "grapheur", en ajoutant une variable z. Des courbes en 3D seront décrites par l'intersection de 2 surfaces $f(x, y, z) = 0$ et $g(x, y, z) = 0$. Vous dessinerez la projection de ces courbes sur le plan Oxy, autrement dit tout point (x, y, z) de la courbe sera affiché en (x, y). Ce grapheur peut afficher les courbes paramétrées $x = f(t), y = g(t)$; il suffit de renommer t en z, et de considérer: $x - f(z) = y - g(z) = 0$. Le principe est de subdiviser l'espace 3D.

3.6 Scènes 2D (à 1 ou 2)

Décrivez des scènes 2D, ou 3D, dans un format libre, et visualisez les en openGl. Par exemple, les scènes 2D sont constituées de carrés, de cercles, de polygones, de segments. Les scènes 3D sont constituées de cubes, tétraèdres, sphères, etc. Le langage de description n'a pas à fournir les variables, les boucles, ni les fonctions. Vous pouvez réutiliser ou vous inspirer du fichier fig.cpp.

4 Projets un peu plus difficiles

Attention, les projets suivants sont un peu plus difficiles, et nécessitent plus de temps. Vous pouvez vous mettre à plus de 2 pour les faire. Faire ces projets vous apprendra beaucoup.

4.1 Petit interprête (1 à 2 jours de travail)

Attention, ce projet est plus difficile ! Modifiez la calculatrice, pour écrire un petit interprète. La syntaxe de votre langage est libre. Seul le type entier est demandé. Votre langage doit permettre de définir la fonction factorielle et la fonction fibonacci avec la récursion naïve. Pour gérer l'évaluation, vous pouvez utiliser une pile de couples : (nom de variable, valeur entière). La valeur d'une variable est la valeur entière stockée la plus haut dans la pile. Pour évaluer un appel de fonction, il faut évaluer (par un appel récursif à votre fonction d'évaluation) les arguments de la fonction, puis empiler les couples pour les paramètres de la fonction et leurs valeurs. Variante : vous pouvez gérer une pile de valeurs par nom de variable.