Sujet Projet Informatique:
Visualiseur de surfaces paramétriques

2011

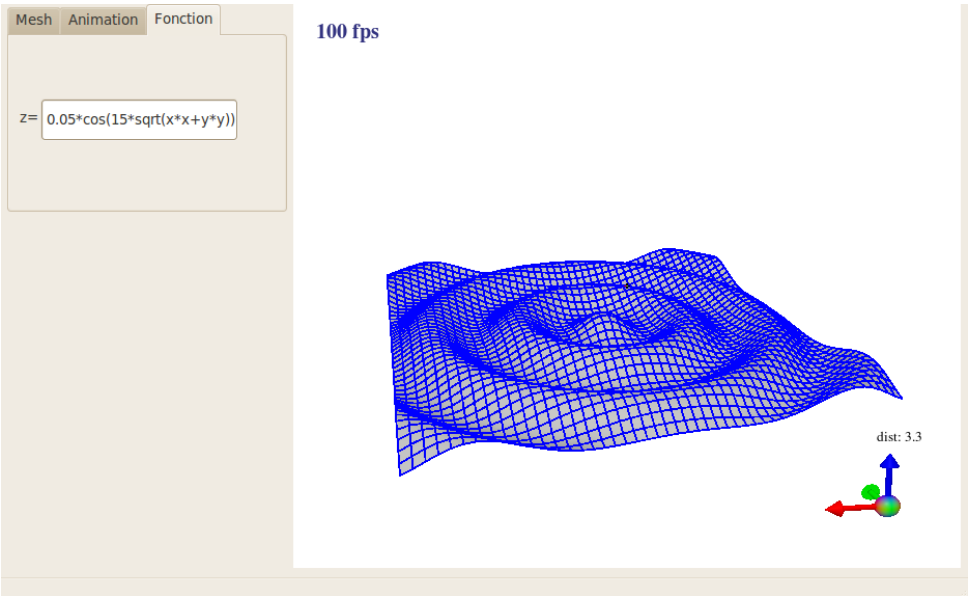

FIGURE 1 – Exemple d'interface et de visualisation de surface paramétrique. La fonction mathématique est écrite textuellement, et sa représentation 3D est mise à jour automatiquement.

1 But

L'objectif de ce projet est de coder un visualiseur de surfaces paramétriques à l'aide de l'API 3D OpenGL.

Le sujet se décompose en deux parties :

1. Evaluation d'une fonction mathématique donnée en tant que chaine de caractères.
2. Affichage de la surface correspondante en 3D.

L'application de ce type de représentation se retrouve dans des logiciels mathématiques (Maple, Mathematica, Matlab, ...), mais également dans des outils de modélisation 3D (Blender, 3DStudio, Maya, ...).

2 Fonctionnement du projet

Un exemple d'interface issue de ce projet peut ressembler à celle illustrée en fig. 1 réalisant le cas particulier de fonctions dites de *champs de hauteurs*. Un utilisateur écrit sous forme de texte une fonction f tel que :

$$f : \left\{ \begin{array}{ccc} \mathbb{R}^2 & \rightarrow & \mathbb{R} \\ (x,y) & \mapsto & f(x,y)\,. \end{array} \right. \tag{1}$$

L'expression est stockée en tant que chaine de caractères, et un arbre d'évaluation est généré en parsant le texte et en prenant en compte la priorité des opérateurs. Cet arbre d'évaluation permet de générer une fonction

```
double evaluate_function(double x,double y);
```

qui, étant donné un couple de valeur (x, y) calcul et renvoie la valeur $f(x, y)$ en parcourant le graphe construit.

Dans un second temps, une surface est ´échantillonnée suivant une grille uniforme (x, y). On affiche alors la surface d'équation $z = f(x, y)$ dans un contexte OpenGL.

3 Étapes du projet

3.1 Évaluation d'une fonction mathématique

Dans un premier temps, on pourra simplifier la tâche de l'évaluation de la fonction mathématique. Dans l'ordre de difficulté :

1. Codage en dur de la fonction.
2. Envoi des opérandes en notation polonaise inversée.
3. Analyse d'un texte avec priorité explicitement définie par l'utilisateur (utilisation impérative des parenthèses).
4. Parsing complet d'une expression textuelle avec création d'un graphe.

L'aspect interface peut également faire l'objet de différents niveaux de d'implémentation

1. Pré-stockage en dur d'un ensemble de fonctions indexées.
2. Lecture dans un fichier.
3. Mise en place d'une interface utilisateur (ex. widgets Qt).

3.2 Affichage 3D

L'affichage 3D se réalisera à l'aide de la librairie OpenGL. La manipulation à la souris de la visualisation pourra se faire avec l'aide de la librairie *glut*, ou directement sout *Qt*. Un programme minimaliste de manipulation de scène 3D vous est donné en tant que point de départ.

Le travail principal consiste à discrétiser uniformément la surface, et à réaliser son affichage à l'aide de triangles ou de quadrangles.

Dans la boucle d'affichage principale réactualisée à vitesse interactive, l'appel explicite à l'affichage d'un triangle se réalise suivant la syntaxe :

```
double x1,y1,... , z3; <- valeurs des coordonnées du triangle

glBegin(GL_TRIANGLES);//debut affichage triangle
glVertex3d(x1,y1,z1); //sommet 1
glVertex3d(x2,y2,z2); //sommet 2
glVertex3d(x3,y3,z3); //sommet 3
glEnd();
```

De manière similaire, on peut afficher un quadrangle directement :

```
glBegin(GL_QUADS);//debut affichage triangle
glVertex3d(x1,y1,z1); //sommet 1
glVertex3d(x2,y2,z2); //sommet 2
glVertex3d(x3,y3,z3); //sommet 3
glVertex3d(x4,y4,z4); //sommet 4
glEnd();
```

Pour obtenir un calcul d'éclairage satisfaisant, il est nécessaire fournir également l'information de normale à la carte graphique.

Le cas d'un triangle dont la normale est constante sur la portion de plan est obtenu par l'appel :

```
glBegin(GL_TRIANGLES);//debut affichage triangle
glNormal3d(nx,ny,nz); //normale du triangle
glVertex3d(x1,y1,z1); //sommet 1
glVertex3d(x2,y2,z2); //sommet 2
glVertex3d(x3,y3,z3); //sommet 3
glEnd();
```

Lorsque plusieurs triangles sont affichés cotes à cotes (maillage), on cherche à donner l'apparence d'une surface globalement lisse. Les normales ne sont alors plus considérées comme constante sur un patch linéaire élémentaire (triangle ou quad), mais comme une donnée variable associée à chaque sommet. Les sommets sont alors vus comme l'échantillonnage d'une fonction lisse dont l'information de normale est connue en ces points particuliers.

Un triangle dont les sommets sont associés à des normales différentes possède la syntaxe suivante :

```
glBegin(GL_TRIANGLES);//debut affichage triangle
glNormal3d(nx1,ny1,nz1); //normale du triangle 1
glVertex3d(x1,y1,z1); //sommet 1
glNormal3d(nx2,ny2,nz2); //normale du triangle 2
glVertex3d(x2,y2,z2); //sommet 2
glNormal3d(nx3,ny3,nz3); //normale du triangle 3
glVertex3d(x3,y3,z3); //sommet 3
glEnd();
```

Finalement, l'affichage d'un ensemble de triangles dont les normales sont définies par sommet suivra la syntaxe suivante :

```
glBegin(GL_TRIANGLES);
for(int k_triangle=0;k_triangle<N_triangle;++k_triangle)
{
  //appels à glNormal3d et glVertex3d
  // des triangles correspondants
  // ...
}
glEnd();
```

Un ensemble de surfaces réalisées par ce type d'approche est montré en fig. 2.

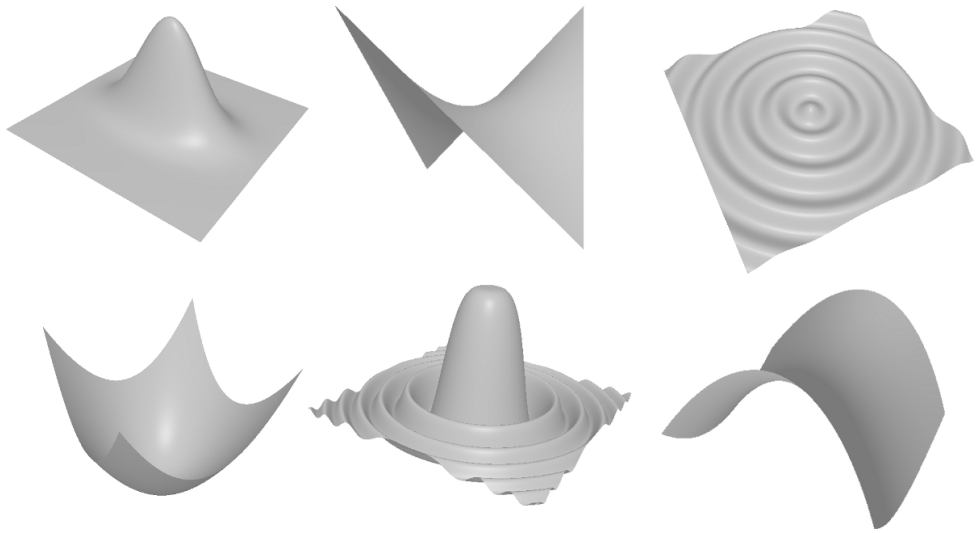

FIGURE 2 – Surfaces de type $z = f(x, y)$ visualisés et mises à jour interactivement.

4 Travail demandé

Vous implémenterez les deux étapes du projet, à savoir l'évaluation d'une fonction mathématique ainsi que son affichage 3D.

Il est indispensable d'être en mesure de réaliser l'affichage 3D correct de votre surface. Il sera notamment nécessaire de bien réfléchir à la structure de donnée permettant de modéliser de manière discrète votre surface. De plus, vous détaillerez le principe suivi pour le calcul de vos normales.

Une fois cette étape réalisée, vous pourrez améliorer l'étape d'analyse et de parsing de la formule textuelle avec la mise en place du graphe d'évaluation (parcours et construction de celui ci), ainsi que mettre en place des outils supplémentaires.

5 Amélioration possible

De nombreuses caractéristiques supplémentaires peuvent être affichées en lien avec une surface. Similairement à ce que proposent les logiciels classiques de visualisation, on pourra réaliser un ensemble d'amélioration.

De manière non exhaustive, on pourra noter quelques cas particuliers.

5.1 Domaine de définition et échantillonnage

On pourra s'intéresser à pouvoir faire varier de manière interactive le domaine de définition de la paramétrisation ainsi que son échantillonnage pour obtenir la surface souhaitée.

5.2 Surfaces paramétriques

On pourra trivialement étendre ce travail aux cas des surfaces paramétriques quelconques, c'est à dire aux fonctions

$$\mathbf{f} : \left\{ \begin{array}{lll} \mathbb{R}^2 & \rightarrow & \mathbb{R}^3 \\ (u,v) & \mapsto & \mathbf{f}(u,v) = (x(u,v), y(u,v), z(u,v)) \, . \end{array} \right. \tag{2}$$

Pour cela, il suffira de considérer deux fonctions supplémentaires pour x et y.

5.3 Couleurs et textures

OpenGL permet de gérer aisément les informations de couleurs et de textures par sommet. On pourra utiliser celle-ci pour améliorer la visualisation de la surface (couleur dépendante de coordonnées/normales, ou donnée en tant que fonction elle-même).

5.4 Animation

L'ajout d'un paramètre temporel t permet de modéliser une surface animée. Il suffira pour cela de pré-stocker ou de calculer à la volée la nouvelle surface puis de l'afficher pour t évoluant.

5.5 Affichage rapide

Le principe d'affichage présenté ici est appelé *mode immédiat* d'OpenGL par l'appel explicite à chaque coordonnée de l'ensemble des triangles ou des quadrangles. C'est un mode d'affichage lent qui n'est plus maintenu pour les versions d'OpenGL avancée.

On pourra se documenter et mettre en place des modes d'affichages plus efficaces (*vertex array*, ou *VBO*).

5.6 Normales analytiques

L'évaluation des normales basée sur une évaluation discrète n'est pas exacte aux sommets considérés. On pourra vérifier expérimentalement sur le cas d'une sphère paramétrique par exemple. Dans un premier temps, on pourra demander à l'utilisateur de fournir les dérivées de la fonction en tant qu'entrée textuelle similairement à la fonction elle-même. On comparera alors le rendu visuel entre l'approche exacte et discrète.

Étant donné la connaissance exacte de la fonction continue sous-jacente, il est cependant possible d'exprimer les dérivées exactes de cette surface. On pourra tenter de réfléchir à la mise en place d'un évaluateur automatique de dérivée basé sur les règles de dérivation, et évitant ainsi à l'utilisateur de définir lui même celle-ci en tant qu'entrée.

5.7 Rendu par ray-tracing

Une fois la surface calculée, on pourra réaliser un export de celle-ci sous un format textuel compatible avec des moteurs de rendu de haute qualitée. On pourra s'intéresser au cas de *PovRay*, ou *Yafaray*. On pourra ainsi réaliser des images ou des animations de surfaces en utilisant des effets de rendus complexes. Un exemple de tel rendu est illustré en fig. 3.

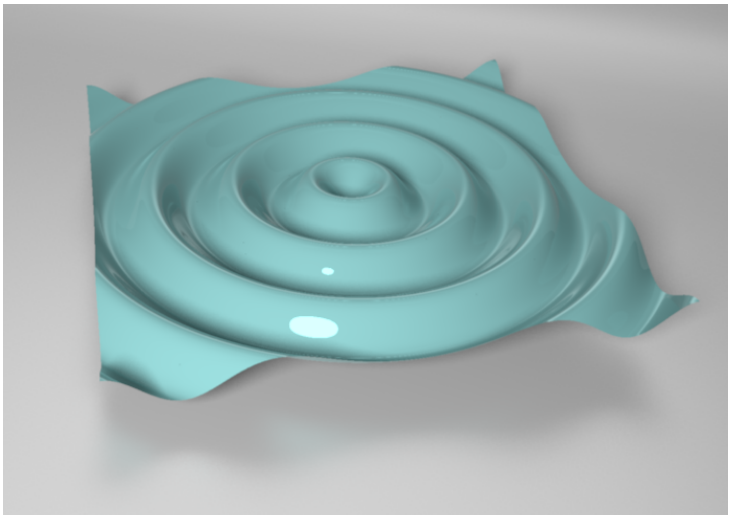

FIGURE 3 – Exemple d'une surface issue du projet exportée dans un format compatible avec un moteur de rendu par *ray-tracing*. Le rendu n'est plus temps-réel.