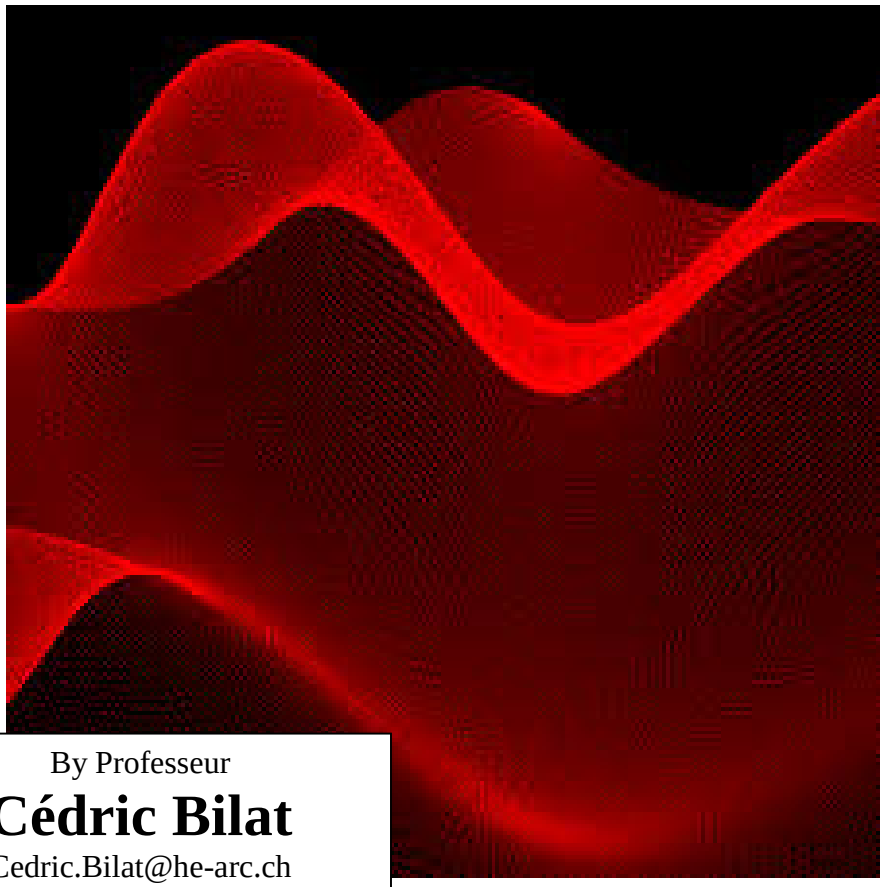
Problèmes Parallelisation



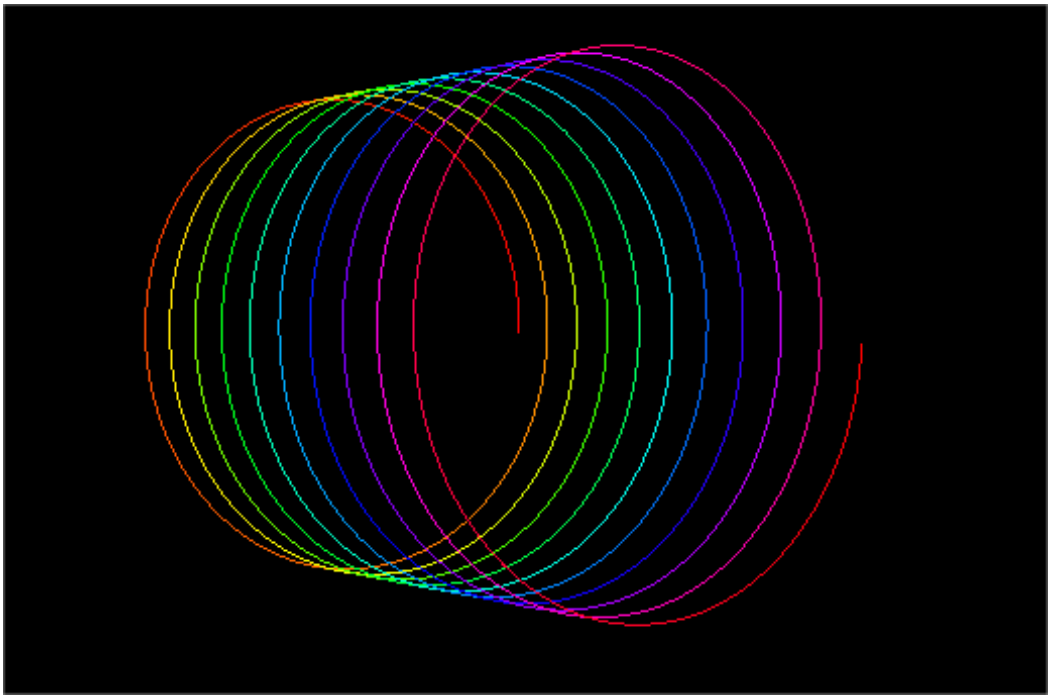
By Professeur
Cédric Bilat
Cedric.Bilat@he-arc.ch

OpenGL & Cuda

Ressort Animé

Contexte :

On s'intéresse à la courbe ci-dessous définit par une suite de points $v_i \in \mathbb{R}^3$.



Il s'agit d'un ressort animé qui se compresse et se décompresse au cours du temps t .

Math :

Soit :

- h la hauteur du ressort.
- $nbTour$ le nombre de spirales.
- r le rayon des spirales.
- n le nombre de sommets formant le ressort.
- t le paramètre d'animation.

Géométrie :

$$v_i = \begin{pmatrix} x_i \\ y_i \\ z_i \end{pmatrix} = \begin{pmatrix} r \cdot \cos\theta_i \\ r \cdot \sin\theta_i \\ i \cdot h \cdot \sin(t)/n \end{pmatrix} \begin{matrix} \in [-r, r] \\ \in [-r, r] \\ \in [-h, h] \end{matrix} \qquad i \in [0, n], t \in [0, \pi]$$

avec $\begin{cases} \theta_i(s_i) = s_i \cdot 2\pi \\ s_i = i \cdot \dfrac{nbTour}{n} \end{cases}$

Couleur :

La couleur c_i du sommet $v_i \in \mathbb{R}^3$ sera donné en HSB, par exemple avec

$$c_i = \begin{pmatrix} h_i \\ s_i \\ b_i \end{pmatrix} = \begin{pmatrix} i/n \\ 1 \\ 1 \end{pmatrix} \in [0,1]^3 \subset \mathbb{R}^3 \qquad i \in [0, n]$$

Comme $i \in [0, n]$, on n'a bien $h_i \in [0,1]$.

Note:

Plus n est grand, plus les spirales du ressort seront bien rondes.
L'animation se fait sur la 3-ième composante z_i.
Si on prend $h = r = 1$, tout se passe dans un box de cote 2.

Implémentation OMP

Le rendu sera effectué en OpenGL à l'aide de deux VBOs :

- Un VBO pour les sommets $v_i \in \mathbb{R}^3$
- Un VBO pour les couleurs des $v_i \in \mathbb{R}^3$.

Attention :

Ici on n'utilise pas l'API Image donné au cours, mais on part "from scratch" pour le rendu OpenGL.

Indication :

Vous trouverez une formation OpenGL accéléré dans le document *Bilat_CudaPracticalGuide*.

Implémentation Cuda

Le principe est le *même* que pour l'implémentation OMP, mais on utilisera ici en plus l'**interopérabilité** entre *Cuda* et *OpenGL*.

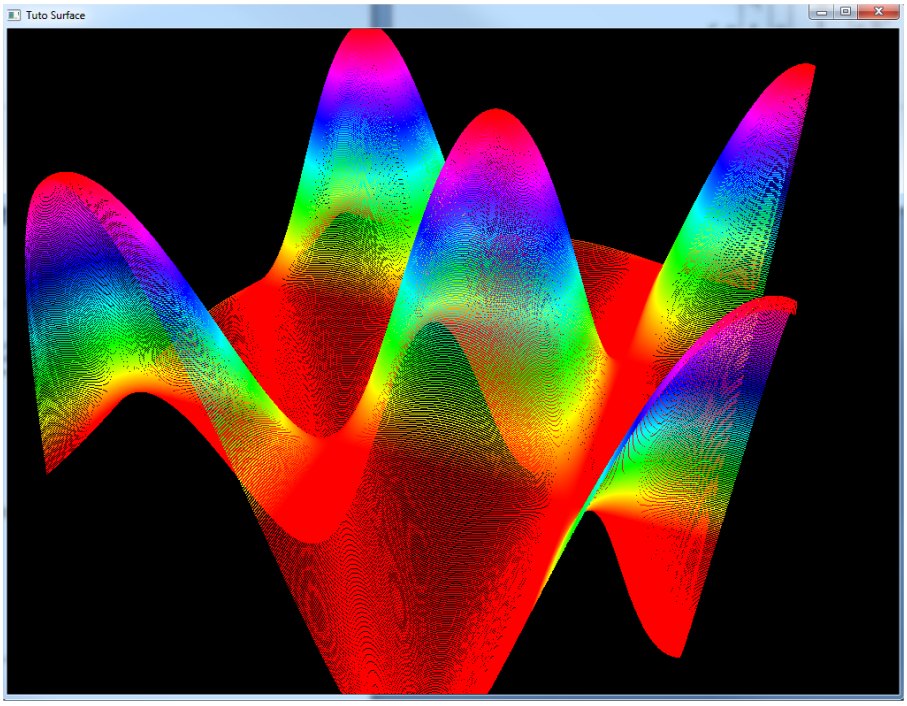
Indication :

Vous trouverez une formation accéléré « Interopérabilité entre *Cuda* et *OpenGL* » dans le document *Bilat_CudaPracticalGuide*.

Surface Animée

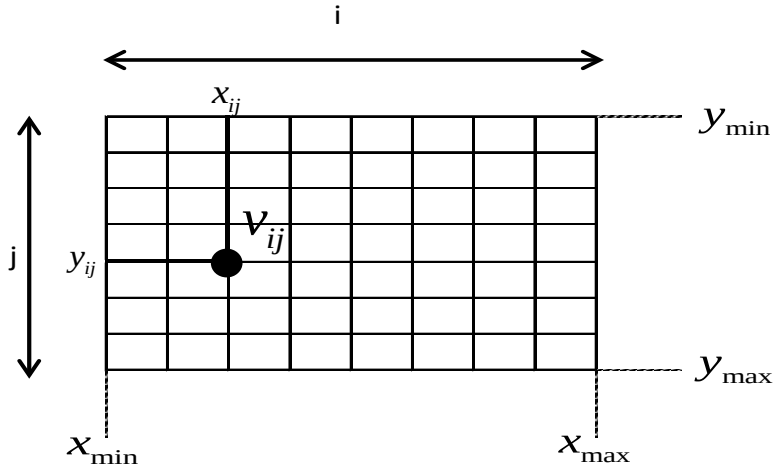
Contexte :

On s'intéresse à la surface ci-dessous défini à l'aide d'une suite de points $v_{ij} = \begin{pmatrix} x_{ij} \\ y_{ij} \\ z_{ij} \end{pmatrix} \in \mathbb{R}^3$:



TODO Faux

Notons que si on ne tiens pas compte de la 3ème composante, les sommets projetée sur le sol $\begin{pmatrix} x_{ij} \\ y_{ij} \end{pmatrix} \in \mathbb{R}^2$ de cette surface sont répartie uniformément sur une grille 2D.



L'animation va influencé seulement la 3ème composante z_{ij} au cours du temps t.

Math :

Soit

- n_x est le nombre de sommets sur l'axe x.
- n_y est le nombre de sommets sur l'axe y.
- n est la période des sinusoïdales.
- t est le paramètre d'animation.

Géométrie :

Soit $i \in [0, n_x], j \in [0, n_y]$ les indices parcourant la grille 2D homogène. On a alors

$$v_{ij} = \begin{pmatrix} x_{ij} \\ y_{ij} \\ z_{ij} \end{pmatrix} = \begin{pmatrix} \frac{i}{n_x} \\ \frac{j}{n_y} \\ \sin\left(x_{ij} \cdot 2\pi n + t\right) \bullet \cos\left(y_{ij} \cdot 2\pi n + t\right) \end{pmatrix} \begin{matrix} \in [0,1] \\ \in [0,1] \\ \in [-1,1] \end{matrix} \qquad t \in [0,1]$$

Essayé par exemple avec $n = 1$ puis $n = 4$

Couleur :

Posons c_{ij} la couleur du sommet v_{ij}. Définissons cette couleur en HSB, par exemple en donnant une value de Hue $\in [0,1]$ en relation avec la hauteur $z_{ij} \in \mathbb{R}$ du sommet. Comme $z_{ij} \in [-1,1]$, on peut prendre le mapping quasi direct suivant :

$$c_{ij} = \begin{pmatrix} h_{ij} \\ s_{ij} \\ b_{ij} \end{pmatrix} = \begin{pmatrix} \frac{1}{2} z_{ij} + \frac{1}{2} \\ 1 \\ 1 \end{pmatrix}$$

Implémentation OMP

Le rendu sera effectué en OpenGL à l'aide de deux VBOs :

- Un VBO pour les sommets $v_{ij} \in \mathbb{R}^3$
- Un VBO pour les couleurs des $v_{ij} \in \mathbb{R}^3$.

Pour ce simplifier la vie, on effectuera un rendu en **mode points**. En effet, contrairement au mode surface, il ne sera ainsi pas nécessaire avec cette simplification de trianguler la surface et de calculer les normales.

Attention :

Ici on n'utilise pas l'API Image donné au cours, mais on part "from scratch" pour le rendu OpenGL.

Indication :

Vous trouverez une formation OpenGL accéléré dans le document *BilatCudaPracticalGuide*.

Implémentation Cuda

Le principe est le même que pour l'implémentation OMP, mais on utilisera ici en plus l'**interopérabilité** entre *Cuda* et *OpenGL*.

Indication :

Vous trouverez une formation accélérée « interopérabilité entre *Cuda* et *OpenGL* » dans le document *Bilat_CudaPracticalGuide*.

End