TP3
Affichage et morphing d'images

Dans ce TDe, nous allons effectuer quelques opérations simples d'affichage avec la librairie OpenGL puis nous programmerons une application de morphing d'image.

1 Exercice 1 — Affichage

Téléchargez puis décompressez le fichier :

`http://perception.inrialpes.fr/people/Boyer/Teaching/Master/TP3.tar.gz`

La compilation s'effectuera à l'aide du Makefile présent dans le repertoire et de la commande make.

Le programme visupm.c permet d'afficher des images au format PGM et PPM et fait appel à la librairie graphique OpenGL pour l'affichage et la gestion des fenêtres. Nous ne nous intéressons dans le cadre de ce TDe qu'à quelques fonctionnalités de base de la librairie. En particulier :

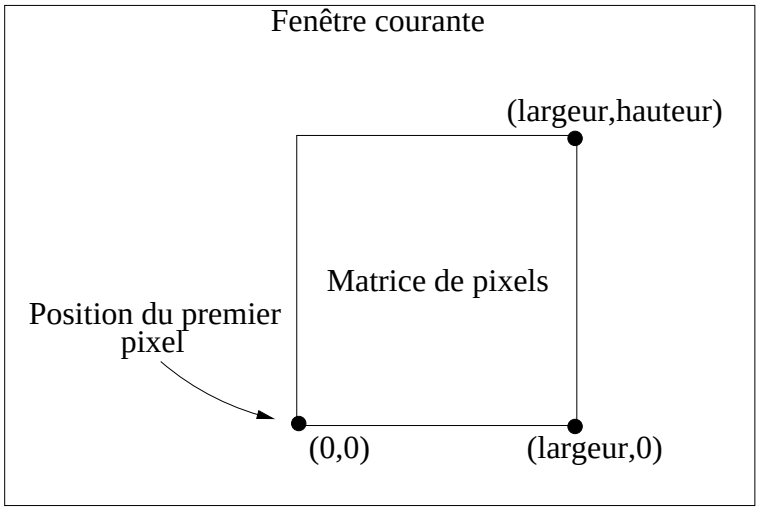

Figure 1: L'affichage d'une image par la fonction glDrawPixels().

Antoine.Letouzey@inrialpes.fr, Edmond.Boyer@imag.fr

- la fonction :
  glDrawPixels(ImgWidth, ImgHeight, ImgFormat, PixelType, Image)
  ImgWidth : largeur de l'image,
  ImgHeight : hauteur de l'image,
  ImgFormat : format de l'image (GL_RGB dans notre cas),
  PixelType: format des pixels (unsigned byte dans notre cas),
  Image : matrice de pixels au format PixelType,
  qui permet d'afficher une matrice de pixels dans la fenêtre courante ;

- la fonction :
  glRasterPos2i(xpos, ypos)
  qui permet de définir la position $(xpos, ypos)$ du premier pixel dans l'espace.

1. À l'aide de ce qui a été fait précédemment et en sachant que les pixels sont représentés, dans la matrice Image, par trois bytes successifs correspondant aux trois niveaux de couleurs rouge, vert et bleu, complétez la fonction LoadPMImage du programme c.

2. Modifiez le programme de sorte que l'image affichée puisse être deplacée à l'aide de la souris.

2 Exercice 2

Tracez un cadre de couleur autour de l'image, cherchez pour cela la fonction OpenGL correspondante en utilisant la documentation OpenGL donnée en lien www sur la page de l'enseignement.

3 Exercice 3 — Morphing

Le morphing que nous allons réaliser est basé sur une interpolation pixel à pixel des couleurs de l'image, indépendament pour chaque canal, rouge, vert ou bleu.

Soit λ un réel prenant des valeurs entre 0 et 1. Le pas ν entre deux valeurs consécutives de λ va déterminer le nombre d'images intermédiaires N du morphing (par exemple, ν=0,1 donne 9 images intermédiaires).

Pour une valeur de λ donnée, l'image intermédiaire I_λ est formée par pondération des pixels des deux images extrêmes I_0 et I_1 (les données du programme). Si l'on interprète comme précédemment une image comme une matrice:

$$I_\lambda = (1 - \lambda).I_0 + \lambda.I_1.$$

Cette interpolation doit être effectuée indépendament pour chaque pixel et pour chaque composante de la couleur, c'est à dire $N \times W \times H \times 3$, où (W, H) est la taille des images.

Antoine.Letouzey@inrialpes.fr, Edmond.Boyer@imag.fr

Complétez le programme `morph.c` fourni. La valeur de λ (variable `L`) est incrémentée respectivement décrementée par les touches + et -, ce qui provoque de plus la mise à jour de l'affichage de l'image correspondante.

4 Améliorer le morphing

Le principe de morphing décrit précédement suppose que les images sources sont parfaitement alignées (par ex., si deux visages sont considérés, le résultat sera "bon" si ces derniers ont même taille et position dans les images) ce qui n'est pas toujours le cas.

La zone de déformation. Une amélioration possible est de définir une zone de correspondance entre les deux images, c'est à dire une zone qui va définir quelle partie de l'image se transforme en quelle partie de l'autre image.

Pour ce faire, on indique trois points dans chaque image, a_0, b_0 et c_0 dans la première et a_1, b_1 et c_1 dans la seconde. Les deux triangles ainsi définis doivent constituer des zones se correspondant. Par exemple, dans le cas de deux visages, chacun des triangles doit grossièrement contenir le visage dans l'image correspondante.

Pour chaque image intermédiaire du morphing I_λ, la zone définie dans les deux images sources va avoir une position donnée par ses trois sommets a_λ, b_λ et c_λ. La position de ces sommets est calculée par interpolation linéaire des sommets sources comme (par exemple) :

$$a_\lambda = (1 - \lambda).a_0 + \lambda.a_1$$

idem pour b_λ et c_λ.

Copiez la fonction `LoadPMImage(...)` dans `morph_geometric.c`, compilez et testez le programme. Avec les touches + et -, vous observez un triangle noir se déplaçant sur l'interpolation précédente des deux images. Ce triangle représente la déformation géométrique entre les deux visages. Le but de la question suivante est de remplir ce triangle.

Remplissage du triangle. Pour remplir le triangle, il faut y copier de la texture provenant des images sources. Ces deux textures doivent être déformées puis combinées (voir figure 2).

La déformation de chaque texture s'effectue à l'aide d'une transformation affine construite à l'aide de 6 points ou 3 correspondances de points. Cette transformation affine est représentée par une matrice 2×3.

Les coordonnées non entières seront interpolées par le pixel le plus proche (=(int)(x+0.5)).

Complétez le programme `morph_geometric.c` à l'aide des fonctions définies dans `morph_geometric_util.h`. Seule la fonction `warping(...)` doit être

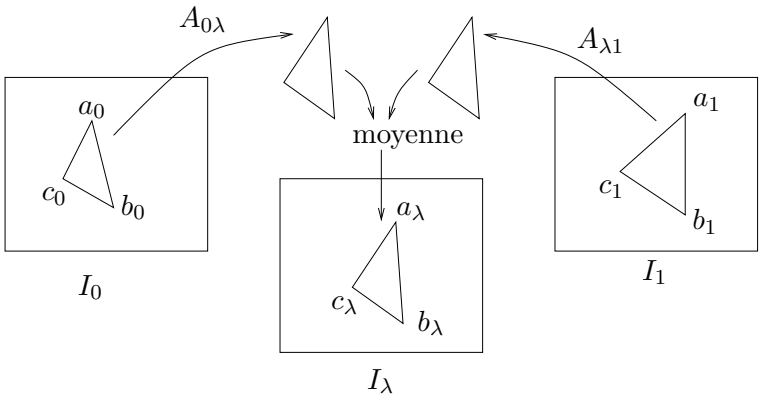

Figure 2: Calcul de la texture du triangle intermédiaire.

complétez. Elle consiste en un parcours de l'image intermédiaire. Lorsque le point courant `dL` est à l'extérieur du triangle intermédiaire `(aL,bL,cL)`, l'interpolation précédente est effectuée. C'est ce que vous avez observé. Lorsque `dL` est à l'intérieur, il faut retrouver le pixel de chaque image source 0 et 1 lui correspondant. Pour ce faire, on utilise les affinitées `A_L0` et `A_L1` déjà calculées qui vont de l'image intermédiaire aux images 0 et 1 respectivement.

L'application de, par exemple, `A_L0` au point `dL` donne le point correspondant sur l'image 0.

Pour aller plus loin... L'algorithme précédent suppose que la géométrie de la scène peut être représentée par un triangle, autrement dit que la scène est plane, ce qui n'est pas vrai en général. Utilisez un ensemble de triangles pour représenter les transformations géométriques entre les deux images.