Algorithmique et Programmation

—

Examen sur machine

—

G1: monasse*(at)*imagine.enpc.fr G2: boulc-ha*(at)*imagine.enpc.fr
G3: salauny*(at)*imagine.enpc.fr G4: arnaud.carayol*(at)*univ-mlv.fr
G5: de-la-gm*(at)*imagine.enpc.fr G6: vialette*(at)*univ-mlv.fr

—

13/12/13

—

1 Enoncé

1.1 Image projetée sur une sphère

On se propose de visualiser une image projetée sur une sphère, comme sur la figure 1. L'image est considérée comme initialement dans le plan $z = 0$ de l'espace à trois dimensions. On considère un centre C d'altitude négative $z_C < 0$. On considère la sphère S de centre C et passant par un milieu d'un bord de l'image. On fait la projection de centre C des points de l'image sur la partie $S^+ = S \cap \{(x, y, z) : z \geq 0\}$. On visualise les points en (x, y).

En réalité, pour construire l'image résultat, on fait la démarche inverse : pour chaque (x, y), on calcule le z (s'il existe) tel que $P = (x, y, z) \in S^+$, puis on intersecte le rayon (CP) avec le plan $z = 0$. Si ce point Q est dans l'image, on prend la couleur en Q et on la met au point (x, y) de l'image résultat.

1.2 Travail demandé

Il est plus important de livrer un code clair et qui compile sans warning, même s'il ne répond pas à toutes les questions. Pour cela, vérifiez à chaque étape que votre programme compile et se lance correctement.

1. Créez un projet Imagine++ avec un fichier contenant le `main`. Prenez le `CMakeLists.txt` d'un projet existant ou celui de `[IMAGINEPP_ROOT]/test/Graphics/` et adaptez-le [1]. Vous aurez aussi besoin d'un fichier image de votre choix que vous copierez dans le même dossier que votre programme, par exemple `forty-licks.png` dans ce dossier.

1. `[IMAGINEPP_ROOT]` est le chemin où Imagine++ est installé, typiquement `/usr/share/Imagine++` sous Windows et `C:/Program Files (x86)/Imagine++` sous Windows

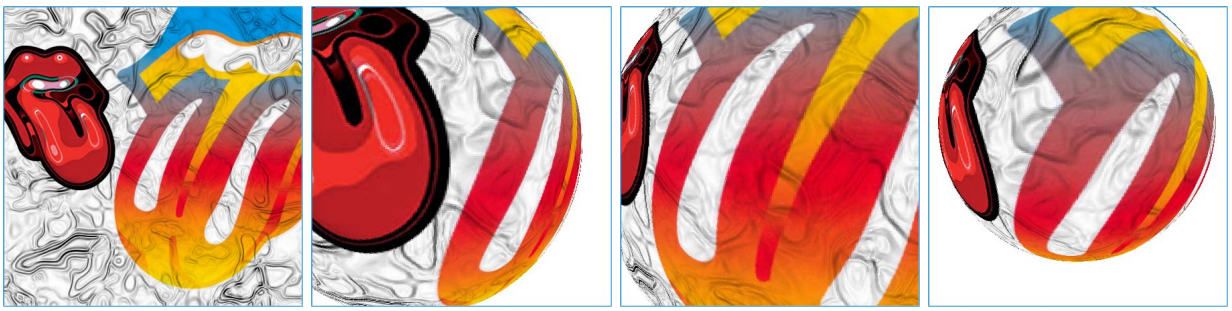

FIGURE 1 – Différentes vues d'une image projetée sur la sphère

2. Dans le `main`, charger l'image en couleur avec la fonction `loadColorImage` et l'afficher.
3. On travaillera avec des images carrées (largeur=hauteur). Dans des fichiers séparés, écrire une structure `imcarree` contenant les pixels et la taille du coté.
4. Ecrire une fonction `decoupe` qui renvoie une `imcarree` dont le coté est la plus petite dimension d'une image passée en argument et prend la sous-image de même origine haut-gauche.
5. Découper l'image initiale dans le main et l'afficher. Par la suite, on ne travaillera plus que sur cette image carrée, dont nous notons w le coté.
6. Dans des fichiers séparés, écrire une structure `vect` définissant un point ou vecteur en 3D.
7. Programmer les opérateurs d'addition, soustraction et multiplication par un scalaire de `vect`.
8. Implémenter la fonction `norme` calculant la norme euclidienne.
9. Placer un centre initial C en $(w/2, w/2, -w/2)$. On prend comme rayon de sphère la distance de C au point $(w/2, 0, 0)$. Ecrire une fonction `projette` qui procède comme expliqué dans l'énoncé :
   (a) en chaque pixel (x, y) de l'image à remplir, on calcule le z tel que $P = (x, y, z)$ soit dans S^+ (s'il n'existe pas, on met le pixel en blanc) ;
   (b) trouver le λ tel que $Q = C + \lambda(P - C)$ ait un $z_Q = 0$;
   (c) si ce point est dans l'image (mais coordonnées pas entières), mettre la couleur du pixel le plus proche.
10. Etendez l'interface graphique : un boucle d'attente de clic où le clic gauche change (x_C, y_C) et où le clic droit fait $z_C = -y_{\text{clic}}$; on reprojette l'image avec le nouveau centre C.

Important : Quand vous avez terminé, créez une archive du projet *à votre nom* en ZIP, RAR ou TGZ (évitez 7z). N'incluez que les fichiers source, pas les fichiers binaires créés par CMake. Ne vous déconnectez pas avant que le surveillant ne soit passé vous voir pour copier cette archive sur clé USB.