Algorithmique et Programmation

—

Examen sur machine

—

G1: monasse*(at)*imagine.enpc.fr G2: facciolg*(at)*imagine.enpc.fr
G3: salauny*(at)*imagine.enpc.fr G4: theophile.dalens*(at)*inria.fr
G5: vinyesm(at)imagine.enpc.fr G6: vialette*(at)*univ-mlv.fr

—

12/12/14

—

1 Enoncé

1.1 Profil de couleur d'image

On se propose de visualiser le profil des couleurs rouge-vert-bleu le long d'un segment défini par l'utilisateur dans une image. Avant de tracer tout nouveau graphique, on attendra un clic de l'utilisateur et on effacera le contenu de la fenêtre.

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.

1.2.1 Tracé de courbes

2. Ouvrez une fenêtre carrée de `taille` pixels dans chaque dimension (une constante que vous choisirez) et tracez la courbe $y = 10\sqrt{x}$ pour $x \in [0, 100]$ discrétisé de 1 en 1. On laissera `marge` pixels (une constante égale à 20) en blanc au bord de la fenêtre, zone dans laquelle on ne tracera jamais.
3. Ecrire une fonction `traceAxes` qui trace le cadre et les "tics" horizontaux et verticaux sur les axes.
4. Dans un fichier `curve.h`, définissez une structure `Curve` stockant deux tableaux `x` et `y` de nombres réels et leur taille commune `n`.
5. Implémenter des fonctions `newCurve` et `deleteCurve`. La première envoie une nouvelle `Curve` en allouant dans les tableaux un nombre d'éléments passé en paramètre, la deuxième libère la place mémoire allouée.
6. Écrire une fonction `regularx` prenant des valeurs min et max en paramétre et discrétisant les `x` d'une `Curve` couvrant régulièrement l'intervalle [`min`,`max`].
7. Demander à l'utilisateur un nombre de points et remplir une `Curve` avec ce nombre de points stockant les valeurs de la fonction $y = \sin x^2$, $x \in [-5, 5]$.
8. Définir une structure `doubleAffine` codant les 4 paramètres d'une fonction affine en dimension 2 $x' = a_x x + b_x$, $y' = a_y y + b_y$.

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

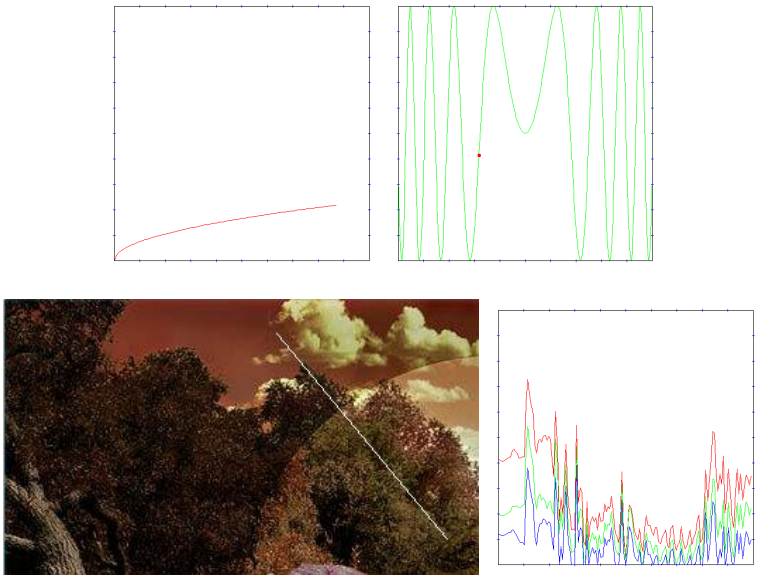

FIGURE 1 – Haut-gauche : tracé de $f(x) = 10\sqrt{x}$ après la question 3. Haut-droite : tracé de $f(x) = \sin x^2$ et sélection d'un point par l'utilisateur, question 13. Bas-gauche : image avec un segment blanc tracé par l'utilisateur, question 16. Bas-droite : profils de couleur le long de ce segment, question 19.

9. Implémenter une fonction `fill` prenant des intervalles [`xmin`, `xmax`] et [`ymin`,`ymax`] et renvoyant une `doubleAffine` appropriée pour couvrir toute la surface de la fenêtre moins les marges : (`xmin`,`ymin`) en bas à gauche et (`xmax`,`ymax`) en haut à droite de la zone d'affichage.
10. Écrire une fonction `fit` prenant une `Curve`, calculant ses bornes en x et y et renvoyant une `doubleAffine`. On codera et utilisera une fonction `minmax` renvoyant les bornes d'un tableau passé en argument.
11. Tracer la courbe de la question 7 en utilisant une fonction `drawCurve` à définir et prenant une `Curve`, une `doubleAffine` et une couleur.
12. Écrire une fonction `indexClick` qui retourne l'index du point d'échantillonnage d'une `Curve` le plus proche d'un point (x, y) cliqué. Pour cela, il faut transformer les points de la courbe par la `doubleAffine` servant à l'afficher.
13. Faire une boucle qui prend des clics de l'utilisateur et affiche le point le plus proche de la courbe tracée. Un clic droit sort de la boucle.

1.2.2 Profil

14. Charger en mémoire une image couleur.
15. Afficher l'image dans une nouvelle fenêtre. Pour gérer le multi-fenêtrage, il suffit de savoir que la fonction `openWindow` renvoie une variable de type `Window`, dont la structure est sans importance. On utilise la fonction `setActiveWindow` prenant une `Window` pour indiquer dans quelle fenêtre les affichages et détections de clics suivants ont lieu. La première fenêtre ouverte devient toujours la fenêtre active jusqu'à nouvel ordre.
16. Faire cliquer deux points à l'utilisateur dans la fenêtre de l'image et tracer le segment les reliant.
17. Suivant que le segment est plutôt horizontal ou vertical, on le discrétisera de 1 en 1 en abscisse ou ordonnée respectivement. Compter le nombre de points et définir des `Curves` `r`, `g` et `b` prenant les intensités rouge, vert et bleu de l'image le long du segment.
18. Remplir une `doubleAffine` appropriée pour afficher les courbes couvrant tous les points en abscisse et un intervalle [0,255] en ordonnée.
19. Tracer les courbes de `r`, `g` et `b` dans leur couleur respective sur le même graphique.

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.