Schmutz Lucas

Sciper : 217498

Décembre 2016

Structure du programme :

La fonction **main** est uniquement composée de déclarations de variables et des fonctions utiles à son fonctionnement. Le programme commence par lire les propriétés de l'image de sortie à l'aide d'une fonction et de pointeurs.
La fonction **erreur_sortie** teste les propriétés précédentes et appelle relativement à l'erreur, une des cinq fonctions d'affichage d'erreur fourni. On déclare ensuite quatre tableaux qui découlent des propriétés de sorties.
La fonction **read_img** appelle la fonction **infos_images** (qui enregistre les paramètres utiles de chacune des images dans un tableau à deux dimensions) et teste ensuite l'erreur de profondeur, le cas échéant, elle appelle la fonction d'affichage de l'erreur de profondeur. Après avoir formaté le tableau **img** à trois dimensions selon **nbL_s** et **nbC_s**, elle lit les entrées utilisateurs et défini si le pixel lu sera présent dans l'image de sortie, dans quel cas elle l'enregistre à sa position finale dans une grille de la taille de l'image de sortie.
On définit ensuite la nouvelle intensité maximale à l'aide de la fonction **def_new_max** qui utilise pour son calcul la fonction **ppmc** qui elle-même appelle la fonction **pgdc**.
Ensuite la fonction **tri _prof** permet d'éviter de trier chacun des pixels du tableau **img** en triant un tableau contenant une copie des profondeurs et en effectuant les mêmes opérations de tri sur un tableau **ordre** allant de 0 aux nombres d'images. Ceci réduit grandement le coût calcul en triant uniquement deux tableaux de tailles **nb_input** et non **nb_input * nbL_s * nbC_s** pixels. Il suffit d'aller consulter dans la fonction **composition** ; la position n du tableau **ordre** qui nous permet d'accéder à la profondeur associée.
La fonction **composition** nous permet ensuite de parcourir le tableau **img** dans ses trois dimensions et selon l'ordre des profondeurs. Elle enregistre ensuite dans le tableau **img_final** le pixel selon ses propriétés d'opacités ou d'image de fond.
La fonction **impression** parcourt ensuite le tableau **img_final** selon ses lignes et colonnes et affiche les pixels lus. Afin de ne pas dépasser 70 caractères par lignes, elle commence par calculer le nombre maximal de caractères que peut prendre un pixel. En utilisant la propriété de la fonction **printf** qui retourne le nombre de caractères qu'elle a affiché, et une variable d'accumulation, elle affichera un caractère de retour à la ligne et remmettra la variable d'accumulation à zéro si le prochain pixel peut potentiellement faire dépasser la limite de 70 caractères.

Complexité de l'algorithme de composition

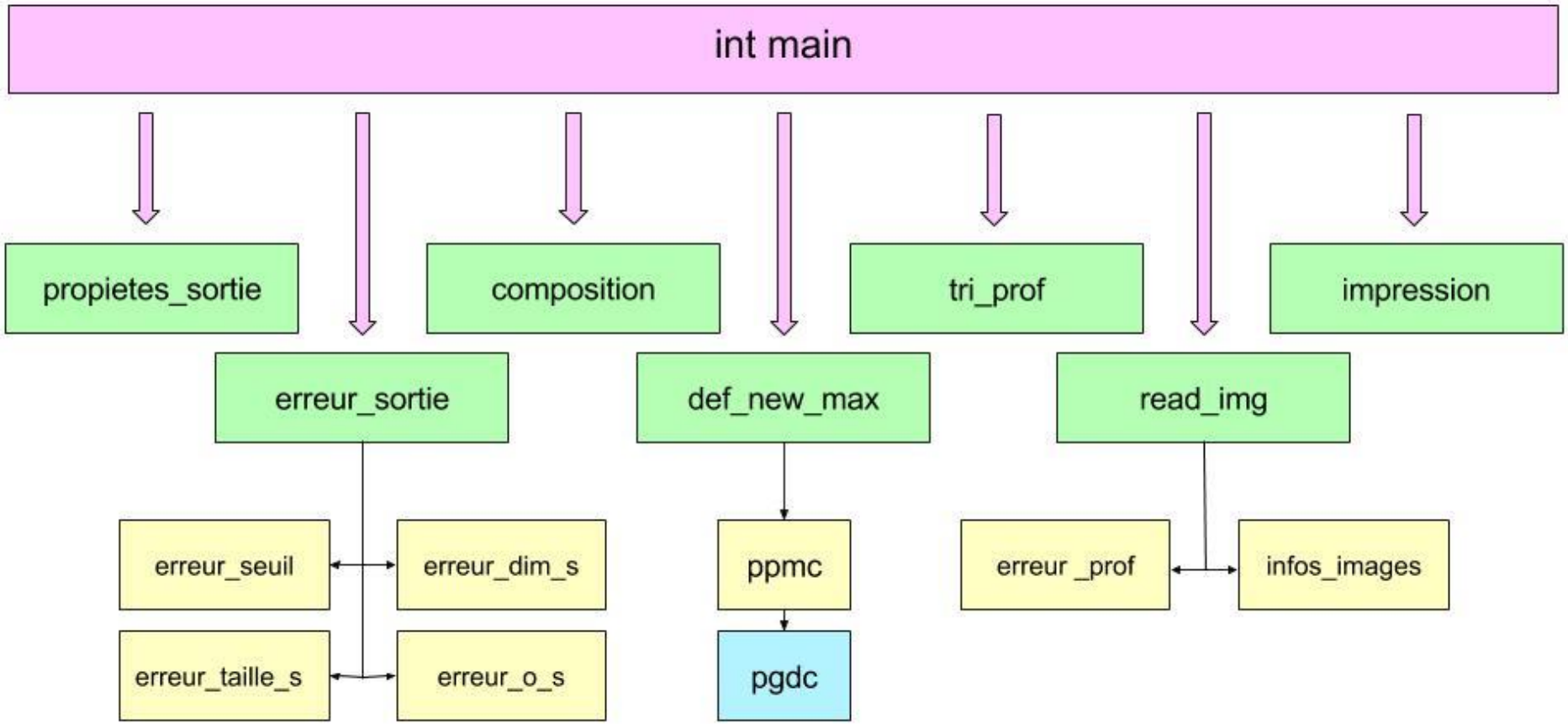
L'algorithme comprend trois boucles selon des paramètres différents et sa complexité est :

O(nbL_s * nbC_s * nb_input)

Composition 2.5D

Entrée : nbL_s, nbC_s, new_max, nb_input,
images[nb_input][nbL_s][nbC_s], image_finale[nbL_s][nbC_s], ordre[nb_input],

```
Pour l de 1 à nbL_s
  Pour c de 1 à nbC_s
    Pour n de 1 à nb_input
      pixel ← images[ordre[n]][l][c]
      intensité_relative ← pixel/intensité maximum (de l'image concernée)
      Si pixel n'est pas vide et intensité_relative < seuil
        Image_finale[l][c] ← new_max * intensité_relative
        Quitter la boucle n
      Sinon si image de fond
        Si pixel vide
          Image_finale ← new_max
       Sinon
         Image_finale ← new_max * intensité_relative
Sortir images_finale
```


int main
propietes_sortie
composition
tri_prof
impression
erreur_sortie
def_new_max
read_img
erreur_seuil
erreur_dim_s
erreur_taille_s
erreur_o_s
ppmc
pgdc
erreur _prof
infos_images