Imagerie 3D

Compte rendu TP1

12 mars 2015

Table des matières

1 Stockage de l'image **3**
- 1.1 Lecture de l'image 3
- 1.2 Accès à une valeur x,y,z 3
- 1.3 Définition d'une valeur 4

2 Application **4**

3 Volume rendering **4**
- 3.1 Algorithme MinIP 4
- 3.2 Algorithme MIP 5
- 3.3 Algorithme AIP 5
- 3.4 Résultats 5

1 Stockage de l'image

Méthode : Utilisation d'une classe Image. Stockage dans un tableau binaire linaire, puis lecture via des calculs sur les indexes.

Tableau utilisé : *VALUE* bin ;* Avec un *typedef unsigned short VALUE ;*

1.1 Lecture de l'image

On lit l'image entièrement et on la stock dans le tableau.

```
1  void load(char* filename) {
     FILE *f_image;
3
     if ((f_image = fopen(filename, "rb")) == NULL) {
5      printf("\nErreur lecture %s \n", filename);
       exit(EXIT_FAILURE);
7    }
     else {
9      if (fread((VALUE*)bin, sizeof(VALUE), getSize(), f_image) !=
         (size_t)(getSize()))
11     {
         printf("\nErrur lecture %s \n", filename);
13       exit(EXIT_FAILURE);
       }
15     fclose(f_image);
     }
17 }
```

1.2 Accès à une valeur x,y,z

```
1  // Pour recuperer la valeur finale
   VALUE getValue(int x, int y, int z) {
3    return convertValue(*getVoxel(x,y,z));
   }
5
   // Convertion double octet
7  VALUE convertValue(VALUE v) {
     int x1 = v/256;
9    int x2 = v - x1*256;
     return x2*256 + x1;
11 }

13 // Recuperer un poiteur vers le voxel
   VALUE* getVoxel(int x, int y, int z) {
15   return &bin[getIndexInTabBin(x,y,z)];
   }
17
```

```
// Calcul d'exploration
int getIndexInTabBin(int x, int y, int z) {
  int v = z*sizeX*sizeY + y*sizeX + x;
  return v;
}
```

1.3 Définition d'une valeur

```
void setValue(int x, int y, int z, VALUE val) {
  *getVoxel(x,y,z) = convertValue(val);
}
```

2 Application

On test la classe en récupérant des valeurs de certains pixels, ainsi que les valeurs maximums et minimums.

```
Image in(448,576,72);
char path[250] = "beaufix.448x576x72.0.6250x0.6250x1.4.img";
in.load(path);
cout << "minValue = " << in.getMinValue() << endl;
cout << "maxValue = " << in.getMaxValue() << endl;
cout << "value = " << in.getValue(200,200,20) << endl;
```

Et on retrouve parfaitement les valeurs spécifiés dans la feuille de TP.

— minValue = 0
— maxValue = 334
— l(200,200,20) = 14

3 Volume rendering

Pour calculer le volume rendering, on crée une nouvelle image, puis on écrit sur la même couche le résultat du calcul appliqué sur toutes les couches, en fonction de l'algorithme

3.1 Algorithme MinIP

```
for (int i = 0; i < in.sizeX; ++i)
  for (int j = 0; j < in.sizeY; ++j) {
    VALUE minVal = in.getValue(i,j,0);
    for (int k = 0; k < in.sizeZ; ++k) {
      minVal = min(in.getValue(i,j,k), minVal);
    }
    out.setValue(i,j,0,minVal);
  }
```

3.2 Algorithme MIP

```
for (int i = 0; i < in.sizeX; ++i)
  for (int j = 0; j < in.sizeY; ++j) {
    VALUE maxVal = in.getValue(i,j,0);
    for (int k = 0; k < in.sizeZ; ++k) {
      maxVal = max(in.getValue(i,j,k), maxVal);
    }
    out.setValue(i,j,0, maxVal);
  }
```

3.3 Algorithme AIP

```
for (int i = 0; i < in.sizeX; ++i)
  for (int j = 0; j < in.sizeY; ++j) {
    unsigned long long val = 0;
    for (int k = 0; k < in.sizeZ; ++k) {
      val += in.getValue(i,j,k);
    }
    out.setValue(i,j,0,val/in.sizeZ);
  }
```

3.4 Résultats