Compte rendu TP traiment d'images

LE PHILIPPE Noé

12 mars 2015

Algorithmes

Lecture du fichier

```
void RawReader::load(std::string path) {
  std::ifstream f (path.c_str(), ios::in | ios::binary);
  unsigned short c;
  //chaque image est representee sous la forme
  //d'un tableau a une dimension de la taille de l'image
  unsigned short *img = new unsigned short [sizeX*sizeY];
  int i = 0;
  while(!f.eof()) {
    f.read((char *)&c,sizeof(short));

    //inversion des bits
    short l = c % 256;
    short r = c / 256;

    img[i++] = l * 256 + r;
    //lorsque la totalitee de l'image est lue
    //elle est ajoutee au vector d'images
    if(i >= sizeX * sizeY) {
      i = 0;
      data.push_back(img);
      img = new unsigned short [sizeX*sizeY];
    }
  }
}
```

Accès à une valeur

```
/**
Les images sont stockee dans un std::vector
Pour avoir le pixel Ã  la couche k du fichier
on lit l'index k du std::vector
*/
unsigned short RawReader::getValue(int i, int j, int k) {
  return data[k][i * sizeY + j];
}
```

Valeur maximum de l'image

```
/**
Parcours de tous les pixels de toutes les images
*/
unsigned short RawReader::getMaxValue() {
  unsigned short maxi = 0;
  for (int i = 0; i < sizeX; ++i) {
    for (int j = 0; j < sizeY; ++j) {
      for (int k = 0; k < sizeZ; ++k) {
        maxi = max(getValue(i, j, k), maxi);
      }
    }
  }
  return maxi;
}
```

Volume Rendering

Il n'y aura que les résultat du volume rendering sur WHATISIT, il serait redondant d'insérerer les résultats pour tous les fichiers alors que les algorithmes sont les mêmes.

FIGURE 1 – Volume rendering : AIP

FIGURE 2 – Volume rendering : MIP

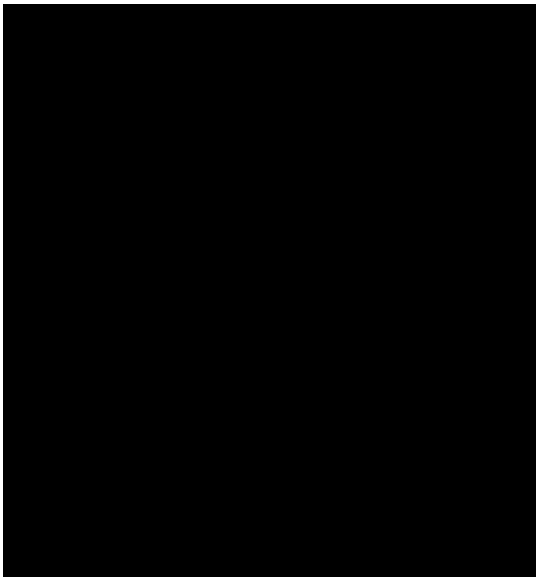

FIGURE 3 – Volume rendering : MINIP

Code source

```
#include <cstring>
#include <iostream>
#include <vector>
#include <fstream>
#include <stdlib.h>
#include <algorithm>

using namespace std;

class RawReader {
private:
        std::vector<unsigned short*> data;
        int sizeX;
        int sizeY;
        int sizeZ;
        void load(std::string path);

public:
        unsigned short getValue(int i, int j, int k);
        RawReader(std::string path, int sizeX, int sizeY,
            int sizeZ);
        unsigned short getMinValue();
        unsigned short getMaxValue();
        void aip(std::string path);
        void mip(std::string path);
        void minip(std::string path);
};
```

```
RawReader::RawReader(std::string path, int sizeX, int sizeY,
     int sizeZ) {
        this->sizeX = sizeX;
        this->sizeY = sizeY;
        this->sizeZ = sizeZ;
        this->load(path);
}

unsigned short RawReader::getValue(int i, int j, int k) {
        return data[k][i * sizeY + j];
}

unsigned short RawReader::getMinValue() {
        unsigned short mini = 65000;
        for (int i = 0; i < sizeX; ++i) {
                for (int j = 0; j < sizeY; ++j) {
                        for (int k = 0; k < sizeZ; ++k) {
                                mini = min(getValue(i, j, k)
                                    , mini);
                        }
                }
        }
        return mini;
}

unsigned short RawReader::getMaxValue() {
        unsigned short maxi = 0;
        for (int i = 0; i < sizeX; ++i) {
                for (int j = 0; j < sizeY; ++j) {
                        for (int k = 0; k < sizeZ; ++k) {
                                maxi = max(getValue(i, j, k)
                                    , maxi);
                        }
                }
        }
        return maxi;
}

void RawReader::load(std::string path) {
        std::ifstream f (path.c_str(), ios::in | ios::binary
            );
        unsigned short c;
        unsigned short *img = new unsigned short [sizeX*
            sizeY];
        int i = 0;
        while(!f.eof()) {
                f.read((char *)&c,sizeof(short));

                short l = c % 256;
                short r = c / 256;

                img[i++] = l * 256 + r;
                if(i >= sizeX * sizeY) {
                        i = 0;
```

```
                        data.push_back(img);
                        img = new unsigned short [sizeX*
                            sizeY];
                }
        }
}

void RawReader::aip(std::string path) {
        std::ofstream f (path.c_str(), ios::out | ios::
            binary);
        unsigned long long * out = new unsigned long long [
            sizeX * sizeY];

        for (int j = 0; j < sizeZ; ++j) {
                for (int i = 0; i < sizeX * sizeY; ++i) {
                        out[i] += data[j][i];
                }
        }

        for (int i = 0; i < sizeX * sizeY; ++i) {
                out[i] = out[i] / sizeZ;
                f.write ((char*)&out[i], sizeof (unsigned
                    short));
        }
        delete(out);
        f.close();
}

void RawReader::mip(std::string path) {
        std::ofstream f (path.c_str(), ios::out | ios::
            binary);
        unsigned short * out = new unsigned short [sizeX *
            sizeY];

        for (int j = 0; j < sizeZ; ++j) {
                for (int i = 0; i < sizeX * sizeY; ++i) {
                        out[i] = std::max(data[j][i], out[i
                            ]);
                }
        }

        for (int i = 0; i < sizeX * sizeY; ++i) {
                out[i] = out[i] / sizeZ;
                f.write ((char*)&out[i], sizeof (unsigned
                    short));
        }
        delete(out);
        f.close();
}

void RawReader::minip(std::string path) {
        std::ofstream f (path.c_str(), ios::out | ios::
            binary);
```

```
        unsigned short * out = new unsigned short [sizeX *
            sizeY];
        for (int i = 0; i < sizeX * sizeY; ++i) {
                out[i] = 65000;
        }

        for (int j = 0; j < sizeZ; ++j) {
                for (int i = 0; i < sizeX * sizeY; ++i) {
                        out[i] = std::min(data[j][i], out[i
                            ]);
                }
        }

        for (int i = 0; i < sizeX * sizeY; ++i) {
                out[i] = out[i] / sizeZ;
                f.write ((char*)&out[i], sizeof (unsigned
                    short));
        }
        delete(out);
        f.close();
}

int main() {
        RawReader *r = new RawReader("/home/noe/
            Téléchargements/WHATISIT/whatisit.301x324x56
            .1.1.1.4.img",
                301, 324, 56);
        std::cout << "min : " << r->getMinValue() << std::
            endl;
        std::cout << "max : " << r->getMaxValue() << std::
            endl;

        std::cout << "at 200 200 20 : " << r->getValue(200,
            200, 20) << std::endl;
        r->aip("aip.0.raw");
        r->mip("mip.0.raw");
        r->minip("minip.0.raw");
}
```