CR TP6 COMMANDRE

Benjamin COMMANDRE

13 Mars 2015

1 Programme c++ de base

1.1 Lecture d'une image au format brut

```
void lireImage(char* nomImg){

FILE* fichier;
long tailleFichier;
size_t result;
fichier = fopen(nomImg, "rb");

if(fichier){
    fseek (fichier, 0 , SEEK_END);
    tailleFichier = ftell (fichier);
    rewind (fichier);

    image = (unsigned short*) malloc (sizeof(unsigned short)*tailleFichier);

    if (image == NULL){
        std::cerr << "Taille du buffer nulle" << std::endl;
    }

    result = fread (image,1,tailleFichier,fichier);

    if (result != tailleFichier){
        std::cerr << "Erreur de lecture" << std::endl;
    }

}else{
    std::cerr << "Impossible d'ouvrir le fichier" << std::endl;
}

fclose(fichier);
}
```

1.2 Renvoi de la valeur d'un voxel

```
unsigned short getValue(int i, int j, int k){
        unsigned short save = (image[i*j*k]>>8) | (image[i*j*k]<<8);
        return save;
}
```

1.3 Affichage de la valeur minimale et maximale des voxels de l'image

```
void valeurs(){

        int min = 9999999;
        int max = 0;
        unsigned short save;

        for(int i =0; i<dimX*dimY*dimZ; i++){
                save = (image[i]>>8) | (image[i]<<8);
                if(save< min){
                        min = save ;
                }

                if(save > max){
                        max = save;
                }
        }

        std::cout << "Valeur minimale : " << min << std::endl;
        std::cout << "Valeur maximale : " << max << std::endl;
}
```

2 Volume Rendering

2.1 Programme

```
void volumeRendering(char* nomImg,char* nomSortie, int visuFlag){

FILE* fichier;
FILE* sortie;
size_t result;
long tailleFichier;
unsigned short save;
unsigned short* buffer;
unsigned short* bufferSortie;

fichier = fopen(nomImg, "rb");
sortie = fopen(nomSortie, "wb");

if(fichier){

        fseek (fichier, 0 , SEEK_END);
        tailleFichier = ftell (fichier);
        rewind (fichier);

buffer=(unsigned short*) malloc (sizeof(unsigned short)*tailleFichier);
bufferSortie=(unsigned short*) malloc (sizeof(unsigned short)*tailleFichier);

        if (buffer == NULL){
                std::cerr << "Taille du buffer nulle" << std::endl;
        }

        result = fread (buffer,1,tailleFichier,fichier);

        if (result != tailleFichier){
                std::cerr << "Erreur de lecture" << std::endl;
        }

if (visuFlag == 1){
        int max = 0;
        for(int i = 0; i < dimX*dimY; i++){
                max = 0;
                for(int z = 0; z < dimZ; z++){
                        save = (buffer[i + z*dimX*dimY]>>8) |
                                (buffer[i + z*dimX*dimY]<<8);
                        if (save > max ){
                                max = save;
```

```
                        }
                }
                save = (max>>8) | (max<<8);
                bufferSortie[i] = save;
        }
}else if (visuFlag == 2){

        int moy = 0;
        for(int i = 0; i < dimX*dimY; i++){
                moy = 0;
                for(int z = 0; z < dimZ; z++){
                        save = (buffer[i + z *dimX*dimY]>>8) |
                                (buffer[i + z*dimX*dimY]<<8);
                        moy = moy + save;
                }
                moy = moy / dimZ;
                save = (moy>>8) | (moy<<8);
                bufferSortie[i] = save;
        }
}else if (visuFlag == 3){
        int min = 999999;
        for(int i = 0; i < dimX*dimY; i++){
                min = 999999;
                for(int z = 0; z < dimZ; z++){
                        save = (buffer[i + z *dimX*dimY]>>8) |
                                (buffer[i + z*dimX*dimY]<<8);
                        if (save < min ){
                                min = save;
                        }
                }
                save = (min>>8) | (min<<8);
                bufferSortie[i] = save;
        }
}
result = fwrite (bufferSortie,1,tailleFichier,sortie);

if (result != tailleFichier){
        std::cerr << "Erreur d'ecriture" << std::endl;
}
}else{
        std::cerr << "Impossible d'ouvrir le fichier" << std::endl;
}

fclose(fichier);
fclose(sortie);
}
```

2.2 Resultats

Les images sont celles issues de l'image BEAUFIX :

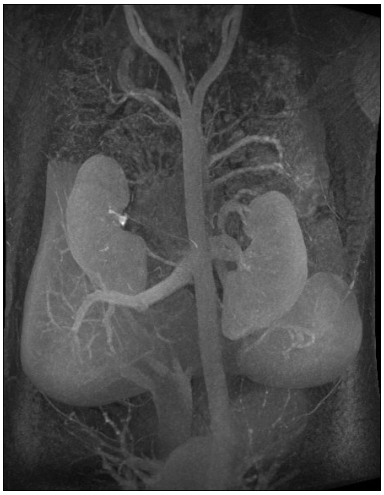

FIGURE 1 – Image de sortie de BEAUFIX avec la methode mip

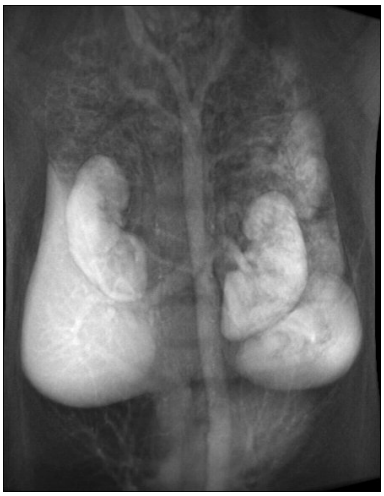

FIGURE 2 – Image de sortie de BEAUFIX avec la methode aip

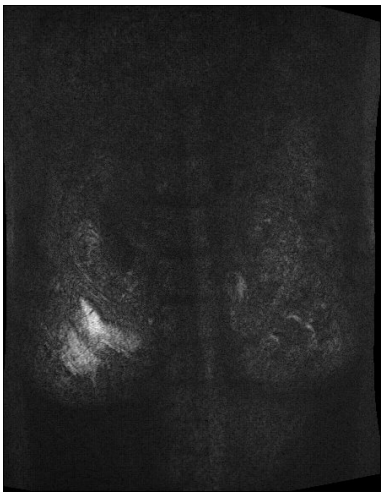

FIGURE 3 – Image de sortie de BEAUFIX avec la methode minip