L3 Info - Option Image UZAN Lucas

UFR IM2AG - UGA

TP5 - Visualisation de courbes/surfaces isovaleur

Partie 1 - Calcul de courbes isovaleur sur slice 3D

Dans la fonction dessin_courbe_iso_planZ(), après avoir construit le carré (avec les coordonnées et les valeurs de chaque point) ....

```
Point2D p0, p1, p2, p3;
double v0, v1, v2, v3;
p0.x=(float)(i-1);
p0.y=(float)(j-1);
v0=buffer_vol[Oz*sx*sy+(i-1)+(j-1)*sx];
// même méthode pour les 3 autres sommets
```

...on construit les 4 combinaisons de triangles a partir des sommets du carré :

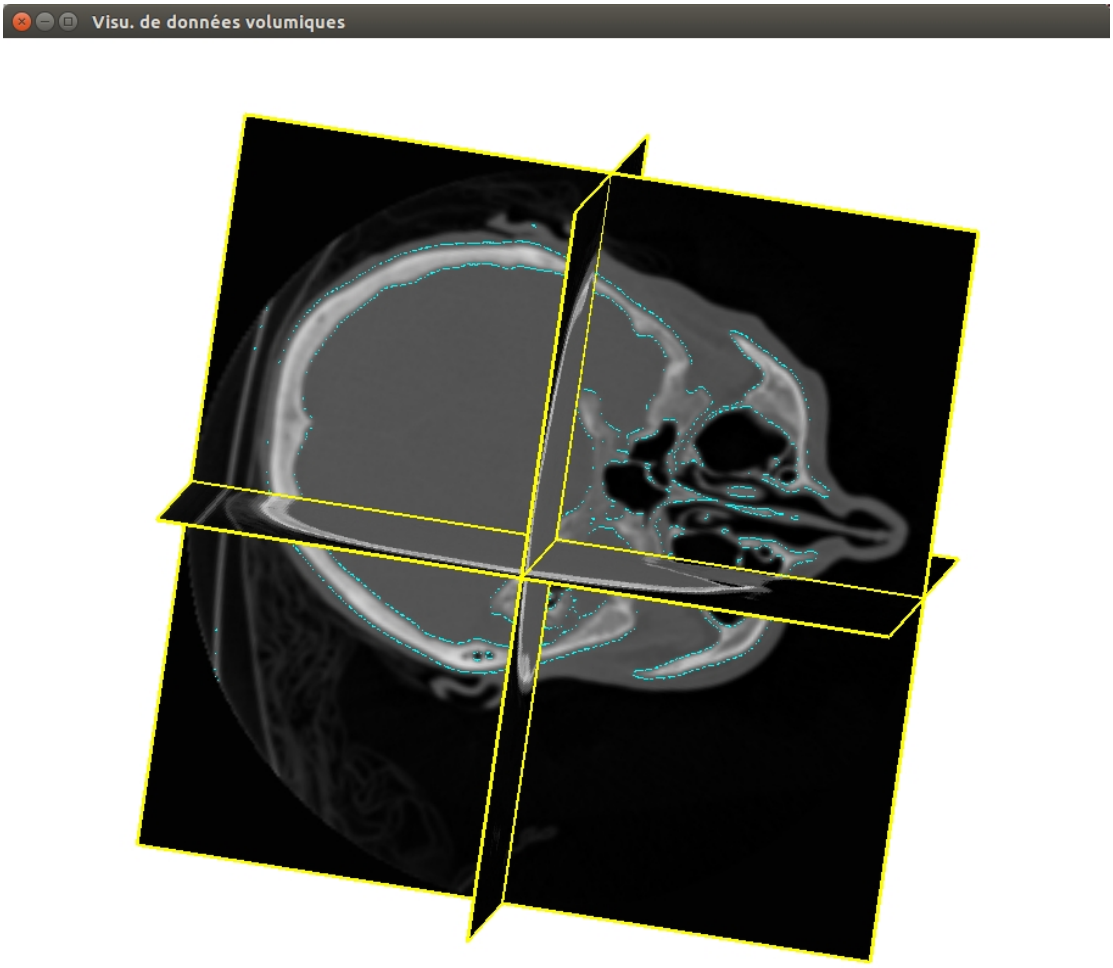
```
Triangle2D tria;
tria.p0 = p0;
tria.p1 = p1;
tria.p2 = p2;
tria.v0 = v0;
tria.v1 = v1;
tria.v2 = v2;
traitement_triangle(tria);
// et les 3 autres combinaisons par la suite...
```

Puis, dans traitement_triangle(), on rassemble les marques de chaque sommet du triangle afin de déterminer la courbe isovaleur et de la dessiner :

```
void traitement_triangle(Triangle2D T){
int m0 = marque(T.v0, isoVal);
int m1 = marque(T.v1, isoVal);
int m2 = marque(T.v2, isoVal);
double point[3];
glLineWidth(epaisseur_courbes_isovaleur); // épaisseur
```

```
glColor3f(0.0,1.0,1.0); // couleur cyan
glBegin(GL_LINES);
if(m0 != m2 && m0 != m2)
{
        intersection_arete_plan(point, T.p0, T.p1, T.v0, T.v1, isoVal);
        intersection_arete_plan(point, T.p0, T.p2, T.v0, T.v2, isoVal);
        glVertex3f(point[0], point[1], point[2]);
}
// suivi des 2 autres conditions ..
```

Le résultat correspond a nos attentes :



Partie 2 - Calcul de surfaces isovaleur

Dans la fonction calcul_surface_isovaleur(), après avoir construit le cube, on construit les 6 tétraèdres a partir des ensembles de sommets du cube : {0,2,3,7},{0,6,1,4},{5,6,1,4},{0,2,6,7},{0,4,6,7},{0,6,1,2}

Chaque sommet de chacun des 6 tétraèdres construits aura donc la valeur du sommet du cube correspondant.

Ce qui donne avec l'ensemble de sommets {0,2,3,7} :

```
tetra.p0 = p[0];
tetra.p1 = p[2];
tetra.p2 = p[3];
tetra.p3 = p[7];
tetra.v0 = val[0];
tetra.v1 = val[2];
tetra.v2 = val[3];
tetra.v3 = val[7];
traitement_tetraedre(tetra);
```

Ensuite, a partir de la marque de chaque sommet d'un tétraèdre ….

```
// Calcul de la marque du sommet P de valeur associee F
int marque(Point3D P, float F, float isoVal)
{
        if(F < isoVal)
                return 0;
        return 1;
}
```

… on calcule les surfaces isovaleur en prenant en compte les 16 configurations possibles et en ajoutant les triangles correspondant :

```
void traitement_tetraedre(Tetraedre T) {
Triangle3D t1;
Triangle3D t2;
int m0 = marque(T.p0, T.v0, isoVal);
int m1 = marque(T.p1, T.v1, isoVal);
int m2 = marque(T.p2, T.v2, isoVal);
int m3 = marque(T.p3, T.v3, isoVal);
```

```
if(m0 != m1 && m0 != m2 && m0 != m3)
{
        t1.p[0] = intersection_arete_hyperplan(isoVal, T.p0, T.p1, T.v0, T.v1);
        t1.p[1] = intersection_arete_hyperplan(isoVal, T.p0, T.p2, T.v0, T.v2);
        t1.p[2] = intersection_arete_hyperplan(isoVal, T.p0, T.p3, T.v0, T.v3);
        AJOUT_TRIANGLE(t1);
} // suivi de tous les autres tests pour les autres configurations possibles...
}
```

Nous permettant de trouver l'objet caché dans le bloc de résine : un homard !!