Voisinages.

Laurent Cancé Francis
07/02/2018

I. Présentation du problème

Dans la considération de maillages, il est besoin de trouver une méthode de détection d'arrêtes concomitantes et de sommets proches afin d'établir des topologies efficientes.

II. Le problème de recherches de sommets avoisinants

Il s'agit de considérer l'algorithme de base suivant renvoyant le sommet :

```
for (n=0;n<nVertices;n++)
{
	res=-1;
	m=0;
	while ((m<n)&&(res<0))
	{
		Vector v=Vertices[n].Position - Vertices[nn].Position;
		if (v.Norme()<SMALLF) res=m;
		m++;
	}

	if (res>=0)
	{
		// traitement
	}
}
```

La détection des sommets proches s'effectue par 3 soustractions, 3 multiplications, 2 additions et un calcul de racine carrée.

Par le précalcul suivant :

```
r=256.0f/sqrtf(MESH_RADIUS);
for (n=0;n<nVertices;n++)
{
	tag=((int)((256+Vertices[n].Position.x*r)))+
			512*(((int)((256+Vertices[n].Position.y*r)))+
			512*((int)((256+Vertices[n].Positon.z*r))));
}
```

il vient l'élimination triviale des sommets disjoints par le simple test de la valeur 'tag' pour chaque sommets ; quand le maillage contient un nombre important de sommets c'est très efficace.

III. Le problème des arrêtes jointes

La détection des triangles adjacents devient vite un calcul imposant dès lors que l'on souhaite sélectionner un groupe de triangles adjacents d'un maillage

Définissons le tableau de références basé sur une fenêtre de jonctions d'indices:

```
// initialisation du pointeur

int * references=new int[nVertices*4];

int ofs2=nVertices*2;


// initialisation des références

for (n=0;n<ofs2;n++)
{
        references[n]=nFaces;
        references[ofs2+n]=0;
}


// calcul des références d'indices d'arrêtes

for (n=0;n<nFaces;n++)
{
        Faces[n].i0=Faces[n].v0+Faces[n].v1;
        Faces[n].i1=Faces[n].v1+Faces[n].v2;
        Faces[n].i2=Faces[n].v2+Faces[n].v0;

        if (n<references[Faces[n].i0]) references[Faces[n].i0]=n;
        if (n<references[Faces[n].i1]) references[Faces[n].i1]=n;
        if (n<references[Faces[n].i2]) references[Faces[n].i2]=n;

        if (n>references[ofs2+Faces[n].i0]) references[ofs2+Faces[n].i0]=n;
        if (n>references[ofs2+Faces[n].i1]) references[ofs2+Faces[n].i1]=n;
        if (n>references[ofs2+Faces[n].i2]) references[ofs2+Faces[n].i2]=n;
}
```

(Avec Faces.v0 Faces.v1 Faces.v2 les indices des sommets d'un triangle du maillage)

La détection de l'arrête [Faces.v0,Faces.v1] s'effectue ainsi :

```
    // arrête 0-1
register int tag=Face.i0;

  int minf=references[tag];
  int maxf=references[nVertices*2+tag];

  for (n=minf;n<=maxf;n++)
  {
            if (tag==Faces[n].i0)
            {
                    if ((Faces[n].v0==Face->v1)&&(Faces[n].v1==Face->v0)) return n;
            }
            else if (tag==Faces[n].i1)
            {
                    if ((Faces[n].v1==Face->v1)&&(Faces[n].v2==Face->v0)) return n;
            }
            else if (tag==Faces[n].i2)
            {
                    if ((Faces[n].v2==Face->v1)&&(Faces[n].v0==Face->v0)) return n;
            }
  }
```

L'algorithme permet de réduire le nombre de test par une fenêtre applicative dynamique et permet même le pré-calcul de champs .01 .12 .21 de faces en utilisation par des algorithmes récursifs.

IV. Conclusion

Le calcul de topologies des maillages s'effectue difficilement sans l'adjonction de systèmes de tris grossiers, car la complexité des maillages est N x N.
La présentation du problème par le détail constitutif en N ou N x 2 permet un traitement beaucoup plus rapide.