RT/LSI-97.13-T

Disseny d'un octree extés

Eva Monclús
Isabel Navazo

Report LSI–97–13–T

BIBLIOTECA POLITÈCNICA SUPERIOR D'ENGINYERIA
Campus Nord

Disseny d'un octree extès

Eva Monclús Lahoya Isabel Navazo Álvaro *
Departament de Llenguatges i Sistemes Informàtics
Universitat Politècnica de Catalunya
Av. Diagonal 647, 8ª, E-08028 Barcelona

24th July 1997

Abstract

The Extended Octree [Nav86] is a generalization of classical Octrees, based on the inclusion of four new node types (Face, Edge, Vertex and QuasiVertex). These new terminal nodes contain a piece of the boundary of the object in order to representate the polyhedral object exactly.

This report consists of the dessign and implementation in C++ of a Extended Octree. Its dessign is defined using the generic octree implementated in [Sal97]. (This report is written in catalan).

*e-mail:{emonclus,isabel}@lsi.upc.es.This work has been suported by the CICYT, under the project TIC 95-630-C05

Índex

1 Introducció **1**

2 Classe octree extès **2**

2.1 Tipus de nodes: classe node_oct 2

2.1.1 Especificació de la classe node_oct 3

2.1.2 Classes derivades de node_oct 5

2.2 Classe eo 7

2.2.1 Especificació classe eo 8

2.2.2 Explicació detallada dels mètodes 9

3 Arbre de plans **11**

3.1 Tipus de plans: plans reals i discrets 12

3.1.1 Especificació classe pla real 12

3.1.2 Explicació mètodes de la classe pla real 12

3.1.3 Especificació classe pla discret 13

3.1.4 Explicació mètodes de la classe pla discret 13

3.2 Jerarquia de classes en l'arbre de plans 13

3.2.1 Especificació classe node_pla 14

3.2.2 Explicació mètodes de la classe node_pla 15

3.2.3 Especificació classe abstracta arbre_d 15

3.2.4 Explicació detallada dels mètodes de *arbre_d* 15

3.2.5 Especificació de la classe arbre plans reals: arbred_pr . . . 15

3.2.6 Especificació de la classe arbre plans discrets: arbred_pd . 16

3.2.7 Constructor de l'arbre de plans discrets 16

4 Conclusions **16**

Referències **17**

A Manual del programador **18**

A.1 Ubicació del paquet octree extès 18

A.1.1 Makefile de l'aplicació 19

A.1.2 Jerarquia de mòduls 20

A.2 Mòdul de construcció octree extès a partir del model brep 21

A.3 Mòdul d'operacions booleanes 22
A.4 Mòdul de construcció del model brep a partir de l'octree extès . . 23
A.5 Mòdul de compactació d'un octree extès 24
A.6 Mòdul de visualització d'un octree extès 25
A.7 Mòdul auxiliar de rutines geomètriques 26
A.8 Detalls per a la seva utilització 26

Índex de figures

1 Nodes terminals de l'octree extès. 1
2 Jerarquia de classes eo. 3
3 Jerarquia de classes pels diferents tipus de nodes. 3
4 Diagrama octree extès – arbre plans. 11
5 Esquema de l'estructura de l'arbre de plans. 12
6 Jerarquia classes en el disseny de l'arbre de plans. 14
7 Jerarquia de directoris del paquet octree extés. 18
8 Jerarquia mòduls de la classe octree extès. 21
9 Arbre de les principals crides en el pas de br a oct. extès. 22
10 Arbre de les principals crides en les operacions booleanes amb octrees. 23
11 Arbre de les principals crides en la conversió d'un oct. extès a br. 23
12 Arbre de les principals crides en la visualització d'un oct. extès. . 26

Índex de taules

1 Complementari d'un node extès. 4
2 Compactació entre nodes. 24

1 Introducció

Un dels esquemes de representació habitual d'objectes en l'ordinador, són els octrees [Sam89]. Donat un univers cúbic que conté al sòlid, un octree representa una partició d'aquest espai en cubs (octants) de mides diferents que contenen una part concreta del sòlid. La partició realitzada del cub inicial es codifica amb un arbre octal. Els nodes interiors d'aquest arbre representen octants que s'han continuat subdividint i se'ls etiqueta com de tipus *gris*. Els nodes terminals de l'arbre són els que emmagatzemen la informació del sòlid.

Aquest document es centra en l'octree extès [Nav86], model utilitzat per la descripció de la frontera d'un sòlid polièdric.

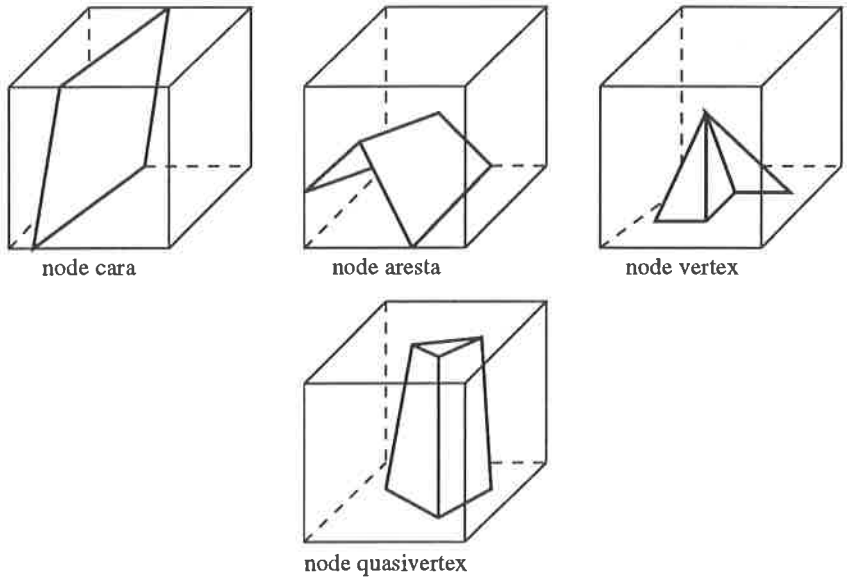

Figura 1: Nodes terminals de l'octree extès.

L'octree extès presenta els següents tipus de nodes terminals (nodes fulla de l'arbre octal) (veure figura 1):

Blanc: node completament exterior al sòlid.

Negre: node completament interior al sòlid.

Cara: node que només conté part d'una cara del sòlid. Es guarda el pla suport de la cara que atravessa el node.

Aresta: node que només conté part d'una aresta del sòlid. Per l'aresta es guarden els dos plans que defineixen l'aresta i la seva configuració [1].

Vèrtex: node que només conté un vèrtex de la frontera del sòlid, i cap més part de la frontera d'aquest. En aquest cas, es guarden els plans que conflueixen en el vèrtex i la configuració de cada aresta.

[1] Com configuració d'aresta entenem la convexitat o concavitat que formen els dos plans.

Quasivèrtex: node que conté un subconjunt de les cares que conflueixen en un vèrtex no interior al node. En aquest cas, es guarden els plans de les cares i la configuració de les arestes que formen el vèrtex.

Gris mínim: node gris de grau de subdivisió de l'arbre màxima. No es guarda informació sobre la geometria en el seu interior.

Aquest document vol presentar el disseny i implementació , en llenguatge *C++*, d'un octree extès. Això comporta a més del disseny de la pròpia classe *octree extès*, el disseny de l'emmagatzemament de les equacions dels plans presents en la frontera del sòlid. A efectes d'optimitzar memòria i per robustesa del model, es mantindrà una única còpia dels plans, els quals, com veurem en la secció 3, es representaran amb una estructura arborescent.

El document s'estructura de la següent manera: En la primera secció es descriu amb detall la jerarquia de classes definides en el disseny de l'octree extés. A continuació, s'explica la estructura de dades emprada pels plans d'un octree i les diferents possibilitats del tipus de plans emprats. Finalment, es dóna un petit manual del progamador, pels usuaris del paquet "octree extès".

Amb aquest nou disseny dels octrees extesos, s'ha actualitzat l'aplicació de demostració dels octrees extesos. Per coneixer el seu nou funcionament, consultar el manual d'usuari [MN97].

2 Classe octree extès

El disseny de l'octree extès es realitza a partir del disseny de l'octree genèric donat en [Sal97]. Concretament s'implementarà un octree extès mitjançant un arbre amb punters *(octreep)*.

L'octree genèric amb punters ofereix l'esquelet de qualsevol octree implementat com un arbre amb punters. En el seu disseny, es recomana haver llegit el report [Sal97], s'ha deixat oberta la informació que conté cada node de l'arbre. Segons la definició de [Sal97] un octree genèric amb punters és:

template <class T> class octreep: public octreebasic<T> { ... }

En la figura 2 es presenta un diagrama de la jerarquia de classes presents en el disseny d'un octree extès. La classe *node_oct* representa la informació que ha de contindre cada node, és a dir, defineix la *T* del template *octreep*. La classe *eo* representa l'octree extès, la qual, com es pot veure en la figura 2, es deriva publicament del template *octreep*.

2.1 Tipus de nodes: classe node_oct

Ja que existeixen diferents tipus de nodes (gris, blanc, negre, etc.), hem establert una classe abstracta anomenada *node_oct*, classe que ens homogenitza els diferents tipus d'informació emmagatzemats en els nodes. A més altres tipus d'octrees (Face octree, ...) també presenten alguns dels nodes descrits a continuació; és per això, que ens ha semblat adient utilitzar aquesta jerarquia de

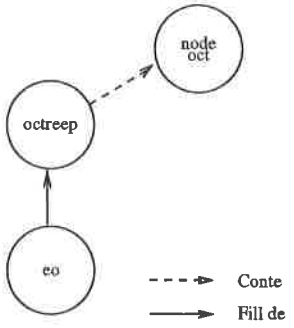

Figura 2: Jerarquia de classes eo.

classes per començar a tindre un paquet de nodes d'octrees reutilitzables en altres models.

En la figura 3, es mostra la jerarquia de classes establertes en el disseny dels diferents nodes d'un octree extès. Com es pot observar, de la classe *node_oct* es deriven les classes que implementen la informació que contenen els diferents tipus de nodes d'un octree extès.

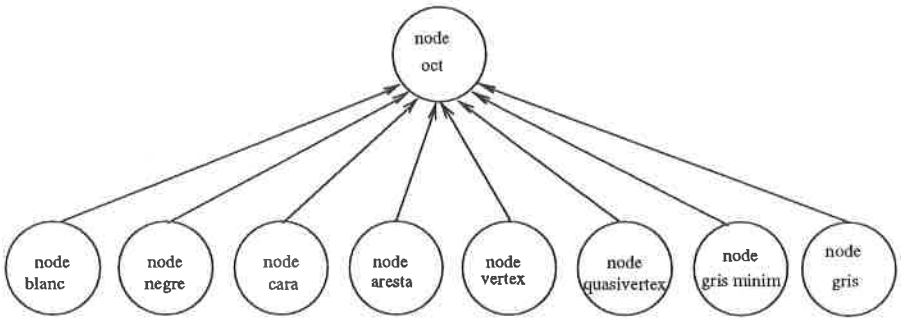

Figura 3: Jerarquia de classes pels diferents tipus de nodes.

L'objectiu d'aquesta classe ha estat fer transparent al programador, els diferents tipus de nodes existents en un octree extès; aconseguint així més claretat i transparència en la implementació de moltes operacions no depenents dels tipus de nodes i en el tractament de la informació comuna a cada node.

2.1.1 Especificació de la classe node_oct

```
class node_oct {
  public:
    virtual t_node_oct node_tipus () const = 0;
    virtual void informacio (t_node_oct&, int&, id_pla*, config*) const = 0;
    virtual node_oct* complementari () const = 0;
    virtual node_oct* copia () const = 0;
```

};

Seguidament, explicarem els mètodes oferts per aquesta classe. Tots els mètodes són virtuals purs, és a dir, estan definits a nivell de classe derivada de *node_oct*.

- virtual t_node_oct **tipus** () const = 0;

  Mètode de consulta per a obtenir el tipus del node.

- virtual void **informacio** (t_node_oct *&tipus*, int *&nplans*,
  id_pla *plans[]*, config *configs[]*)
  const = 0;

  Mètode de consulta per a obtenir la informació d'un node sense necessitat de coneixer el seu tipus. Depenent del tipus de node, existiran paràmetres que no tindran cap valor. En general, d'un node ens interessa coneixer:

  - el seu tipus.
  - nombre de plans.
  - identificadors dels plans, presents en la geometria del sòlid, que conté el node.
  - configuració de les arestes formades pels plans que conté el node.

- virtual node_oct* **copia** () const = 0;

  Mètode per a obtenir un nou node amb el mateix contingut.

- virtual node_oct* **complementari** () const = 0;

  Mètode de consulta per a obtenir el node complementari d'un node.

  A la taula següent s'explica el que s'enten per node complementari de cada tipus de node:

Tipus node	Complementari	
blanc	negre	
negre	blanc	
gris_minim	×	
gris	×	
cara	cara	cara oposada, és a dir, equació pla amb l'orientació complementaria
aresta	aresta	plans i configuració oposats
vèrtex	vèrtex	plans oposats i configuracions oposades amb sentit de recorregut invers
quasivèrtex	quasivèrtex	

Taula 1: Complementari d'un node extès.

A continuació, presentarem les classes que defineixen els diferents tipus de nodes d'un octree extès.

Tots els tipus de nodes en els quals es guarda informació sobre els plans ofereixen dos constructors diferents depenent de com reben la informació referent a un pla:

- donant els coeficients a, b, c, d de l'equació implícita $ax + by + cz + d = 0$
- donant l'identificador del pla en l'arbre de plans.

A més, els nodes que contenen informació específica ofereixen un mètode de consulta particular de cada tipus.

També, els nodes que emmagatzemen informació geomètrica (aresta, cara, vèrtex i quasivèrtex) ofereixen un mètode que construeix els polígons que limiten els troços de cares del sòlid interiors al node.

2.1.2 Classes derivades de node_oct

```
class node_gris: public node_oct {
  public:
    node_gris ();
    t_node_oct tipus () const;
    void informacio (t_node_oct&, int&, id_pla*, config*) const;
    node_gris* complementari () const;
    node_gris* copia () const;
    ~node_gris ();
  };

class node_negre: public node_oct {
  public:
    node_negre ();
    t_node_oct tipus () const;
    void informacio (t_node_oct&, int&, id_pla*, config*) const;
    node_oct* complementari () const;
    void reconstruir (punt_3d*, int*, int&) const;
    node_negre* copia () const;
    ~node_negre ();
  };

class node_blanc: public node_oct {
  public:
    node_blanc ();
    t_node_oct tipus () const;
    void informacio (t_node_oct&, int&, id_pla*, config*) const;
```

```
    node_oct* complementari () const;
    node_blanc* copia () const;
    ~node_blanc ();
  };

class node_cara: public node_oct {
    id_pla pla;
  public:
    node_cara (id_pla);
    node_cara (float*);
    t_node_oct tipus () const;
    void informacio (id_pla&) const;
    void informacio (t_node_oct&, int&, id_pla*, config*) const;
    void reconstruir (punt_3d*, int&) const;
    node_cara* complementari () const;
    node_cara* copia () const;
    ~node_cara ();
  };

class node_aresta: public node_oct {
    id_pla llista_plans[2];
    config config_aresta;
  public:
    node_aresta (id_pla*, config);
    node_aresta (float abcd[2][4], config);
    t_node_oct tipus () const;
    void informacio (id_pla*, config&) const;
    void informacio (t_node_oct&, int&, id_pla*, config*) const;
    void reconstruir (punt_3d*, punt_3d*, int&, int&) const;
    node_aresta* complementari () const;
    node_aresta* copia () const;
    ~node_aresta ();
  };

class node_vertex: public node_oct {
    int num_plans;
    id_pla* llista_plans;
    config* config_arestes;
  public:
    node_vertex (int, id_pla*, config*);
    node_vertex (int, float abcd[][4], config*);
    t_node_oct tipus () const;
    void informacio (int&, id_pla*, config*) const;
    void informacio (t_node_oct&, int&, id_pla*, config*) const;
```

```
    void reconstruir (punt_3d*, int*, int&) const;
    node_vertex* complementari () const;
    node_vertex* copia () const;
    ~node_vertex ();
  };

class node_quasivertex: public node_oct {
    int num_plans;
    id_pla* llista_plans;
    config* config_arestes;
  public:
    node_quasivertex (int, id_pla*, config*);
    node_quasivertex (int, float abcd[][4], config*);
    t_node_oct tipus () const;
    void informacio (int&, id_pla*, config*) const;
    void informacio (t_node_oct&, int&, id_pla*, config*) const;
    void reconstruir (punt_3d*, int*, int&) const;
    node_quasivertex* complementari () const;
    node_quasivertex* copia () const;
    ~node_quasivertex ();
  };

class node_gris_minim: public node_oct {
  public:
    node_gris_minim ();
    t_node_oct tipus () const;
    void informacio (t_node_oct&, int&, id_pla*, config*) const;
    node_gris_minim* complementari () const;
    node_gris_minim* copia () const;
    ~node_gris_minim ();
  };
```

2.2 Classe eo

Un octree extès es defineix com un octree genèric amb punters (octreep) instanciant el template (T) amb la classe *node_oct* (veure figura 2).

La classe *eo* es deriva publicament d'*octreep*, per tant a més dels mètodes específics de la classe eo, s'oferiran també els mètodes públics d'octreep [Sal97]. Els mètodes heretats de la classe *octreep* són:

- set{nom, origen, longitudaresta}: mètodes d'actualització del nom i l'univers de l'octree.

- get{nom, origen, longitudaresta, profmaximapos, profmaximareal, resolucio, esbuit, esvalid}: mètodes de consulta d'atributs generals i de l'estat de l'octree.
- crearnode: mètode de creació de nodes.
- subdividirnode, arrelarnodes, setarrel, esborrararrel, esborrarfills: mètodes per la creació i modificació estructura de l'arbre.
- getnode{arrel, fill, pare, punt, vei}: mètodes de selecció de nodes.
- setnode: mètode d'establiment i modificació contingut dels nodes.
- esnodeterminal, existeixnode, get{longitud, prof, origen, extrem }node, getnode: mètodes de consulta d'atributs dels nodes
- recorregut {Preordre, Postordre, Paral·lela (FTB, BTF) i Perspectiva (FTB, BTF)}: mètodes pel recorregut de l'arbre octree.

2.2.1 Especificació classe eo

```
class eo: public octreep<node_oct* > {
    int status;
    public:
    eo ();
    eo (String, float, float, float, punt_3d, unsigned char);
    eo (String);
    eo (String, String);
    void informacio (inf_eo&) const;
    bool escriure (String) const;
    bool pas_eo_br (String) const;
    eo* compactar () const;
    eo* operator * (const eo&) const;
    eo* operator + (const eo&) const;
    eo* operator − (const eo&) const;
    eo* complementari () const;
    bool visualitzar (const Camera&, const Finestra&, bool) const;
    paralelepiped capsa_contenidora (int) const;
    void copiar_subarbre (eo*, idnode, idnode);
    void copiar_subarbre_complementat (eo*, idnode, idnode);
    int error () const;
    void set_error (int);
    ~eo();
    };
```

2.2.2 Explicació detallada dels mètodes

- **eo** ();

  Constructor per defecte de *eo*.

- **eo** (String *nom*, float *lx*, float *ly*, float *lz*, punt_3d *origen*, unsigned char *prof*);

  Construeix un *eo* amb els atributs globals que s'indiquen i amb el node arrel de tipus *gris*. A més, reserva espai pels vuit fills de l'arrel.

  Els atributs globals als que ens referim són:

  - nom
  - univers paral·lelepipèdic: origen (punt 3d), dim_x, dim_y i dim_z.
  - profunditat màxima

- **eo** (String *path*);

  Constructor d'un octree extès a partir de la lectura del fitxer <path>. Si hi ha hagut algun tipus d'error en la lectura: format incorrecte, manca d'espai, etc., l'atribut *status* contindrà el tipus d'error.

  El fitxer llegit està escrit en el format textual Mdtl [VRB97].

- **eo** (String *path_br*, String *path_eo*);

  Constructor d'un octree extès a partir del model de fronteres d'un sòlid. Primer de tot es realitza la lectura de l'objecte amb el conseqüent omplert de l'estructura de dades *br*. Si hi ha hagut algun error, l'atribut status informa del tipus d'error.

  El fitxer que conté el sòlid br es suposa que està escrit en el format textual Mdtl.

  Les precondicions d'aquest constructor són:

  - La capsa contenidora de l'objecte s'ha de trobar en l'octant positiu.
  - La longitud de les arestes ha d'esser menor o igual que la màxima longitud permesa per la classe.

- void **informacio** (inf_eo&) const;

  Mètode de consulta del nombre de nodes existents de cada tipus.

- bool **escriure** (String *path*) const;

  Escriu l'octree en el fitxer <path> seguint el format textual Mdtl [VRB97]. Retorna cert si s'ha generat algun error: falta d'espai en disc, etc. L'atribut *status* conté el tipus d'error.

- bool **pas_eo_br** (String *path*) const;

  Mètode de pas d'un octree extès a un sòlid br, generant el fitxer <path> que conté el sòlid br obtingut en el format Mdtl. L'acció retorna cert si s'ha produit algun error: falta d'espai en disc, etc. L'atribut *status* conté el tipus d'error.

- eo* **operator** * (const eo *&eo1*) const;

  Realitza la intersecció entre els dos octrees extesos.

- eo* **operator** + (const eo sl &eo1) const;

  Realitza la unió entre els dos octrees extesos. La unió es realitza seguint la fórmula:

$$A \cup B = \neg(\neg A \cap \neg B)$$

- eo* **operator** − (const eo *&eo1*) const;

  Realitza la diferència entre els dos octrees seguint la fórmula:

$$A - B = A \cap \neg B$$

- eo* **complementari** () const;

  Creació d'un nou octree extès a partir de la seva complementació. Això consisteix en realitzar un recorregut de l'arbre complementant cada node de l'octree seguint la taula 1.

- bool **visualitzar** (const Camera *&camera*, const Finestra *&finestra*, bool clipping) const;

  Visualització d'un octree extès. Les classes Camera i Finestra es defineixen en [Gar97]. En aquesta funció es presupossa que la camera està totalment definida i que la finestra es troba ja mapejada. El paràmetre *clipping* ens indica si cal realitzar el retallat dels nodes que no són projectats en el window (p.ex: cas de realitzar zoom).

- paralelepiped **capsa_contenidora** (int *nivell*) const;

  Càlcul capsa contenidora aproximada de l'octree a partir del seu recorregut fins a la profunditat especificada pel paràmetre *nivell*.

- void **copiar_subarbre** (eo* *origen*, idnode *subarbre*, idnode *desti*);

  Realitza la còpia del subarbre (*subarbre*) de l'octree extès *origen* al node *desti* de l'octree extès que crida al mètode.

- void **copiar_subarbre_complementat** (eo* *origen*, idnode *subarbre*, idnode *desti*);

  Igual que l'acció anterior però complementant la informació dels nodes del subarbre *origen*.

- ~eo ()

  Destructor d'un octree extès. Realitza un recorregut per l'arbre alliberant la memòria de cada node.

En l'apèndix A s'aprofunditzarà més sobre el disseny de les comandes més importants: operacions booleanes, visualització, compactació i conversió entre els models brep i octree extès.

3 Arbre de plans

Com s'ha explicat en la definició dels tipus de nodes d'un octree extès (veure secció 2.1.1) es requereix informació dels plans que defineixen la frontera del sòlid.

Primer de tot enunciarem les restriccions de disseny que hem adoptat referent als plans d'un octree extès:

- els plans es consideren orientats i normalitzats.
- per garantir la robustesa del model, l'equació d'un pla només es guarda una vegada.
- dos plans es consideren el mateix si es troben a la mateixa distància de l'origen de coordenades i l'angle entre ells és menor que un cert valor *anglemin* [2].

En el diagrama 4 s'especifica la relació entre l'octree extès i els plans (emmagatzemats emprant una estructura arborescent). Un octree extès (eo) és un arbre octal on cada node de l'arbre guarda informació de la frontera del sòlid (node_oct). Ja que la informació és geomètrica, és necessari guardar les equacions dels plans, els quals, com hem anat dient al llarg del document es guarden una única vegada emprant una estructura arborescent (arbre_d). Per tant dins dels nodes d'un octree extès es guardarà un pla com una referència a un node (node_pla) d'una estructura arborescent que el conté.

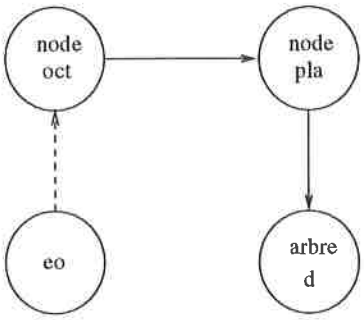

Figura 4: Diagrama octree extès – arbre plans.

Per optimitzar la cerca d'un pla, aquests s'estructuren en forma d'arbre binari ordenat segons la distància del pla a l'origen, és a dir, el valor absolut de la component d del pla. L'estructura de l'arbre de plans es presenta en la figura 5.

Cada node de l'arbre representa a un conjunt de plans amb la mateixa d. Com d pot ser positiva o negativa, cada node té associat dues llistes de plans (plans amb d positiva i plans amb d negativa). A cada registre de les llistes de plans es guarda l'equació del pla, un atribut amb informació auxiliar al pla, un contador amb el nombre de nodes que referencien aquest pla i l'identificador al seu pla oposat si existeix.

[2] *anglemin* és una constant de l'aplicació amb al valor 3.0E-04

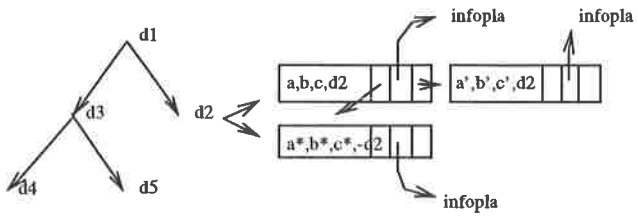

Figura 5: Esquema de l'estructura de l'arbre de plans.

3.1 Tipus de plans: plans reals i discrets

La informació d'un pla consisteix en la seva equació (a, b, c, d nombres reals). Però també podem considerar els plans discretitzats segons una funció predeterminada. Això comporta poder escollir entre dos tipus de plans diferents: pla real o pla discret.

Per tal de fer transparent aquest fet a la resta de classes i al codi que utilitza plans, s'ha dissenyat una classe abstracta anomenada *pla* que servirà d'interfície per les classes o funcions que treballin amb plans.

```
class pla {
  public:
    virtual void equacio (float abcd[4]) const =0;
    virtual bool igual_amb (pla*) const = 0;
    virtual bool oposat_amb (pla*) const = 0;
  };
```

3.1.1 Especificació classe pla real

```
class pla_real: public pla {
    float abcd[4];
    friend bool igual_sentit_normals (const pla_real*, const pla_real*);
    friend bool oposat_sentit_normals (const pla_real*, const pla_real*);
  public:
    pla_real (float abcd[4]);
    void equacio (float abcd[4]) const;
    bool igual_amb (pla*) const;
    bool oposat_amb (pla*) const;
  };
```

3.1.2 Explicació mètodes de la classe pla real

- **pla_real** (float *abcd[4]*);

  Constructor de la classe a partir dels coeficients de l'equació implícita $ax + by + cz = d = 0$.

- void **equacio** (float *abcd[4]*) const;

  Mètode de consulta per a obtenir els coeficients de l'equació del pla.

- bool **igual_amb** (pla* *pl*) const;

  Dos plans es consideren iguals si es troben a la mateixa distància de l'origen de coordenades i l'angle que formen es menor que *anglemin*.

- bool **oposat_amb** (pla* *pl*) const;

  Dos plans es consideren oposats si son coincidents però amb orientació oposada.

3.1.3 Especificació classe pla discret

```
class pla_discret: public pla {
    long abcd[4];
    double s;
    static f_discretitz *f_discr;
    static f_inv_discretitz *f_inv_discr;
  public:
    pla_discret (float abcd[4]);
    void equacio (float abcd[4]) const;
    bool igual_amb (pla*) const;
    bool oposat_amb (pla*) const;
  };
```

3.1.4 Explicació mètodes de la classe pla discret

- **pla_discret** (long *abcd[4]*);

  Constructor de la classe a partir de la discretització dels coeficients de l'equació implícita mitjançant la funció *f_discr*.

- void **equacio** (float *abcd[4]*) const;

  Mètode de consulta per a obtenir l'equació real del pla aplicant la funció inversa de discretització *f_inv_discr*.

- bool **igual_amb** (pla* *pl*) const;

  Dos plans es consideren iguals si tenen els mateixos coeficients.

- bool **oposat_amb** (pla* *pl*) const;

3.2 Jerarquia de classes en l'arbre de plans

Ja que depenent del tipus de pla que s'utilitzi les operacions i atributs de l'arbre que els emmagatzema són diferents, s'ha optat per seguir la mateixa filosofia que pel cas dels plans. Així, existeixen dos tipus d'arbres: *arbred_pr* (arbre amb plans reals) i *arbred_pd* (arbre amb plans discrets). També, s'ha dissenyat la

classe abstracta *arbred* per tal de poder independitzar les aplicacions del tipus d'arbre que utilitzin. La classe *node_pla* representa la informació d'un pla dins de l'arbre (veure figura 5).

En la figura 6 podem observar com queda la jerarquia de classes que ofereix aquests conceptes.

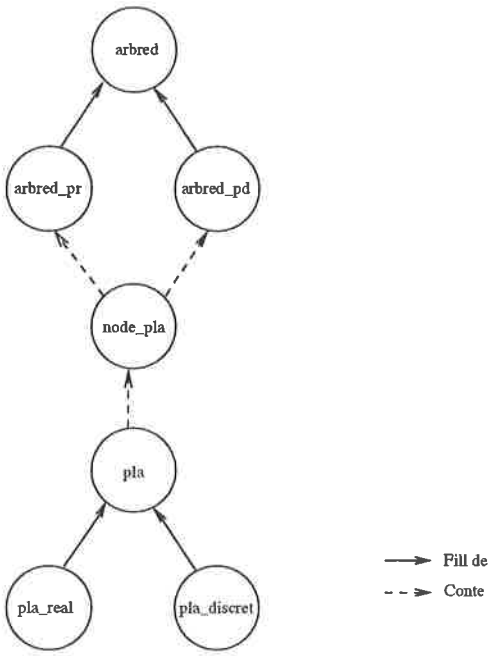

Figura 6: Jerarquia classes en el disseny de l'arbre de plans.

Com s'ha explicat en aquesta secció, hem dissenyat dos classes abstractes (arbre_d i pla) per poder-se öblidar"del tipus de plans amb els quals treballa la nostra aplicació.

3.2.1 Especificació classe node_pla

```
class node_pla {
    pla* pla;
    long cont;
    inf_pla infor;
    node_pla* oposat;
  public:
    node_pla (const pla*);
    node_pla* get_oposat () const;
    inf_pla informacio () const;
    void equacio (float abcd[4]) const;
  };
```

3.2.2 Explicació mètodes de la classe node_pla

- **node_pla** (const pla*);

  Constructor de la classe a partir d'un apuntador a un pla (pla real o discret).

- node_pla* get_oposat () const;

  Mètode de consulta del pla oposat. Si no existeix retorna NULL.

- inf_pla informacio () const;

  Mètode de consulta de la informació que conté un pla.

- void equacio (float abcd[4]) const;

  Mètode de consulta de l'equació del pla que conté el node.

3.2.3 Especificació classe abstracta arbre_d

```
typedef node_pla* id_pla;
class arbred {
  public:
    virtual id_pla afegir_pla (float abcd[4]) = 0;
    virtual void afegir_pla_oposat (const id_pla id&) = 0;
    virtual void esborrar_pla (id_pla id) = 0;
  };
```

3.2.4 Explicació detallada dels mètodes de *arbre_d*

- virtual id_pla afegir_pla (float abcd[4]) = 0;

  Afegeix el pla amb equació $ax + by + cz + d = 0$ en l'arbre de plans. Si el pla ja existia retorna el seu identificador sino l'afegeix.

- virtual void afegir_pla_oposat (const id_pla id&) = 0;

  Afegir el pla oposat del pla representat per *id*.

- virtual void esborrar_pla (id_pla id) = 0;

  Mètode per esborrar un pla de l'arbre de plans. Primer es decrementar el contador del node, si el contador es zero, esborra el pla del node corresponent de l'arbre.

3.2.5 Especificació de la classe arbre plans reals: arbred_pr

```
class arbred_pr: public arbred {
    float d;
    arbred_pr* filld,* fillesq;
    list<id_pla> apmes, apmenys;
  public:
```

```
    arbred_pr();
    id_pla afegir_pla (float abcd[4]);
    void afegir_pla_oposat (const id_pla id&);
    void esborrar_pla (id_pla id);
    ãrbre_pr ();
  };
```

3.2.6 Especificació de la classe arbre plans discrets: arbred_pd

```
class arbred_pd: public arbred {
    long d;
    list<id_pla> apmes, apmenys;
    arbred_pd* fillesq,* filld;
  public:
    arbred_pd (f_discretitz *f_discr,f_inv_discretitz *f_inv_discr);
    id_pla afegir_pla (float abcd[4]);
    void afegir_pla_oposat (const id_pla id&);
    void esborrar_pla (id_pla id);
    ãrbre_pd ();
  };
```

3.2.7 Constructor de l'arbre de plans discrets

La classe *pla_discret* conté les funcions de discretització, com s'ha vist en 3.1.3. Aquests dos atributs són *statics*, això vol dir, que tots els plans discrets creats tindran les mateixes funcions.

Ens ha semblat lògic, que aquests atributs siguin inicialitzats quan es crei l'arbre de plans discrets.

4 Conclusions

S'ha dissenyat l'octree extés com un arbre octree gèneric amb punters mitjançant la instanciació del template amb la informació present en els nodes terminals d'un octree extès. Amb aquest nou disseny, s'ha actualitzat l'aplicació d'utilitats de l'octree extès. Per coneixer els detalls de l'actualització es pot consultar el manual [MN97].

Referències

[Gar97] Joan Jordi Garcia. Pfc: Eines de desenvolupament de software sobre la plataforma de modelatge i visualització de volum: Voldmi. 1997.

[MN97] Monclús, Eva and Navazo, Isabel. Manual de l'aplicació d'utilitats de l'octree extès. Technical report, Dept. of LSI, Universitat Politècnica de Catalunya, 1997.

[Nav86] Isabel Navazo. *Contribució a les tècniques de modelat geomètric d'objectes polièdrics usant la codificació amb arbres octals.* PhD thesis, Universitat Politècnica de Catalunya, 1986.

[Sal97] Francesc Sala. Disseny d'un arbre octree genèric. Technical Report LSI-97-2-T, Dept. of LSI, Universitat Politècnica de Catalunya, 1997.

[Sam89] H. Samet. *Applications of Spatial Data Structures.* Adison Wesley, 1989.

[VRB97] Vazquez, P., Rios, A., and Brunet,P. La representació textual mdtl: Gramàtica i apis. Technical report, Dept. of LSI, Universitat Politècnica de Catalunya, 1997.

A Manual del programador

A.1 Ubicació del paquet octree extès

El paquet *octree extès* es troba en el directori `octrees/extended_oct`.

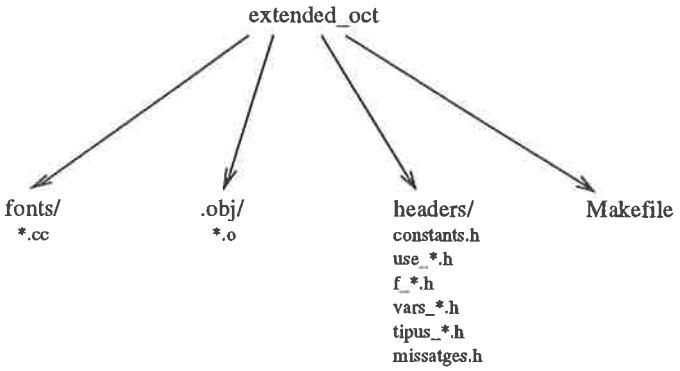

Figura 7: Jerarquia de directoris del paquet octree extés.

- En el directori **fonts** es troben els fitxers fonts de del paquet.
- En el directori **headers** es troben els headers necessaris per compilar el paquet. La nomenclatura és la següent:
  - els fitxers que comencen amb **tipus_** defineixen els objectes visibles.
  - els fitxers que comencen amb **use_** contenen tots els includes que necessita un mòdul.
  - els fitxers que comencen amb **f_** defineixen totes les funcions exportades pel mòdul.
  - el fitxer **constants.h** conté totes les constants globals de l'aplicació.
  - el fitxer **error_eo.h** conté tots els tipus d'errors que tracta el paquet, els altres es tractaran com a error intern.
- En el directori **.obj** es guarden els objectes dels fitxers fonts.

El directori `octrees/document/` conté tots els manuals relacionats amb els octrees.

- Manual de l'aplicació d'utilitats de l'octree extès.
- Disseny de l'octree extès (aquest mateix)
- en el subdirectori `oct_basic` es troba el manual [Sal97] on s'explica el disseny de l'octree genèric.

A.1.1 Makefile de l'aplicació

```
ARQDEP      := Depend.$(HOSTTYPE)

HEADERS := headers/
FONTS   := fonts/
OBJ     := .$(HOSTTYPE)_obj/
CC      := g++
C       := gcc

CCFLAGS = -g -I$(HEADERS) -fexternal-templates
CFLAGS  = -g -I$(HEADERS)


#
# Paths de cerca
#

vpath %.o $(OBJ)
vpath %.h $(HEADERS)
vpath %.c $(FONTS)
vpath %.cc $(FONTS)

include $(ARQDEP)

#-------------------------------
#   Regles Generiques
#-------------------------------


$(OBJ)%.o : $(FONTS)%.cc
$(CC) $(CCFLAGS) -c $< -o $@

$(OBJ)%.o : $(FONTS)%.c
$(C) $(CFLAGS) -c $< -o $@



OBJECTES = extended_octree.o node_oct.o punt_3d.o vector_3d.o \
        fitxers.o Visual.o br.o Matriu.o paralelepiped.o opb.o \
        pla.o reconstruir.o geom.o br.o basiques.o Punts.o \
        airepla2.o compactar.o br_eo.o eo_br.o  opb_aux.o \
        visualitzacio.o Camera.o Viewport.o octreebasic.o



.PHONY: all


all: $(addprefix $(OBJ), $(OBJECTES))
```

```
.PHONY: clean


#----------------------
#        CLEAN
#----------------------

clean:
        rm $(OBJ)*.o
        rm $(FONTS)*~
        rm $(HEADERS)*~



.PHONY: depend

#------------------
# dependencies
#------------------

F_FONTS = geom.cc airepla2.c fitxers.cc node_oct.cc \
        octreebasic.cc pla.cc basiques.cc extended_octree.cc \
        recorreguts.cc opb_aux.cc eo_br.cc visualitzacio.cc \
        punt_3d.cc paralelepiped.cc Viewport.cc Camera.cc \
        Matriu.cc Punts.cc Visual.cc vector_3d.cc br.cc geom.cc


depend: $(addprefix $(FONTS), $(F_FONTS))
$(CC) $(CCFLAGS) -MM $^ > $(ARQDEP)
```

A.1.2 Jerarquia de mòduls

En la figura 8 es presenta el diagrama de la jerarquia de mòduls del paquet octree extès.

A continuació, explicarem l'objectiu de cadascun dels mòduls funcionals.

- visualitzacio: realitza la visualització dels octrees.
- eo_br: realitza la conversió d'octree extès a br.
- br_eo: realitza la conversió de br a octree extès.
- opb: realitza les operacions booleanes entre octrees.
- opb_aux: funcions auxiliars de les operacions booleanes.
- compactar: realitza la compactació d'un octree extès.

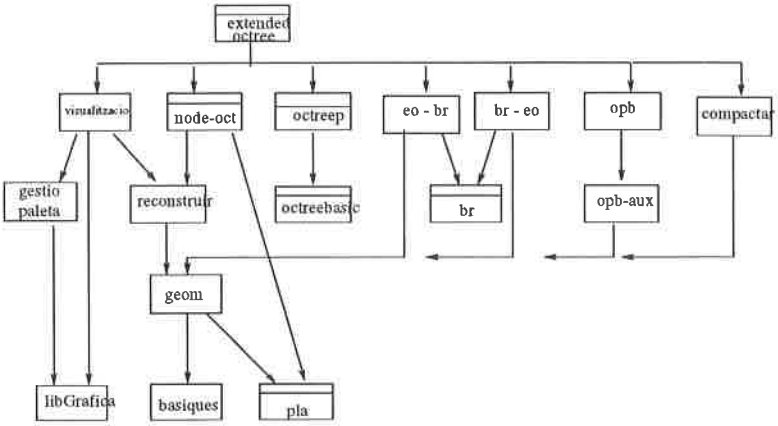

Figura 8: Jerarquia mòduls de la classe octree extès.

- gestio paleta: gestiona la paleta emprada en la visualització d'un octree.
- reconstruir: implementa les accions de reconstrucció de la geometria dels nodes extesos.
- geom: nucli geomètric de l'aplicació.
- basiques: funcions geomètriques bàsiques.
- libGrafica: llibreria que ofereix totes les funcionalitats relacionades amb les finestres gràfiques encapsulant l'entorn gràfic emprat (e.x: XWindow) [Gar97].

A.2 Mòdul de construcció octree extès a partir del model brep

En la figura 9 es mostra l'organigrama de les principals crides en la conversió.

En primer lloc, es llegeix el fitxer que conté el model de fronteres d'un objecte omplint l'estructura de dades *br*. A continuació s'accedeix al model de fronteres per inicialitzar estructures de dades que es requeriran en processos posteriors (*ini_dmi* i *ijk*). Basicament es tracta de crear noves relacions topològiques sobre el model de fronteres i de inicialitzar les llistes d'informació dels elements geomètrics que pertanyen a l'objecte i són interiors al cub inicial.

L'acció *octrees* utilitzant la rutina *clipping* distribueix la informació geomètrica interior a un node *gris* entre els seus vuit descendents; generant noves llistes d'informació per cadascun dels fills.

A partir de les llistes generades, l'acció *clasnode* identifica el tipus de cada node d'acord amb el nombre i classe dels elements geomètrics inclosos en elles. Si del anàlisi del tipus del node es dedueix que és *gris* i no s'ha arribat al tamany mínim del cub, es continua el procés de subdivisió recursiva, respecte a la superfície d'aquest node, cridant a l'acció *octtree* amb la informació d'aquest node.

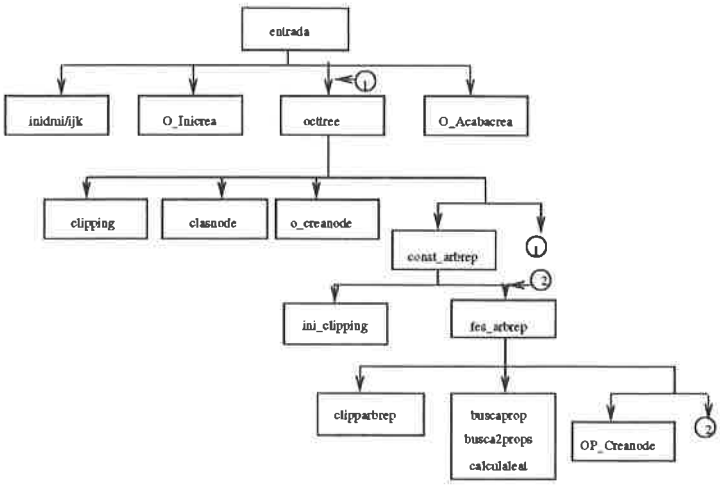

Figura 9: Arbre de les principals crides en el pas de br a oct. extès.

A.3 Mòdul d'operacions booleanes

Les operacions booleanes sobre octrees extesos, com van dir en la seva definició es realitzen totes a partir de la intersecció i complementació. Per tant, excloent possibles complementacions dels arbres d'entrada les operacions booleanes queden transformades en el càlcul de la intersecció dels dos octrees.

En la figura 10 es mostra l'organigrama de les principals crides a les funcions desenvolupades en les operacions booleanes. S'ha agrupat totes les crides inicials a possibles operacions booleanes en una única per claretat del esquema. A més es supossa que és en aquesta rutina *opbool* on s'inicialitzen les complementacions necessàries.

La intersecció entre dos octrees extesos es realitza fent un recorregut simultani dels dos arbres d'entrada. En aquest recorregut, es van intersectan nodes homòlegs (mateixa mida i ubicació en l'espai), i segons el seu tipus es generen un o més nodes del nou arbre.

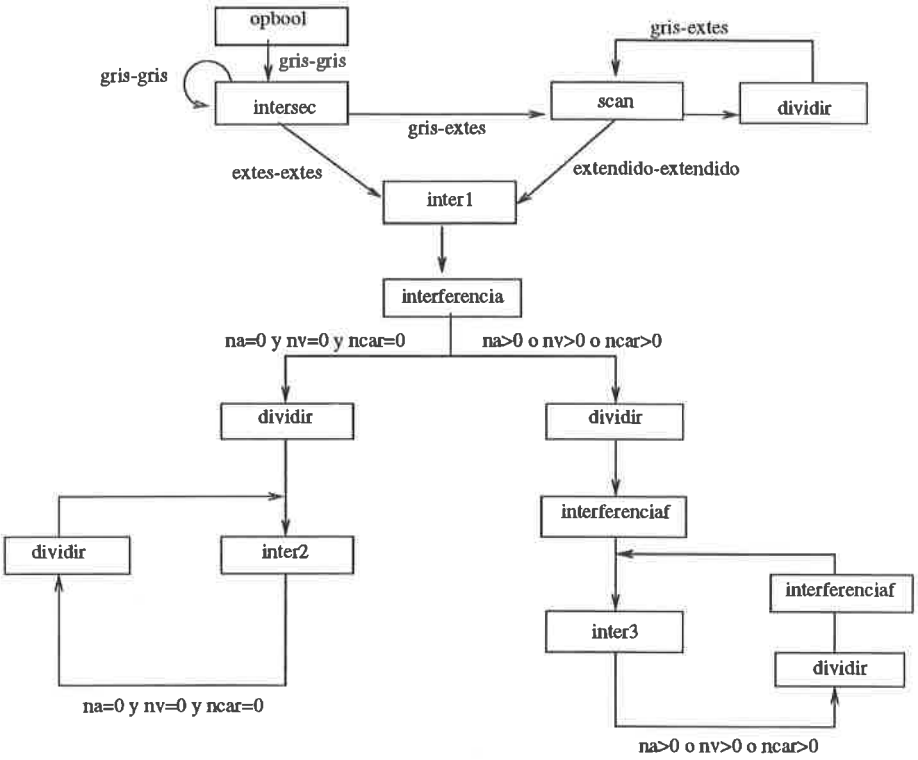

Figura 10: Arbre de les principals crides en les operacions booleanes amb octrees.

A.4 Mòdul de construcció del model brep a partir de l'octree extès

Aquest mòdul rep com a dada el model d'un octree extés i genera un model de fronteres de la seva superfície. En la figura 11 es mostra l'organigrama general que especifica la filosofia del mòdul.

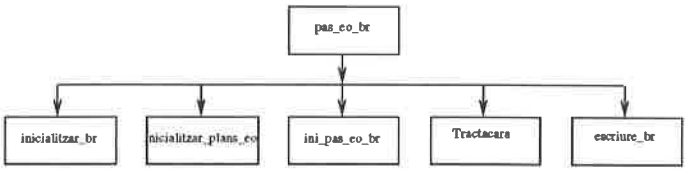

Figura 11: Arbre de les principals crides en la conversió d'un oct. extès a br.

Primer de tot *pas_eo_br* inicialitza el TAD br (*inicialitza_br*) i l'estructura auxiliar que requerirà per realitzar el procés de reconstrucció (*inicialitzar_plans_eo*). La rutina *ini_pas_eo_br*, realitza un recorregut per l'octree accedint als nodes vèrtexs i omplint per cada node l'estructura auxiliar amb l'informació del vèrtex. Tot seguit, ja es disposa de tots els vèrtexs i plans existents en el sòlid i de tots els vèrtexs que es troben sobre un determinat pla, per tant per completar el

model br tans sols és necessari crear les relacions topològiques que enllacen els vèrtexs situats sobre un mateix pla per formar cares.

L'acció *Tractacara* per cada llista de plans realitza un procés de reconstrucció de polígons amb els seus vèrtexs.

És important dir, que en el procés de reconstrucció del model br no es realitzen càlculs geomètrics, tans sols s'utilitzen relacions implícites en el model octree extès. Per tant, la robustesa de l'algorisme queda garantitzada.

A.5 Mòdul de compactació d'un octree extès

La compactació d'un octree extès consisteix en trobar un nou octree més concís, és a dir, amb menor nombre de nodes, que modeli el mateix sòlid.

El procés de compactació es basa en un recorregut en postordre de l'arbre, compactant primer els nodes de nivell inferior i posteriorment, si és possible, el node interior *gris*. Un node serà compactable quan la informació geomètrica que contenen els seus vuit descendents es pogui reduir a un node terminal de l'octree.

L'acció *compactar* inicialitza el procés de compactació accedint als vuit fills descendents del node arrel i per cada fill es crida a la funció *compacta_rec*. Quan aquesta acció es troba en el nivell inferior de l'arbre, procedeix a la compactació dels vuit fills d'aquest nivell. Per això, l'acció *join* realitza el procés de compactació dels nodes dos a dos, que es redueix a una simple casuística depenent del tipus dels nodes (veure taula 2). L'acció *ordena_fills* fixa l'ordre en que s'ha de realitzar el join entre germans. La funció *es_compactable* indica si vuit nodes germans poden ser a priori compactats, és a dir, si cap d'ells és gris.

O_1/O_2	Vèrtex	QVèrtex	Aresta	QAresta	Cara	Negre	Blanc
Vèrtex	Fals	$V_1 = V_2$	A∈QV	QA∈V	C∈V	Cert	Cert
QVèrtex	-	$QV_1 = QV_2$	A∈QV	QA∈QV	C∈QV	Cert	Cert
Aresta	-	-	*	*	*	Cert	Fals
QAresta	-	-	-	*	*	Cert	Cert
Cara	-	-	-	-	*	Cert	Fals
Negre	-	-	-	-	-	Cert	Fals
Blanc	-	-	-	-	-	-	Cert

Taula 2: Les caselles marcades amb * són aquelles que pot canviar el tipus del node resultant.

A.6 Mòdul de visualització d'un octree extès

El mòdul *visualització* conté totes les rutines auxiliars de la visualització.

Com hem dit en 2.2.2, en aquesta funció es presupossa que la camera està totalment definida i que la finestra es troba ja mapejada.

L'organigrama d'aquest procés es mostra en la figura 12. Primer de tot, veurem quins són els paràmetres que afecten a la visualització d'un octree extès:

- funció il·luminació: especifica el model d'il·luminació emprat en el càlcul del color.
- taca: indica el tamany del node per sota del qual els nodes terminals no es visualitzaran o bé s'aproxima la seva projecció per un polígon standard (projecció node (cub), acció *tacar*).
- llum fixa: booleà que indica si el focus de llum es situa d'acord a la posició de l'observador. S'ha escollit posar-ho 15° a l'esquerra i a 15° per sobre d'ell.
- color base: color difus del sòlid modelat via l'octree extès. No és res més, que un index a la paleta emprada en la visualització.
- draw_marc: booleà que indica si es pinta el contorn dels nodes.
- retallat:El paràmetre *clipping* ens indica si cal realitzar el retallat dels nodes que no són projectats en el window. Amb això volem dir que si la projecció a 2D del node (cub) es troba fora del window, no relitzarem la recostrucció del node. Aquest paràmetre serà cert quan estem realitzant un zoom, per tal de fer més ràpida la visualització.

l'acció *inicialitzar_visualitzacio*, com diu el seu nom inicialitza les estructures de dades requerides per la visualització. A continuació es realitza un recorregut en preordre dibuixant la reconstrucció de l'interiro dels nodes cara, aresta, vèrtex, quasivèrtex i gris mínim. La rutina *dibuixa* utilitza un model d'il·luminació especificat en la funció *funcio_iluminacio* per calcular el color de cada cara en funció de la seva orientació. L'acció *pipeline* aplica la transformació món-dispositiu a cadascun dels polígons 3D a visualitzar. S'ha d'observar que l'ordre en el processament dels fills d'un node gris depen de l'octant en que es troba la direcció que apunta cap a l'observador, segons indica la funció *ordre*.

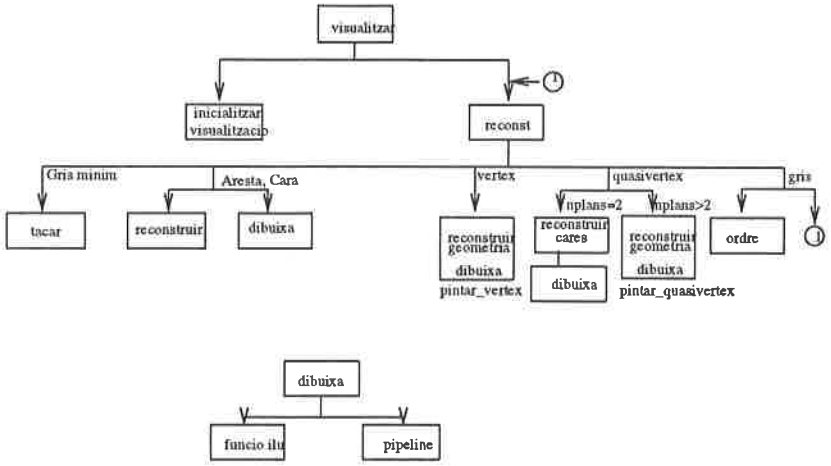

Figura 12: Arbre de les principals crides en la visualització d'un oct. extès.

A.7 Mòdul auxiliar de rutines geomètriques

El mòdul *geom* està format per totes les rutines geomètriques del paquet octree extès. Existeixen rutines de:

- Rutines de classificació punt-sòlid.
- Rutines de classificació respecte un cub.
- Càlcul d'interseccions.
- Càlcul punts representants dels nodes.
- Subdivisió d'un node extès.

A.8 Detalls per a la seva utilització

La classe octree extès treballa amb l'arbre de plans "genèric", és a dir, utilitza la classe *arbred* per donar llibertat a l'usuari d'aquesta classe en la elecció del tipus de plans amb els que vol treballar (veure secció 3). És per això que cal inicialitzar la variable *arbred* **arbre_d* amb algun dels arbres de plans oferts (arbred_pr arbred_pd).

Una altra qüestió es la inicialització de la paleta emprada en la visualització dels octrees. La visualització d'un octree presupossa que s'ha inicialitzat una paleta, i utilitza la funció *transforma_c_logic_fisic* per pintar en una finestra [Gar97].

Altres variables que cal inicialitzar per treballar amb el paquet són:

- nivell_màxim: grau de subdivisió màxim pels octrees.
- min_esc: tamany aresta del cub de resolució de l'arbre màxima.
- les variables de visualització (veure secció A.6) i l'entorn gràfic [Gar97].