Acceleració RT

Carlos Andujar

Desembre 2012

Continguts

- **Control Adaptatiu de la Recursivitat**
- Acceleració de la intersecció raig primari -escena
- Acceleració intersecció raig-escena
- Jerarquia de Volums Englobants
- Subdivisió uniforme
- Octrees
- Partició binaria de l’espai (*BSP trees*)

Control adaptatiu recursivitat

```
acció rayTracing
   per i en [0..w-1] fer
   per j en [0..h-1] fer
      raig:=calcularRaigPrimari(i, j, camera);
      color:=traçarRaig(raig, escena, μ, 0, 1);
      setPixel(i, j, color);
   fper
   fper
faccio
```

Nivell

Contribució

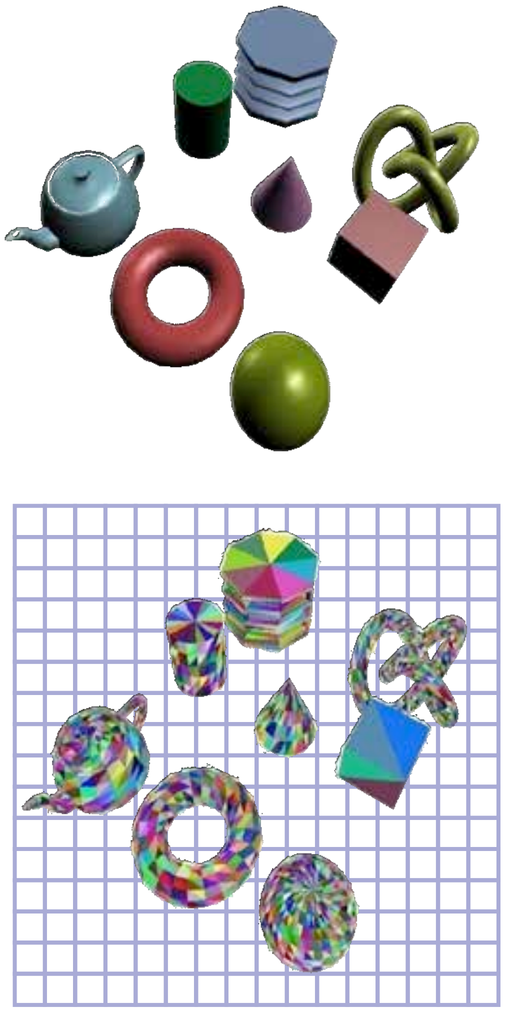
Control adaptatiu recursivitat

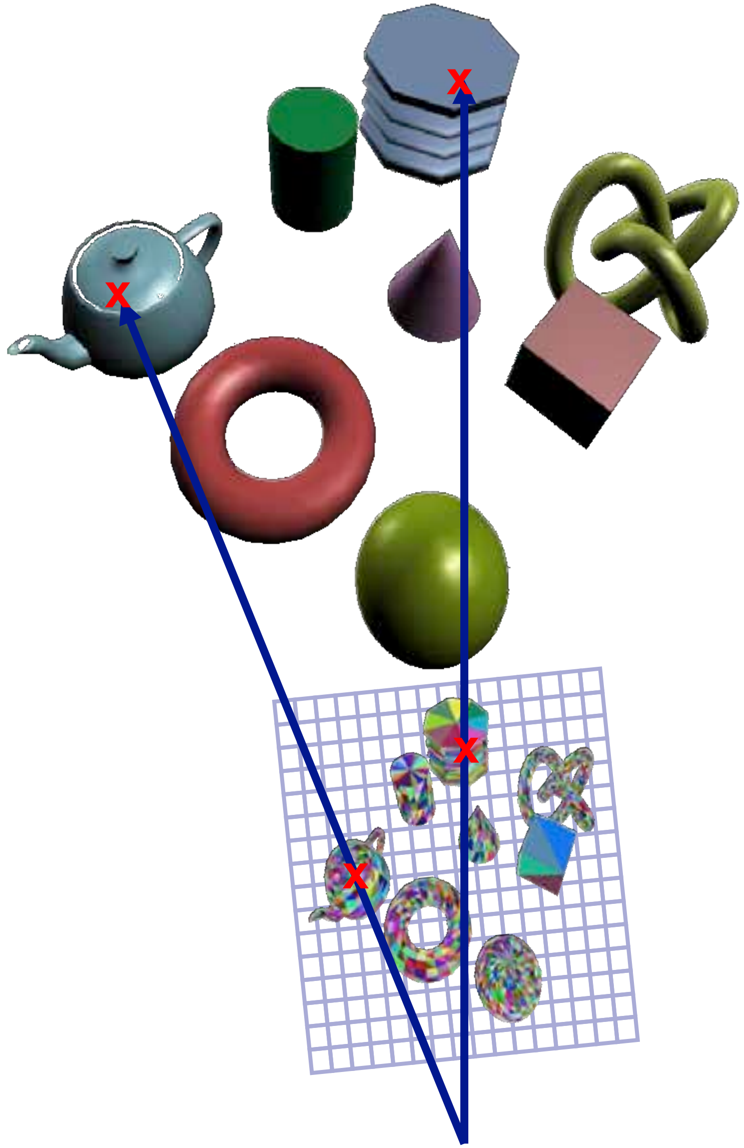
```
funció traçarRaig(raig, escena, μ, niv, contrib)
  si niv < NIVELL_MAX  ∧  contrib > MIN_CONTRIB llavors
    info:=calculaInterseccio(raig, escena)
    si info.hiHaInterseccio() llavors
      color:=calcularIF(info,escena);
      si esReflector(info.obj) llavors
         raigR:=calculaRaigReflectit(info, raig)
         color:=color+ K_R·traçarRaig(raigR, escena, μ, niv+1, K_R·contrib)
      fsi
      si esTransparent(info.obj) llavors
         raigT:=calculaRaigTransmès(info, raig)
         color:=color+ K_T·traçarRaig(raigT, escena, info.μ, niv+1,K_T·contrib)
      fsi
    sino color:=colorDeFons
    fsi
  sino color:=Color(0,0,0);  // o colorDeFons
  fsi
  retorna color
ffunció
```

Continguts

- Control Adaptatiu de la Recursivitat
- **Acceleració de la intersecció raig primari**
- Acceleració intersecció raig-escena
- Jerarquia de Volums Englobants
- Subdivisió uniforme
- Octrees
- Partició binaria de l'espai (*BSP trees*)

Acceleració intersecció raig primari





Continguts

- Control Adaptatiu de la Recursivitat
- Acceleració de la intersecció raig primari
- **Acceleració intersecció raig-escena**
- Jerarquia de Volums Englobants
- Subdivisió uniforme
- Octrees
- Partició binaria de l'espai (*BSP trees*)

Acceleració intersecció raig-escena

- L'algorisme bàsic d'intersecció raig-escena requereix comprobar la intersecció amb tots els objectes de l'escena → cost O(N)
- Hi ha una sèrie de tècniques que permeten assolir una **búsqueda sublinial** mitjançant algun tipus de **subdivisió.**

Tipus de subdivisions en gràfics

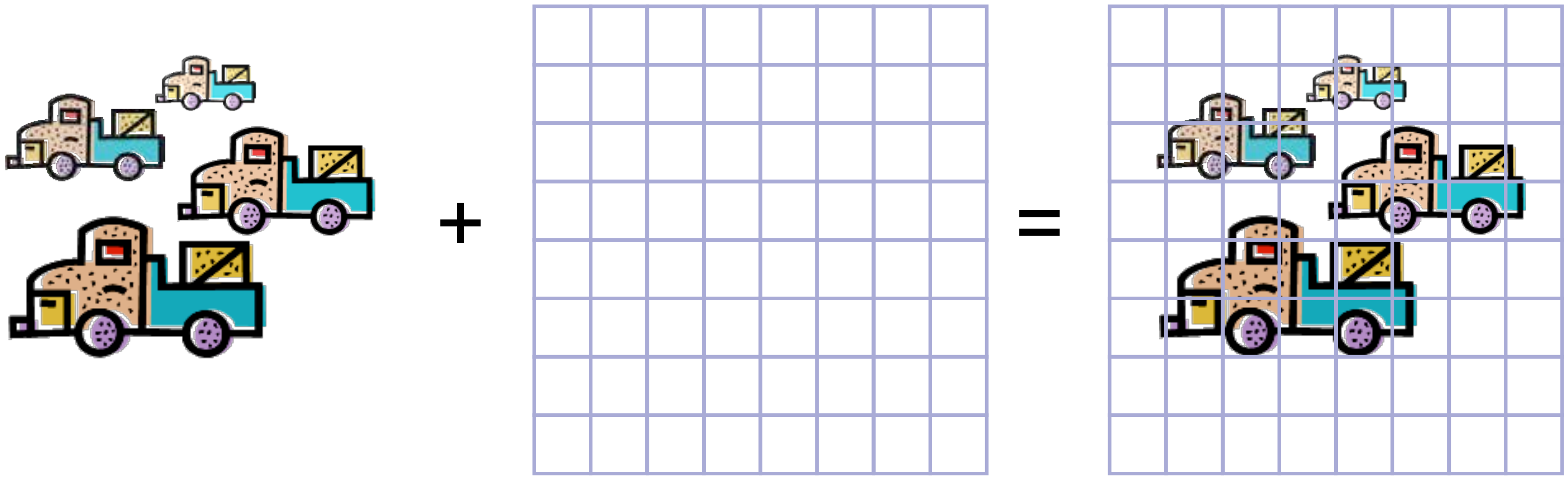
Subdivisió de **l'espai**

- Es **particiona l'espai** en subregions (cel·les)
- Un **punt de l'espai pertany a una única cel·la fulla**
- Un **objecte pot ocupar més d'una cel·la fulla**
- Exemples: *Voxelització, octrees, BSP, kd-tree...*

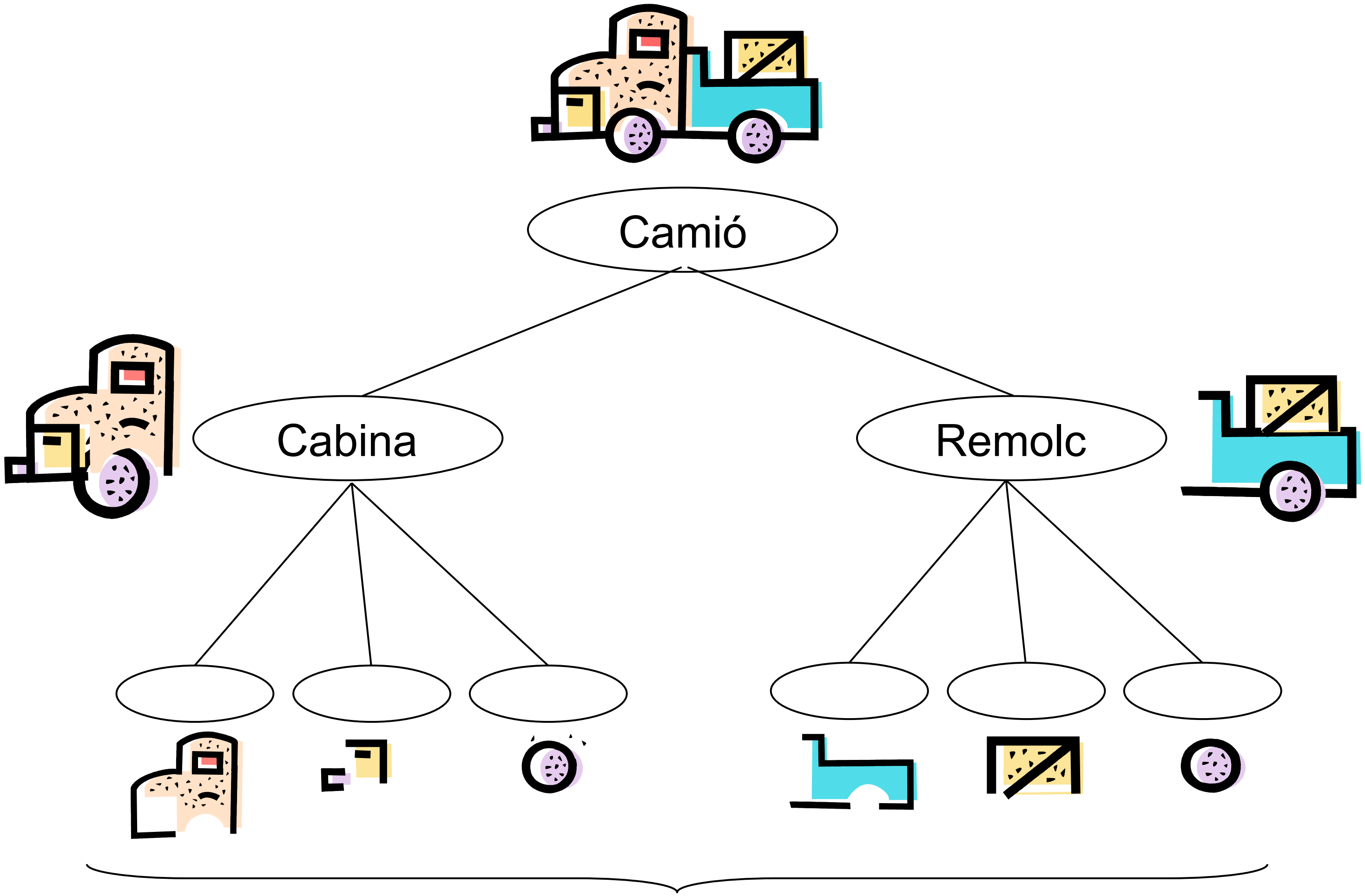
Subdivisió dels **objectes**

- Es **particionen els objectes** en grups; equivalent a agrupar objectes
- Un **objecte pertany a una única fulla**
- Un **punt de l'espai pot estar en múltiples fulles**
- Exemples: *jerarquies de volums englobants*

Exemple de subdivisió de l’espai



Exemple de subdivisió dels objectes



La geometria està repartida entre els nodes fulla de l'arbre

Acceleració intersecció raig-escena

Subdivisions més importants per Ray-tracing:

- **Jerarquia de Volums Englobants** (*Bounding Volume Hierarchies*)
- **Subdivisió Uniforme de l'Espai** (*Uniform Space Subdivision*)
- **Arbres octals** (*Octrees*)
- **Partició Binaria de l'Espai** (*Binary Space Partition, BSP*)

Continguts

- Control Adaptatiu de la Recursivitat
- Acceleració de la intersecció raig primari
- Acceleració intersecció raig-escena
- **Jerarquia de Volums Englobants**
- Subdivisió uniforme
- Octrees
- Partició binaria de l'espai (*BSP trees*)

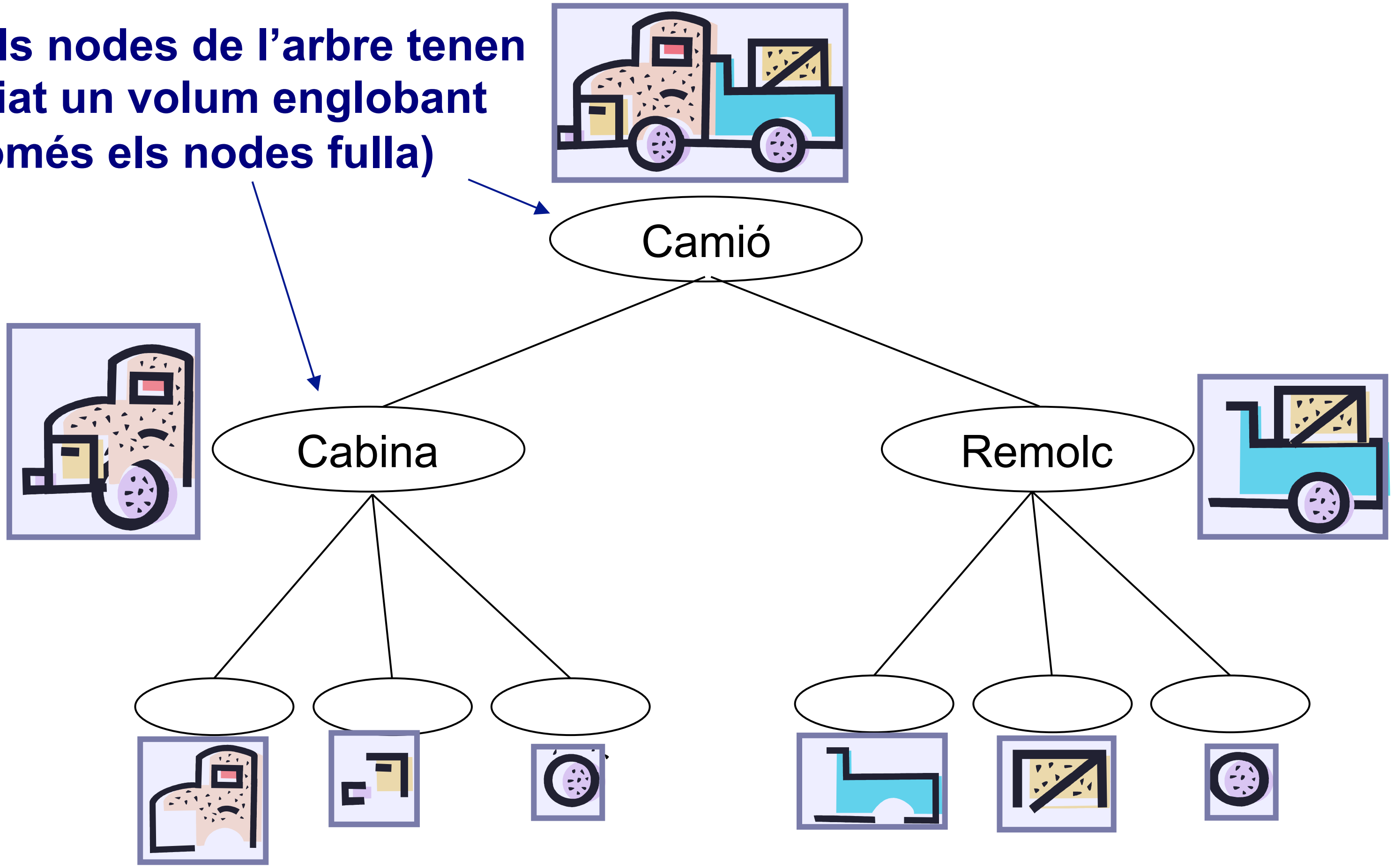
Jerarquies de volums englobants

Bounding Volume Hierarchies (BVH)

- Associar un volum englobant a cada **objecte individual** ens permet descartar ràpidament els casos de no-intersecció amb **l'objecte**.
- Si **agrupem objects propers i calculem el volum englobant del grup**, podrem descartar ràpidament casos de no-intersecció amb tot el grup.

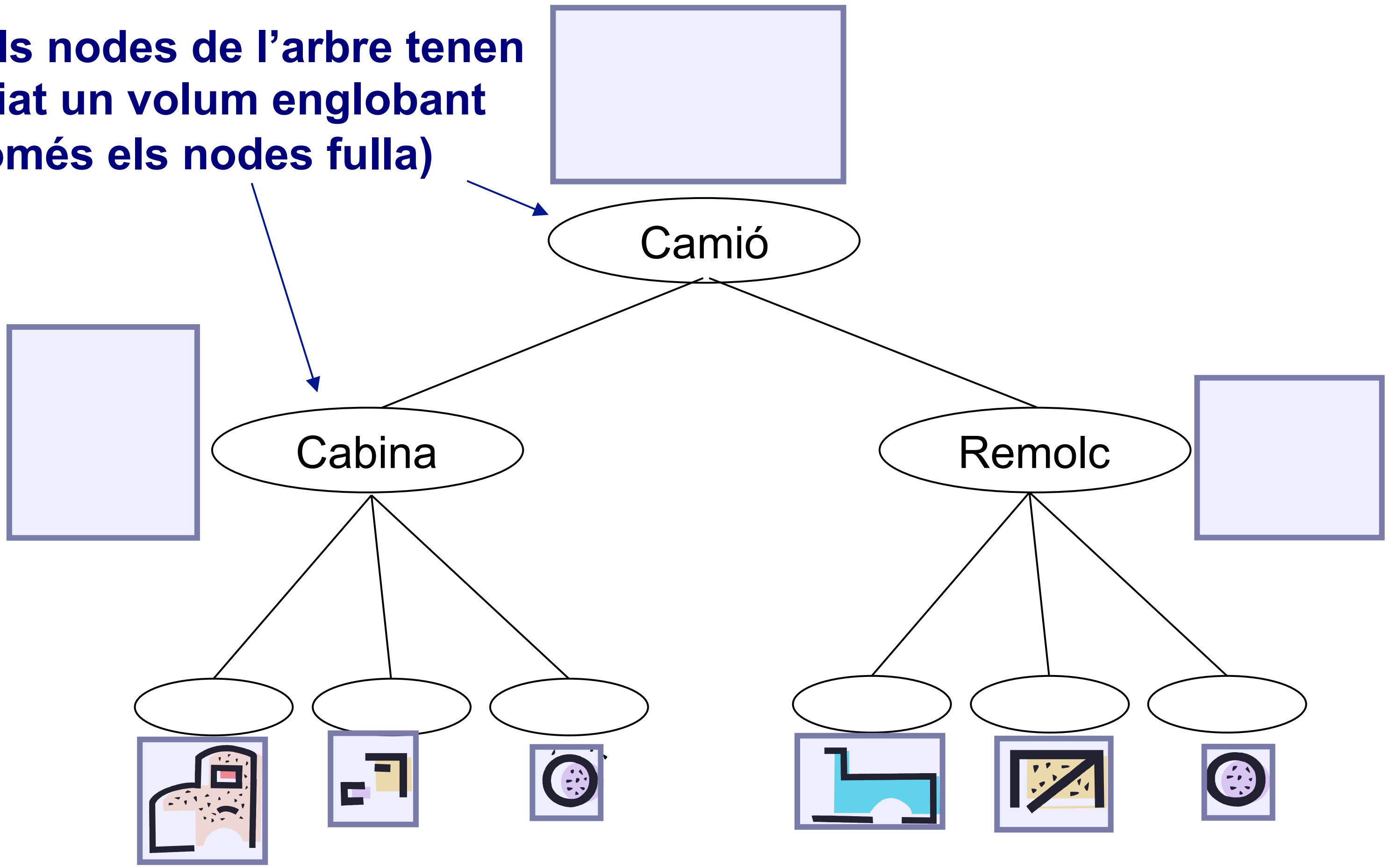
Exemple de subdivisió dels objectes

Tots els nodes de l'arbre tenen associat un volum englobant (no només els nodes fulla)

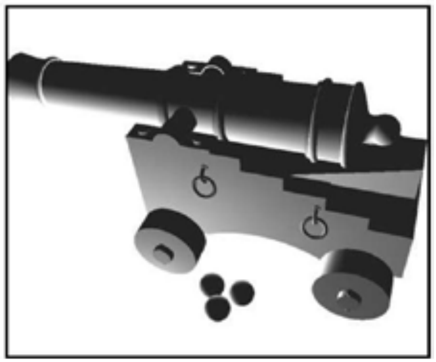


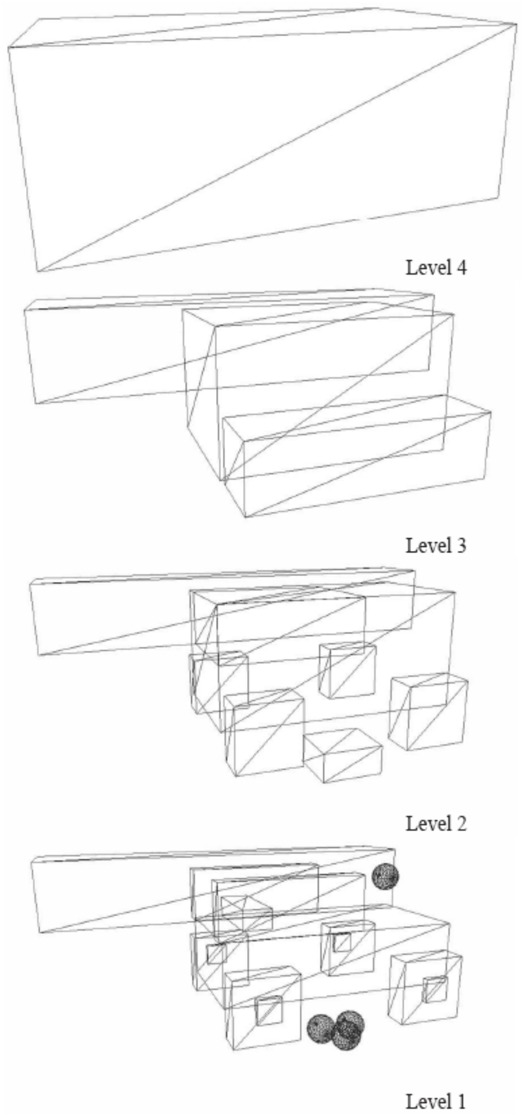
Exemple de subdivisió dels objectes

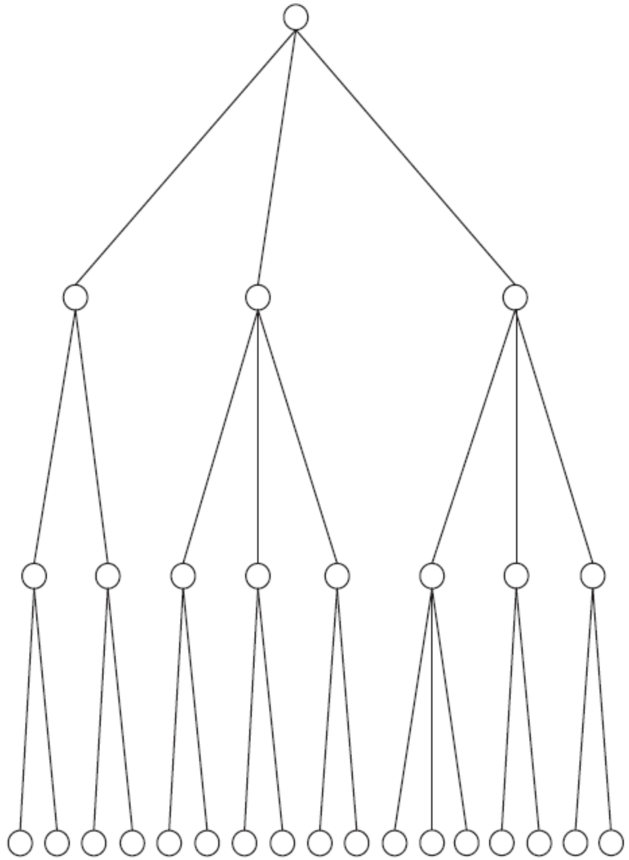
Tots els nodes de l'arbre tenen associat un volum englobant (no només els nodes fulla)



Exemple 3D

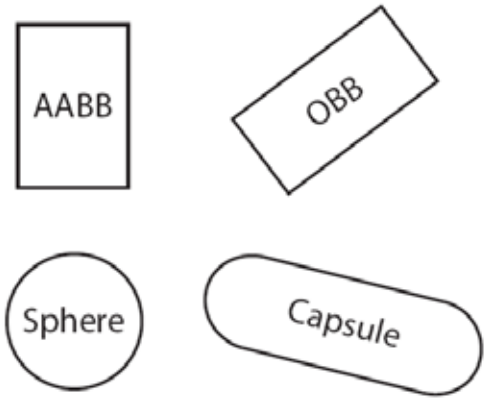




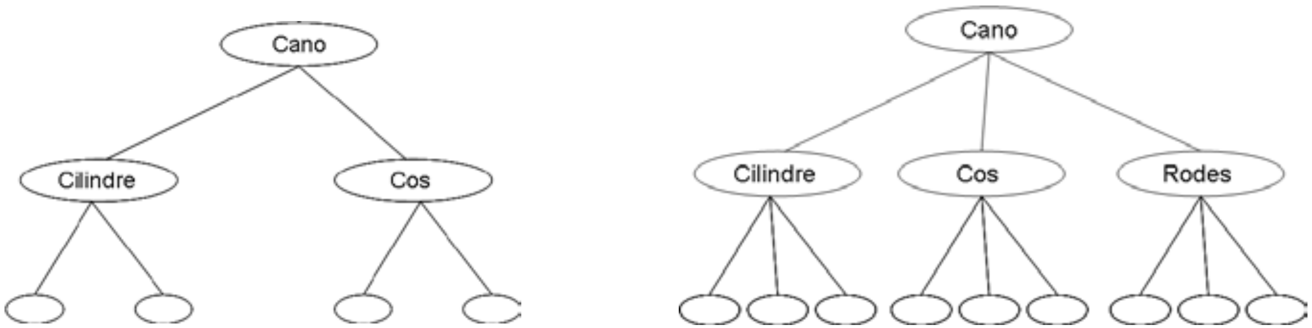


Classificació de BVH

- Segons el tipus de volum englobant:



- Segons l'aritat de l'arbre (binari, n-ari...)

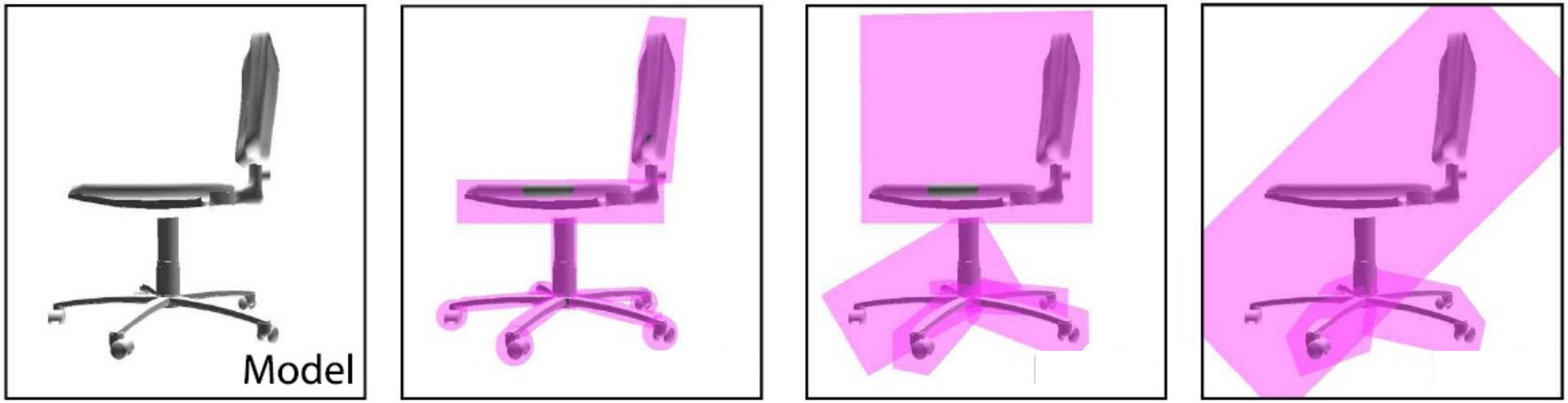


Representació de l'arbre

```
class Node : public Surface
{
  virtual bool hit(ray, λmin, λmax, hitRecord);
  ...
  Node *left, *right;
  Capsa capsa;  // o qualsevol volum englobant
  vector<Surface*> objectes;
};
```

Intersecció raig-escena amb BVH

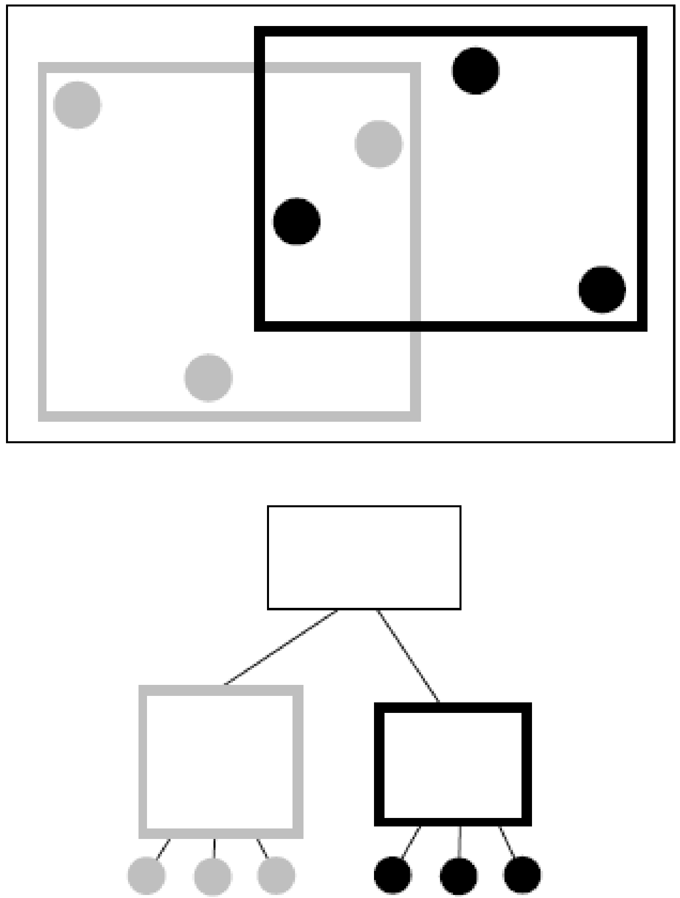
Si el raig no interseca el volum englobant d'un node, no cal comprovar la intersecció amb els fills



Intersecció raig-escena amb BVH

Propietats:

- Els volums associats als fills d'un node poden solapar-se.
- Si un objecte està dins el volum associat a un node no necessàriament pertany a un descendent del node.
- Si el raig interseca el node, cal comprovar la intersecció amb **tots els fills.**



Intersecció raig-escena amb BVH

Algorisme recursiu (arbre binari):

- Comprovar intersecció amb la capsa del node. Si no intersecta la capsa → **el raig no intersecta el node**
- Altrament, comprovar la intersecció amb el subarbre esquerra i el subarbre dret, i retorna la intersecció més propera (recursiu).

Intersecció raig-escena amb BVH

funció Node::hit(raig, λ_{min}, λ_{max}, infoHit) **retorna booleà**

si capsa.hit(raig, λ_{min}, λ_{max}) **llavors**

si ésTerminal() **llavors**

<*comprova intersecció amb les primitives*>

altrament

hitLeft := left→hit(ray, λ_{min}, λ_{max}, infoLeft)

hitRight := right→hit(ray, λ_{min}, λ_{max}, infoRight)

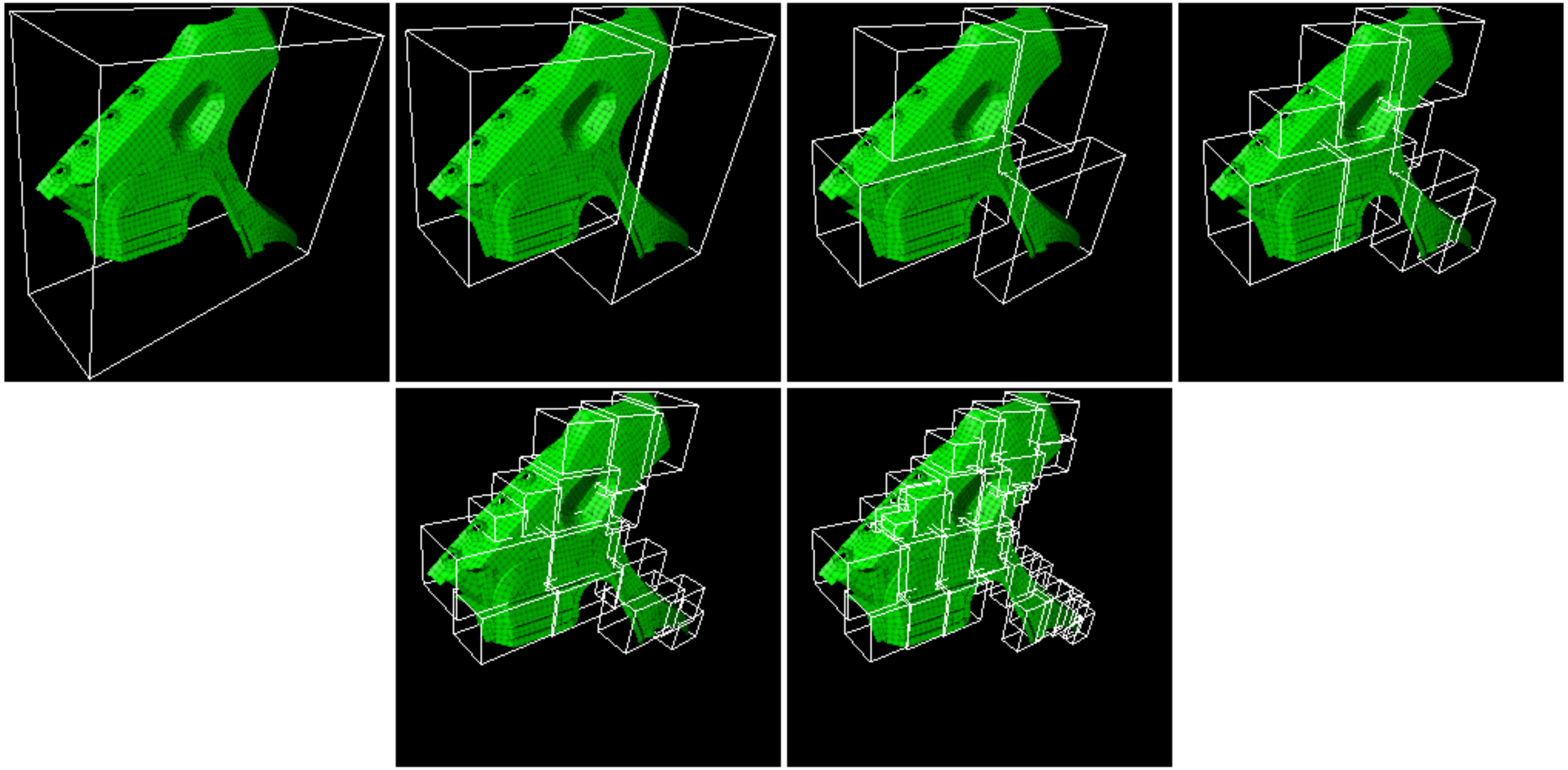
<retorna la intersecció més propera de les dues>

fsi

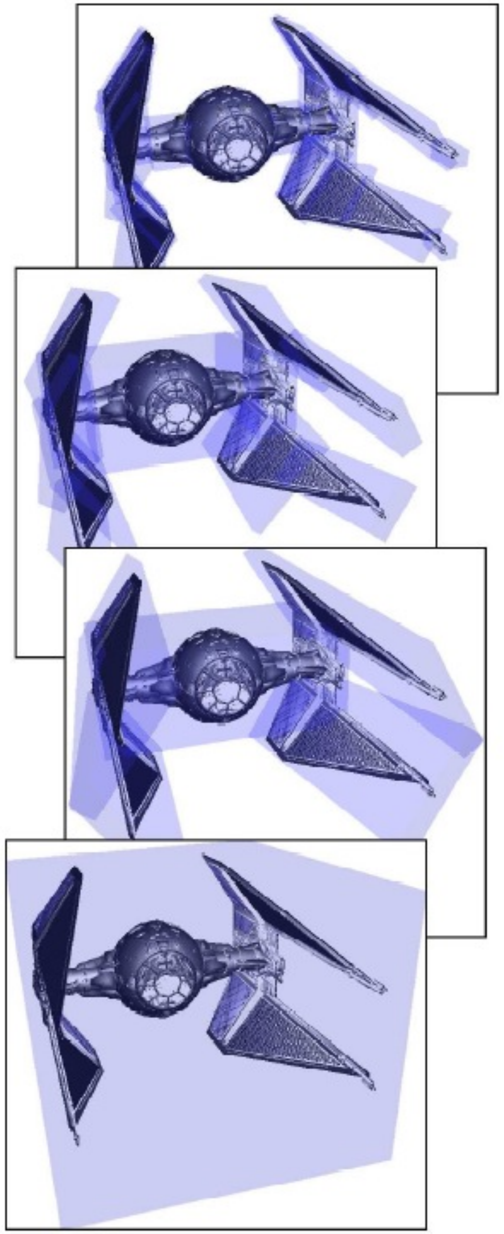
altrament retorna fals;

ffunció

Construcció jerarquia AABB



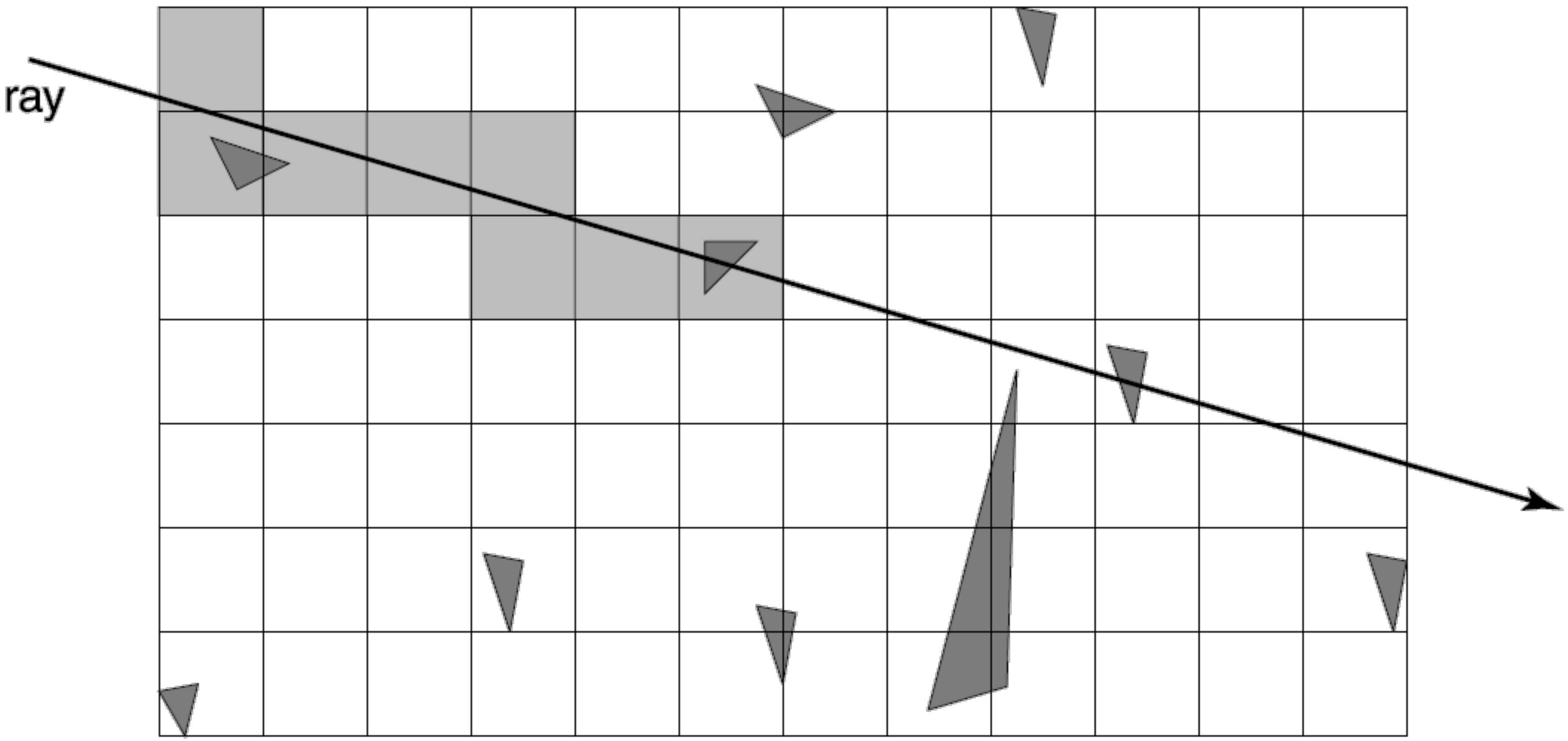
Un altre exemple



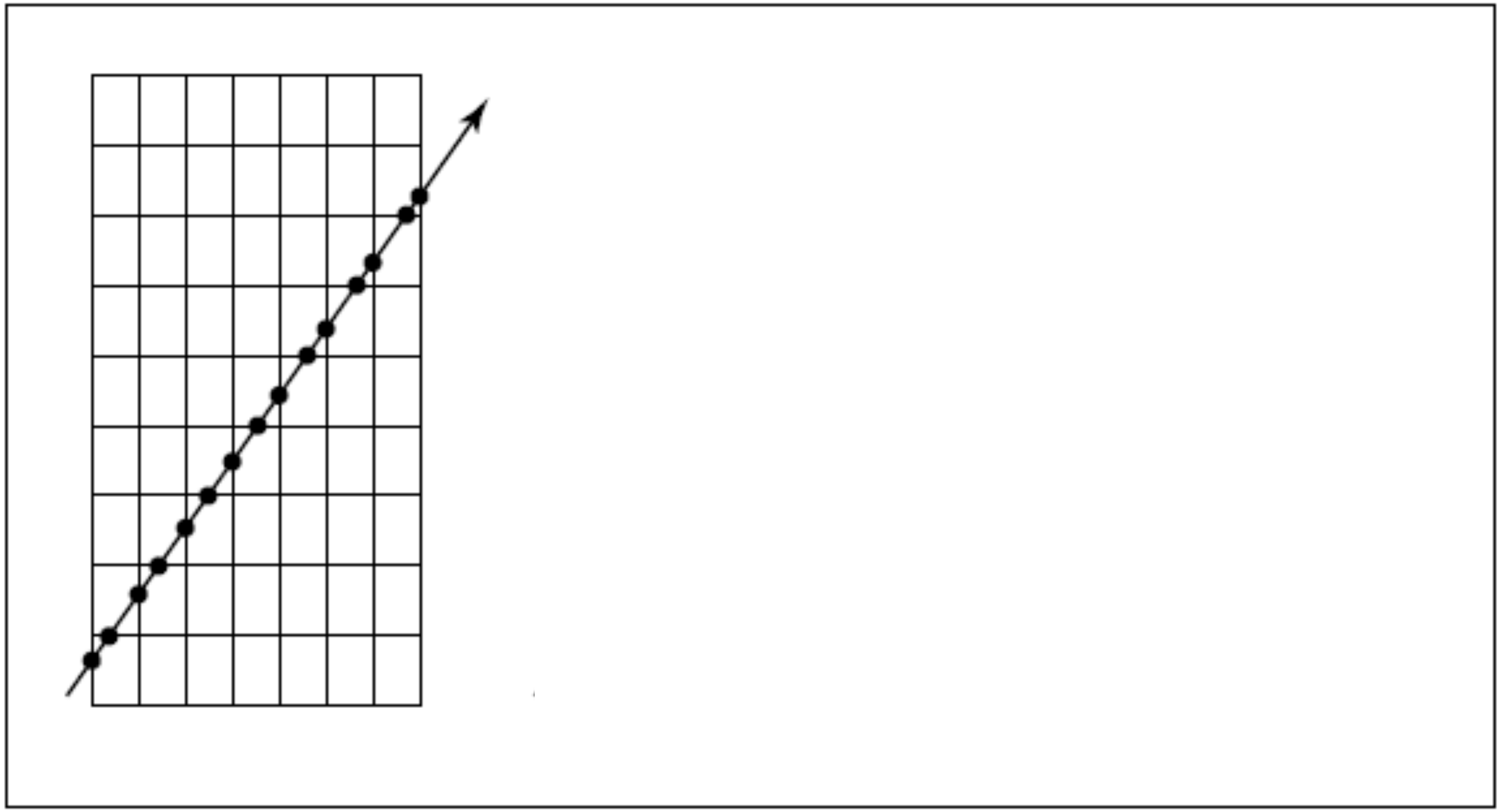
Continguts

- Control Adaptatiu de la Recursivitat
- Acceleració de la intersecció raig primari
- Acceleració intersecció raig-escena
- Jerarquia de Volums Englobants
- **Subdivisió uniforme (voxelització)**
- Octrees
- Partició binaria de l’espai (*BSP trees*)

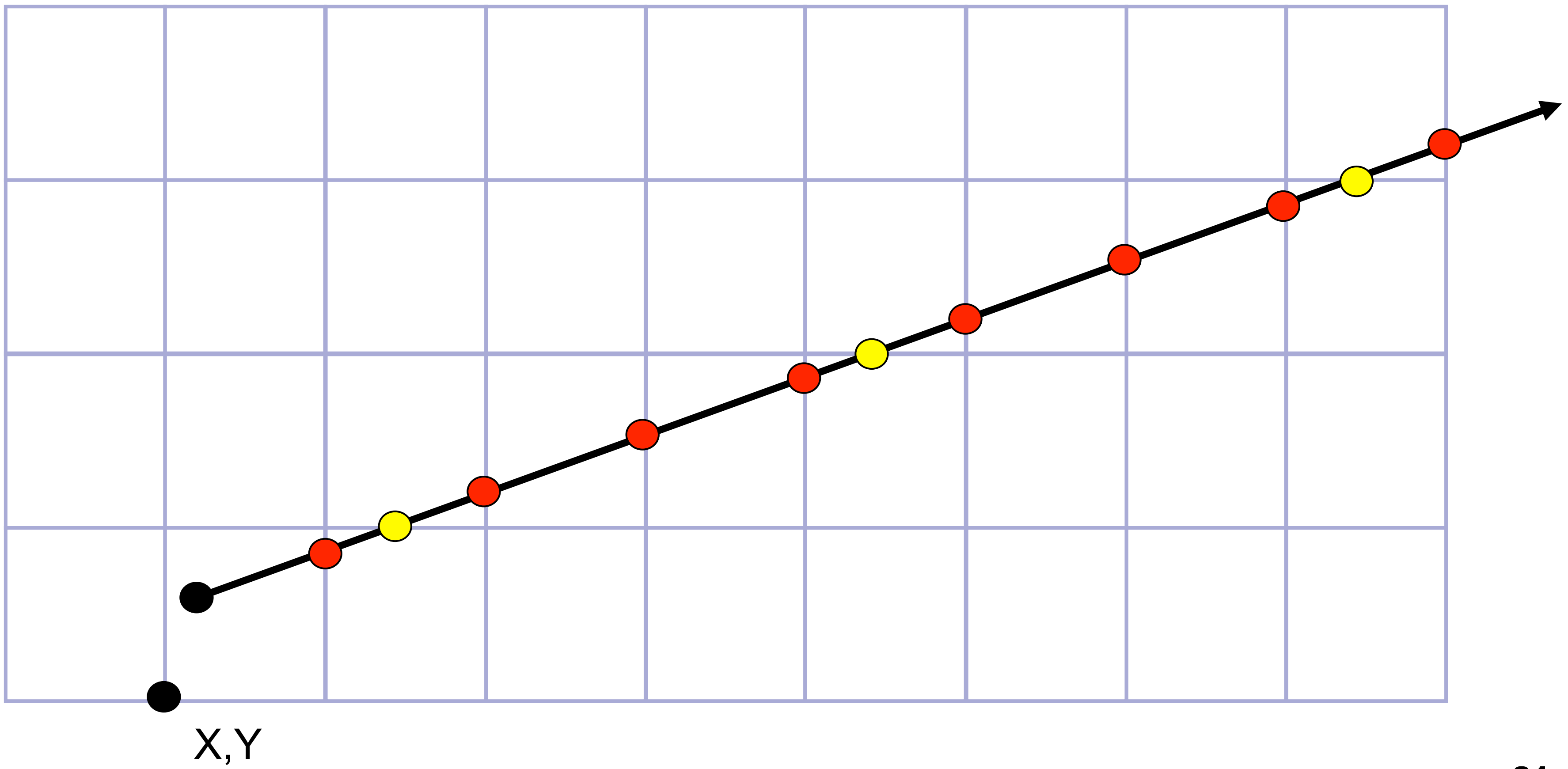
Subdivisió uniforme (voxelització)



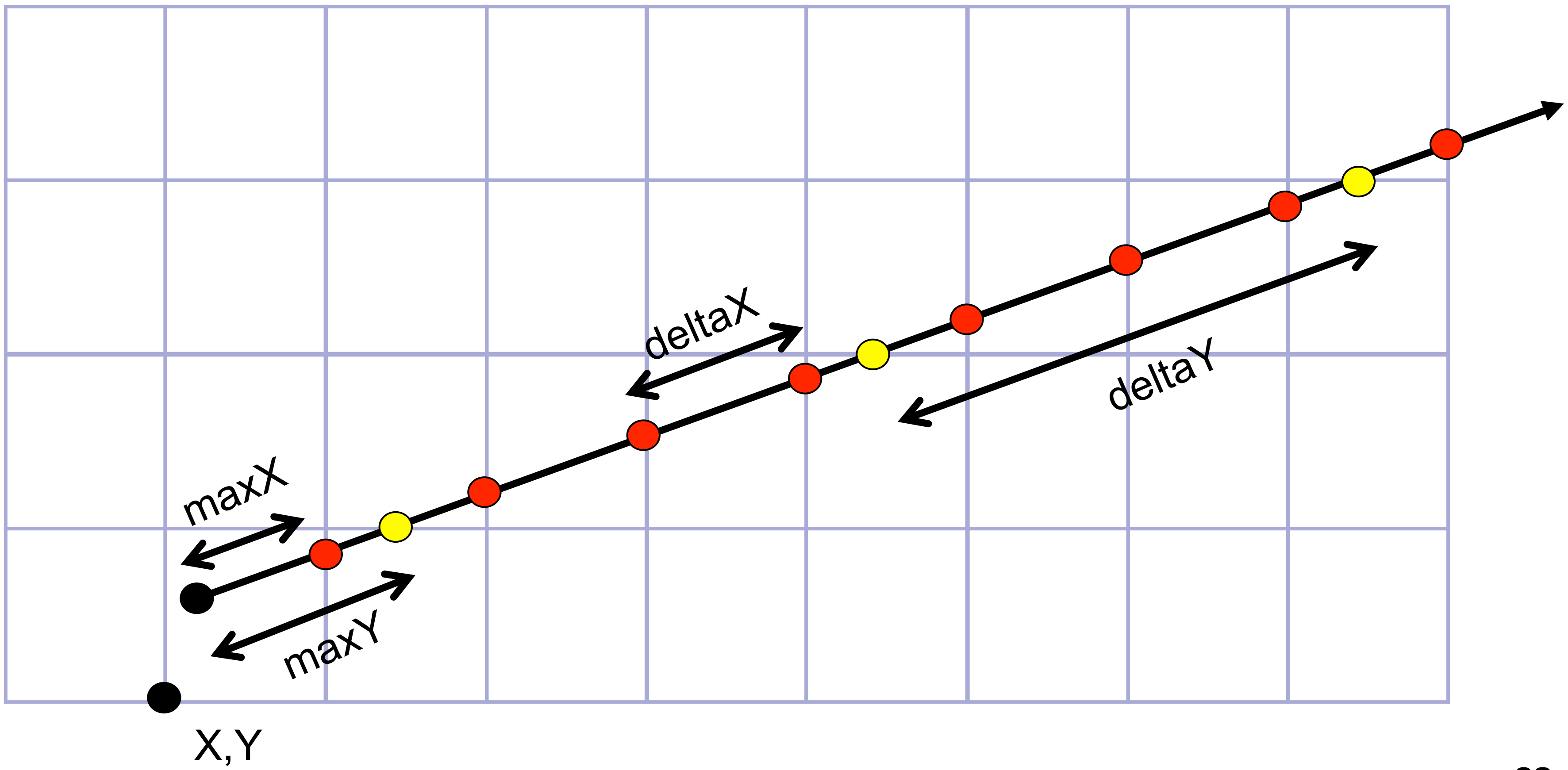
Recorregut dels voxels



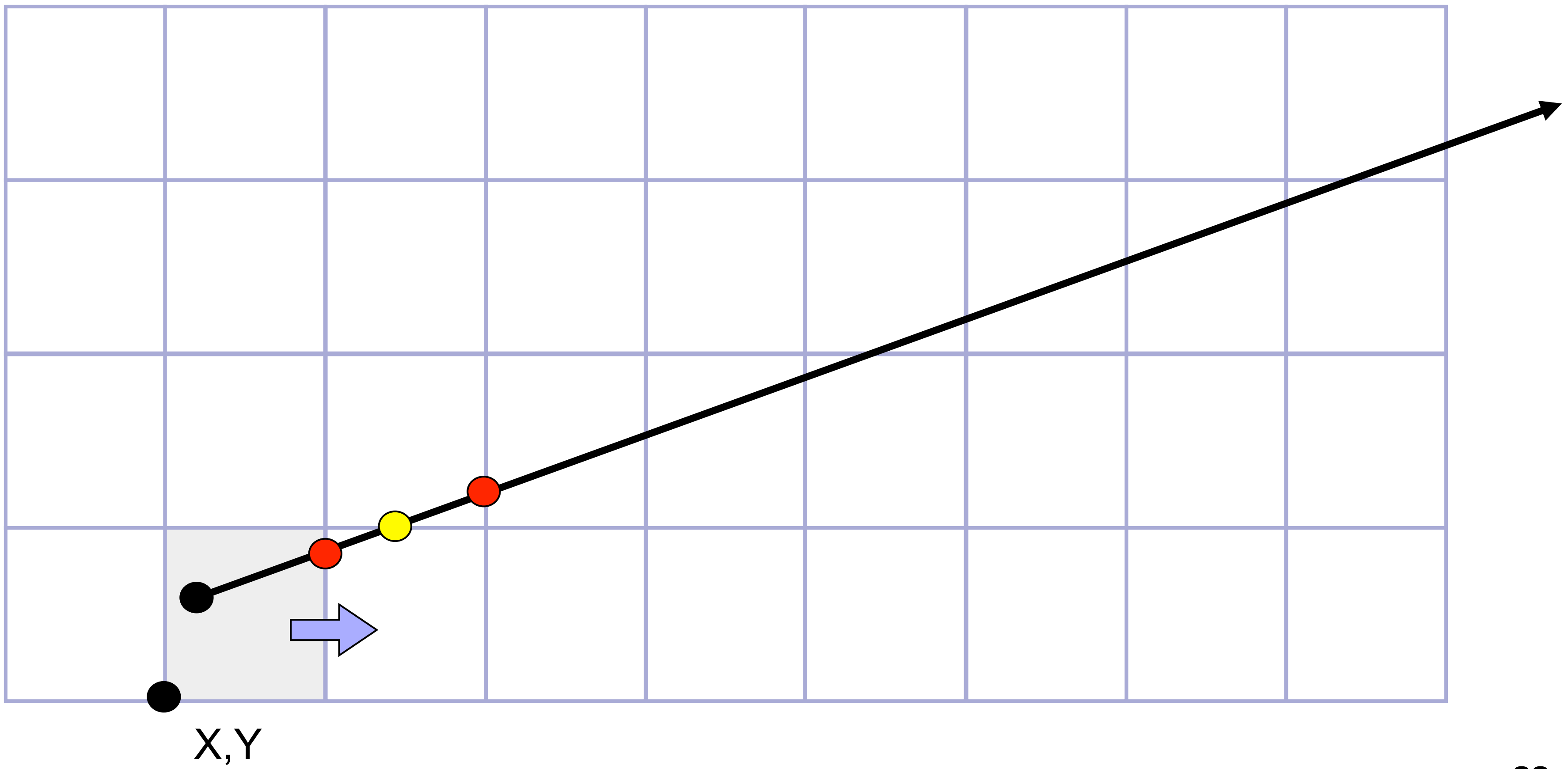
Recorregut DDA



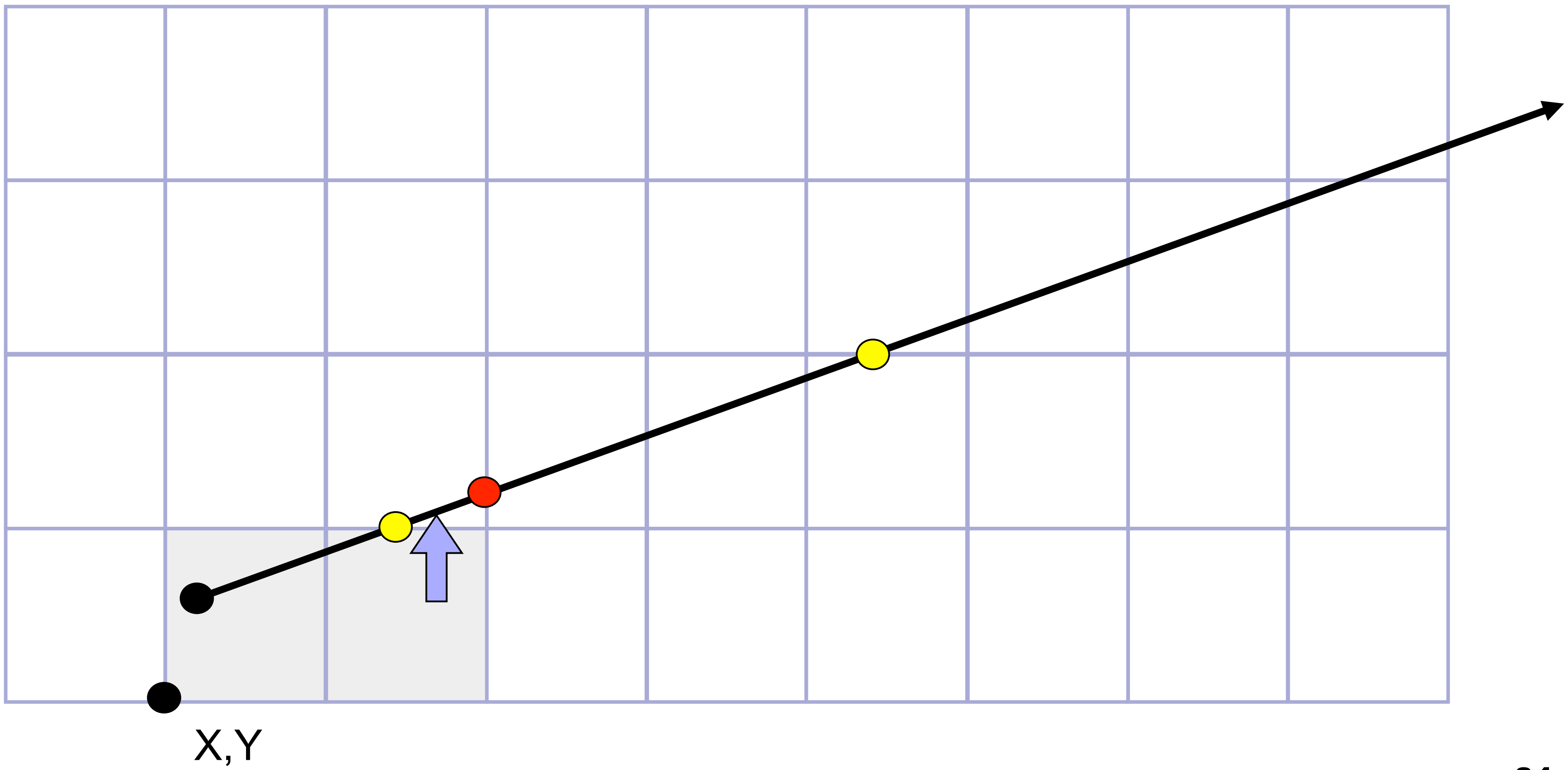
Recorregut DDA



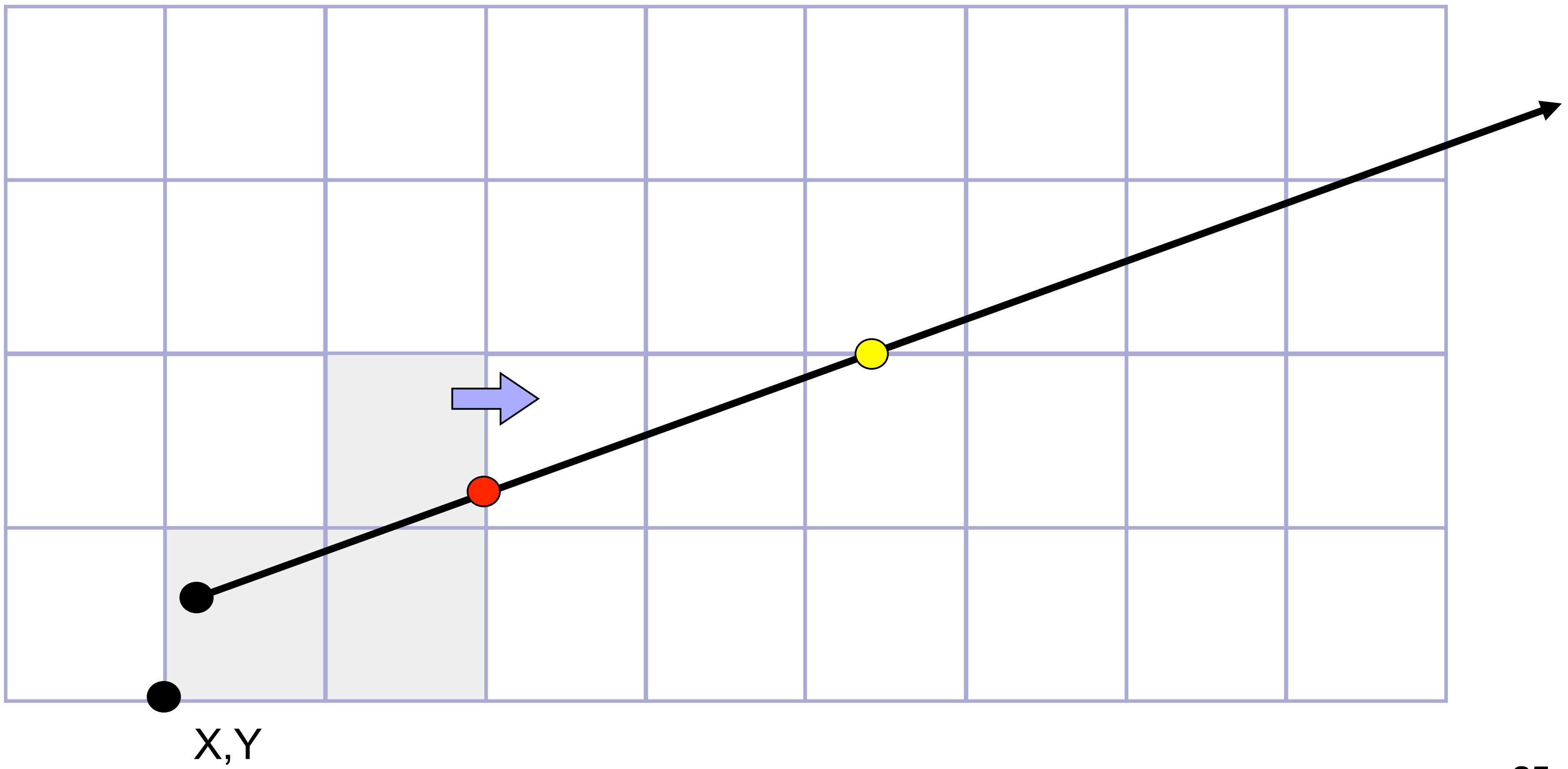
Recorregut DDA: Simulació



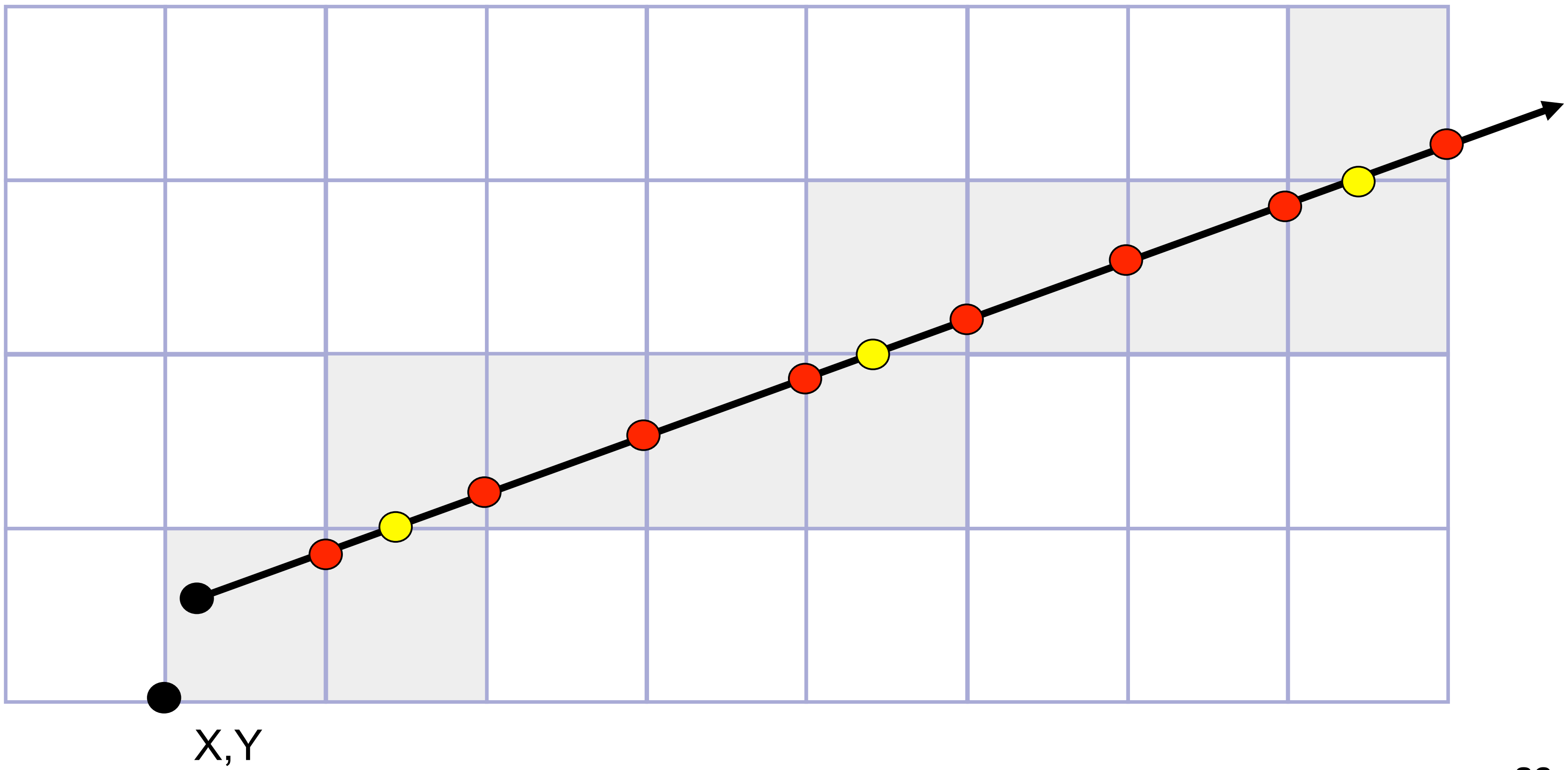
Recorregut DDA: Simulació



Recorregut DDA: Simulació



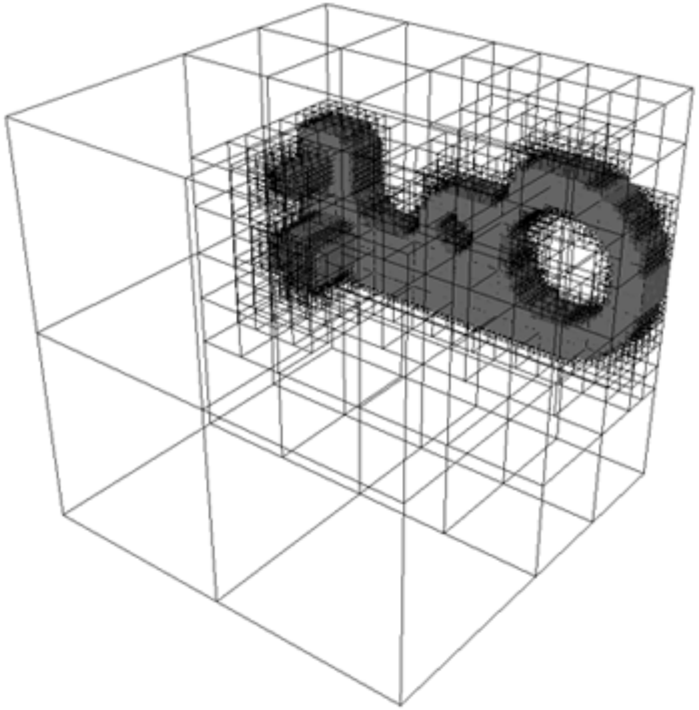
Recorregut DDA: Simulació



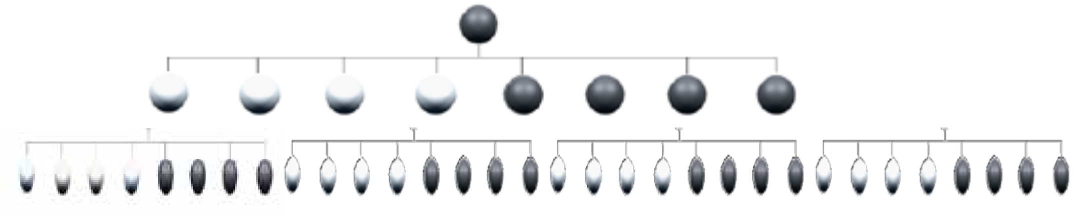
Continguts

- Control Adaptatiu de la Recursivitat
- Acceleració de la intersecció raig primari
- Acceleració intersecció raig-escena
- Jerarquia de Volums Englobants
- Subdivisió uniforme (voxelització)
- **Octrees**
- Partició binaria de l'espai (BSP)

Octrees



Subdivisió de l'espai



Estructura de l'octree

Octrees: representació

```
class OctreeNode : public Surface
{
    virtual bool hit(ray, λmin, λmax, hitRecord);
    ...
    OctreeNode* fills[8];  // apuntadors als 8 fills
    Surface *obj;  // NULL excepte en nodes terminals
};
```

Octrees: construcció

```
constructor OctreeNode(objectes, n, capsa)
 si #objectes < MAX_OBJ ó  n > MAX_DEPTH  llavors
         fills[0..7]:=NULL;
         obj := objectes;
 altrament
   per i en [0..7]
     vector<Surface*> objectesFill;
     objectesFill := <objectes que intersecten el node fill>
     fills[i] = new OctreeNode(objectesFill, n+1, octant(capsa,i) )
  fper
fconstructor
```

INTERSECCIÓ TRIANGLE-CAPSA

Intersecció triangle-capsa

- Tomas Akenine-Möller, *Fast 3D Triangle-Box Overlap Testing, Journal of Graphics Tools, 6(1), 2001*
- Codi font: http://jgt.akpeters.com/papers/AkenineMoller01/tribox.html

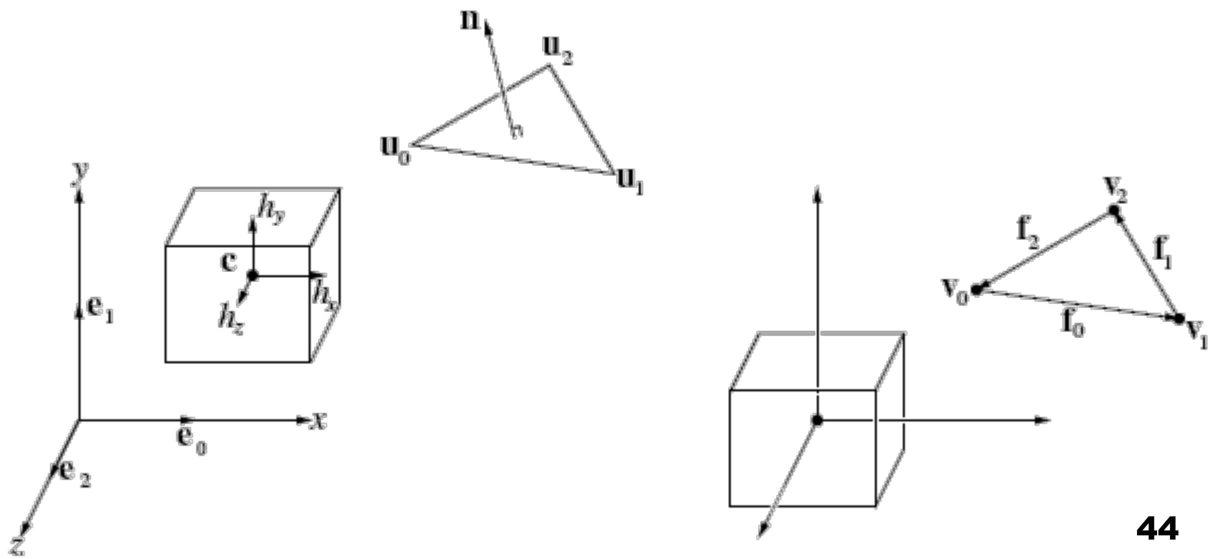
Intersecció triangle-capsa

Entrada:

- Capsa: centre **c** i semiarestes hx, hy, hz
- Triangle: vèrtexs (u0, u1, u2)

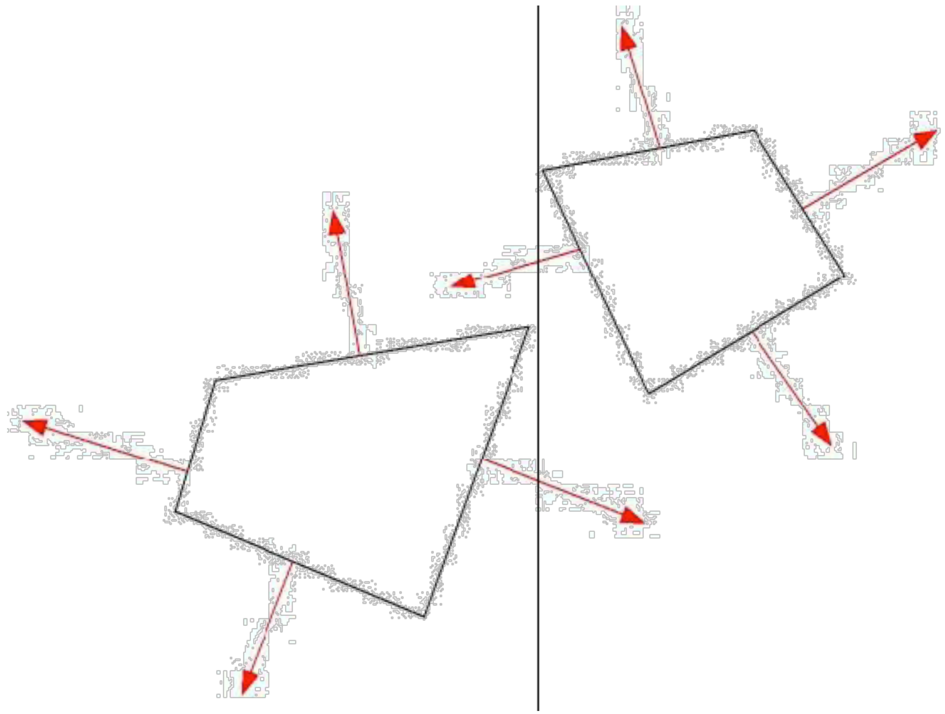
Inicialització:

- Es traslllada el triangle per portar el centre de la capsa a l'origen
- Triangle transformat: (v0, v1,v2)



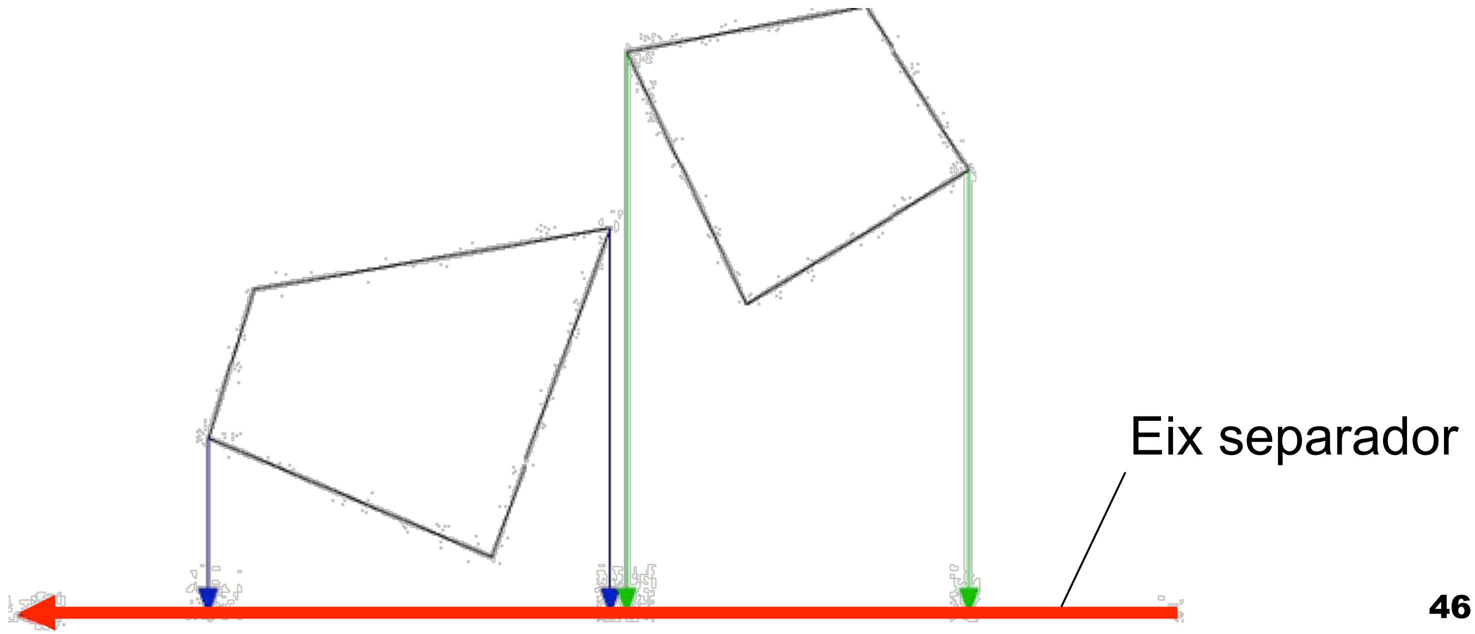
Separating Plane Theorem

- Dos objectes convexos A, B, no s'intersecten $\leftrightarrow$ existeix un pla separador Π tal que A està en un semiespai i B a l'altre semiespai.

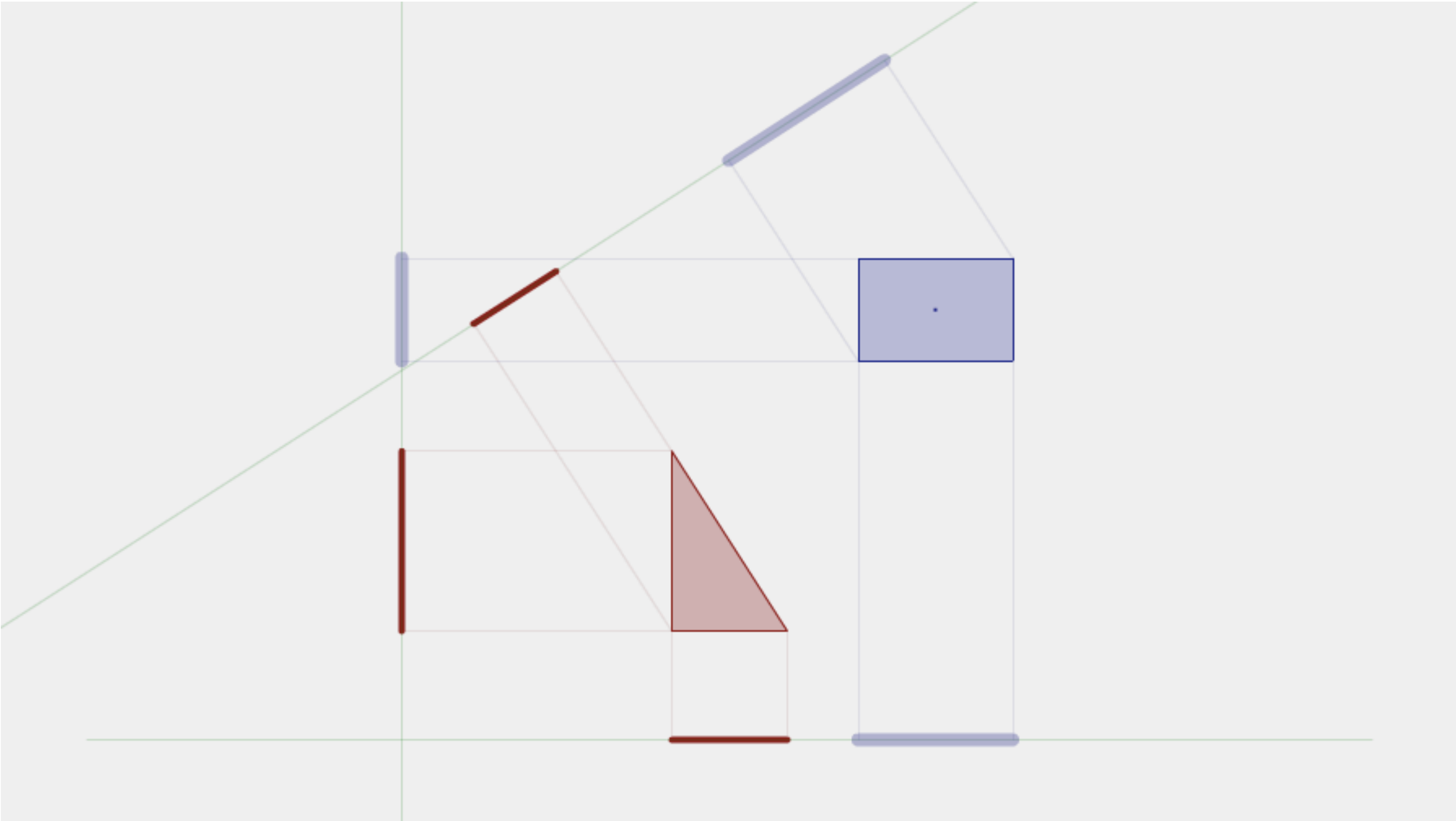


Separating Axis Theorem

- Sigui Π un pla separador de A i B. Qualsevol recta perpendicular al pla s'anomena **eix separador** perquè les projeccions ortogonals de A i B sobre aquesta recta són disjuntes.
- Existeix un eix separador ↔ A i B no s'intersecten.



Exemple 2D (triangle-capsa)



SAT per poliedres convexos

- Dos **poliedres convexos** A i B són disjunts ↔ un d'aquests eixos és separador**:**
  - Normal d'una cara de A
  - Normal d'una cara de B
  - Producte vectorial d'una aresta de A amb una aresta de B

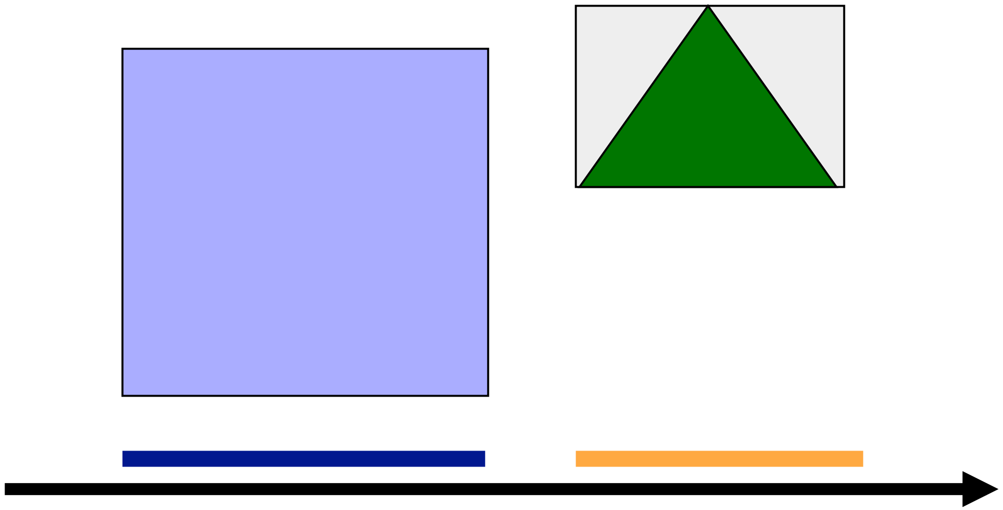
Intersecció triangle-capsa

- L'algorisme comprova tres tipus d'eixos (13 tests):
  a) Les tres normals de les cares del cub: (1,0,0), (0,1,0), (0,0,1) → **3 eixos**
  b) La normal del triangle (n_x, n_y, n_z) → **1 eix**
  c) El producte vectorial d'una aresta del cub (els tres eixos coordenats) amb una aresta del triangle (tres direccions) → **9 eixos**

- Si en algun moment es troba que l'eix és separador, l'algorisme retorna que **no** hi ha intersecció.

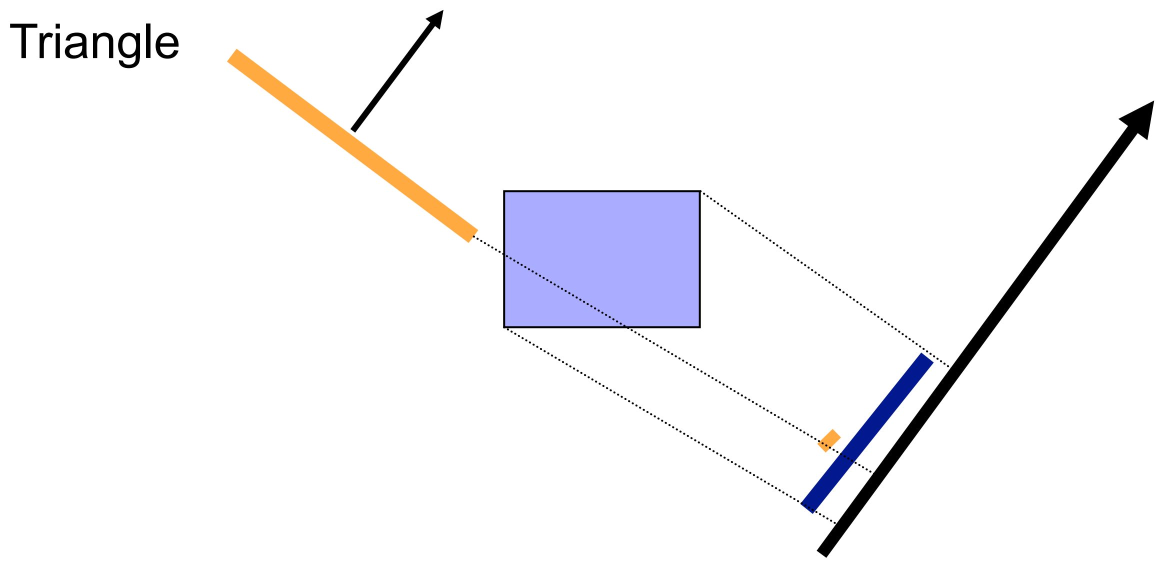
Test (a)

Pels eixos coordenats (1,0,0), (0,1,0), (0,0,1) n'hi ha prou amb considerar la capsa amb la capsa englobant del triangle.



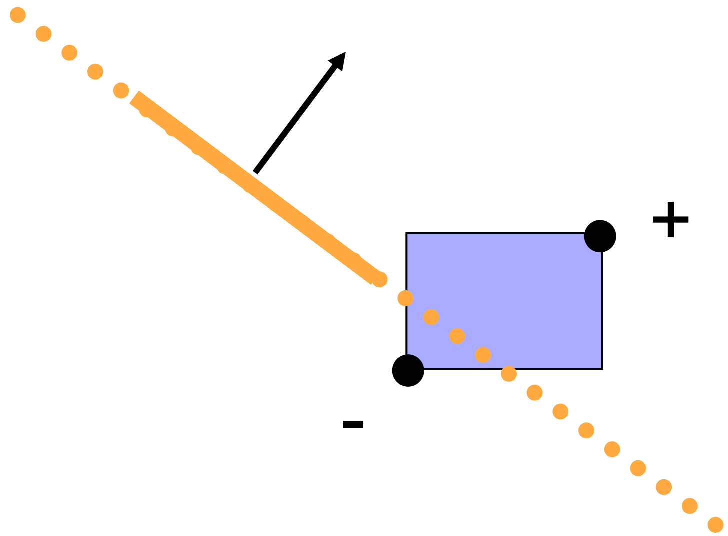
Test (b)

Per la normal del triangle (n_x, n_y, n_z), el test equival a comprovar si el pla del triangle intersecta la capsa.



Test (b)

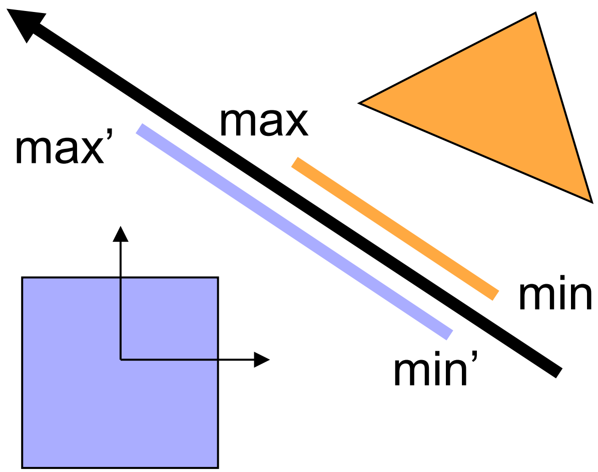
- Es tracta doncs d’un test d’intersecció pla-AABB.
- N’hi ha prou amb considerar els dos vèrtexs de la AABB (els de la diagonal més aliniada amb la normal del triangle). Si els vèrtexs pertanyen a semiespais diferents → el pla intersecta la capsa → l’eix no és separador



Test (c)

Pels nou eixos restants, els passos a fer són:

- Projectar els 3 vèrtexs del triangle sobre l'eix
- Obtenir el mínim i el màxim d'aquests valors
- Projectar els 8 vèrtexs de la capsa sobre l'eix
- Obtenir-ne el mínim i el màxim
- Comprovar el possible solapament dels dos intervals (1D, trivial)

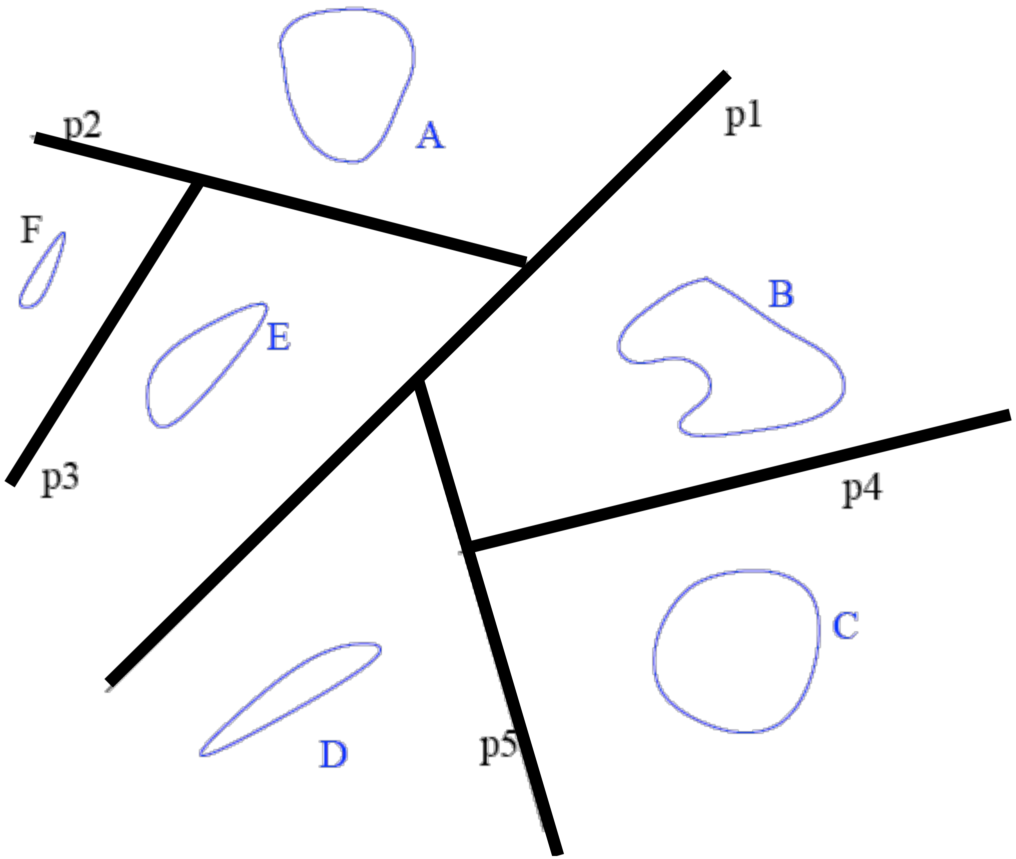


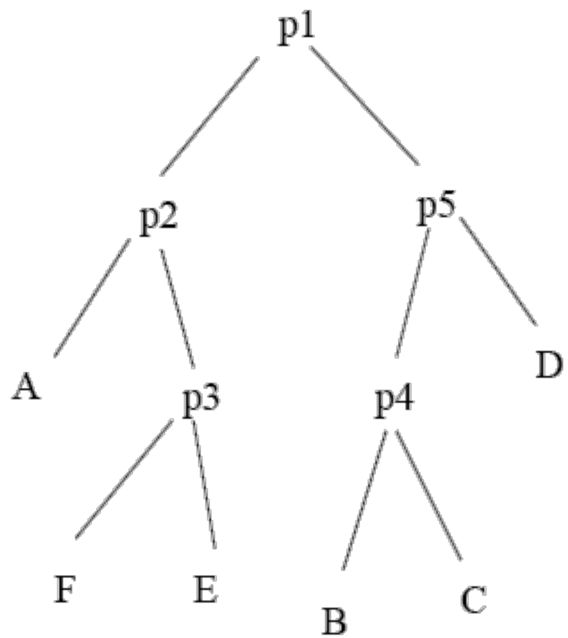
Continguts

- Control Adaptatiu de la Recursivitat
- Acceleració de la intersecció raig primari
- Acceleració intersecció raig-escena
- Jerarquia de Volums Englobants
- Subdivisió uniforme (voxelització)
- Octrees
- **Partició binaria de l'espai (BSP)**

Partició Binaria de l'Espai (BSP)

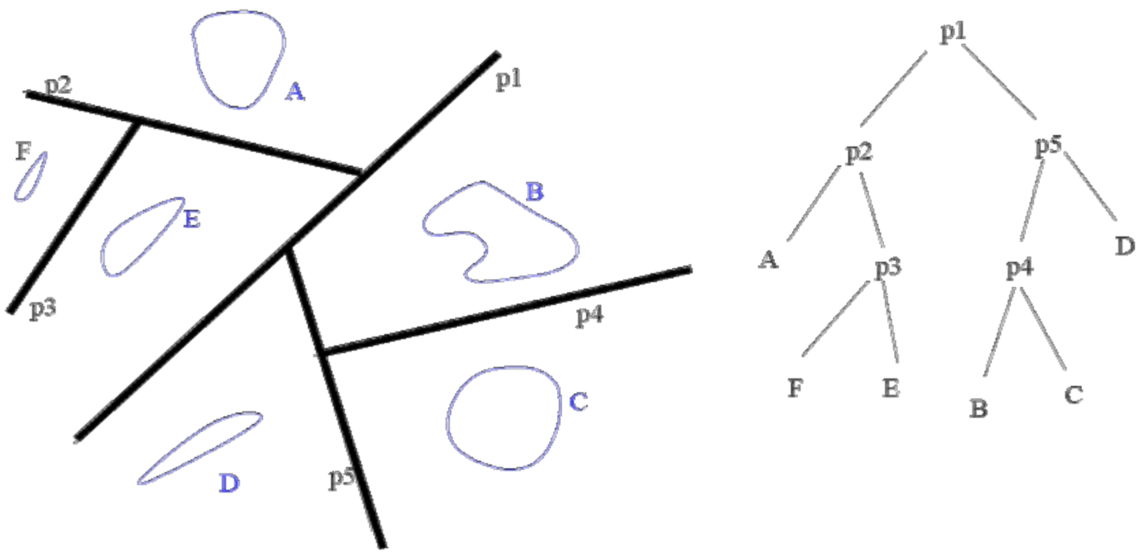
- Un **arbre BSP** és el resultat de subdividir l'espai de forma recursiva mitjançant **plans.**





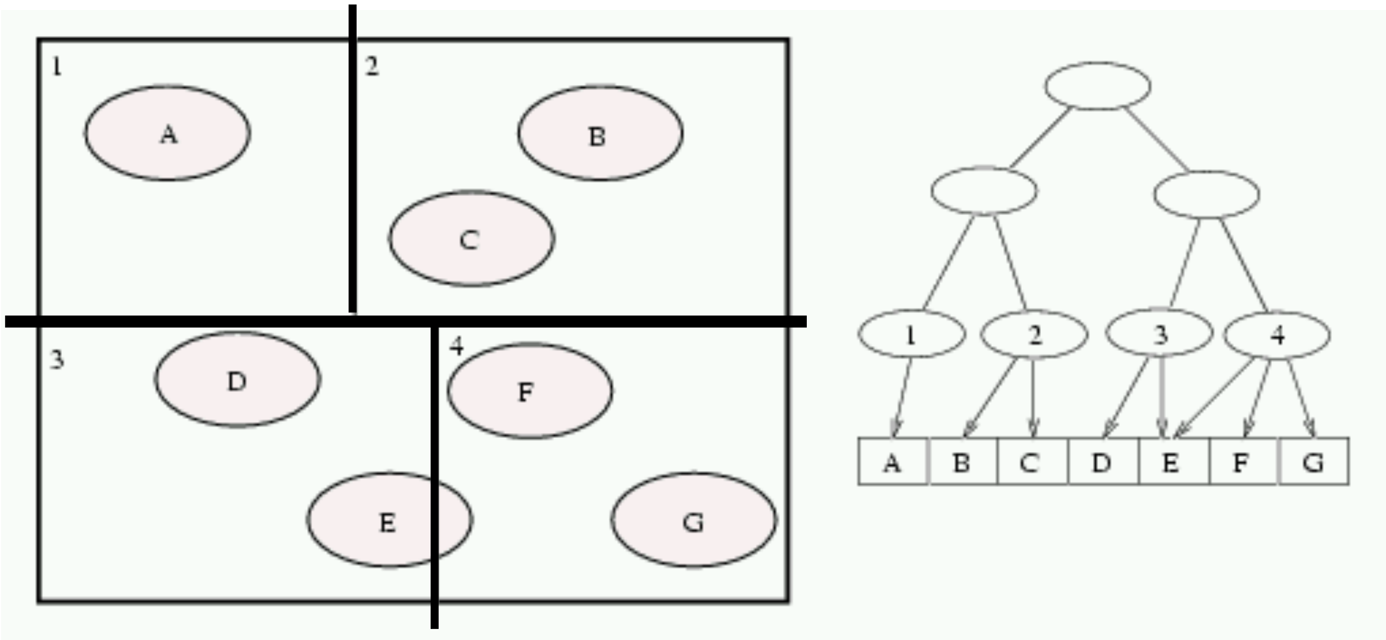
Partició Binaria de l'Espai (BSP)

- Cada node interior de l'arbre conté un **pla.**
- El **node arrel** representa tot l'espai.
- Cada fill d'un node representa la intersecció de la regió del pare amb el **semiespai positiu/negatiu** del pla.
- Un BSP subdivideix l'espai en **regions convexes.**



BSP per RayTracing

- Per eficiència s'agafen **plans aliniats** amb els eixos.
- Els nodes fulla contenen **apuntadors als objectes** de la capsa associada al node.
- Un mateix objecte pot estar referenciat en molts nodes.

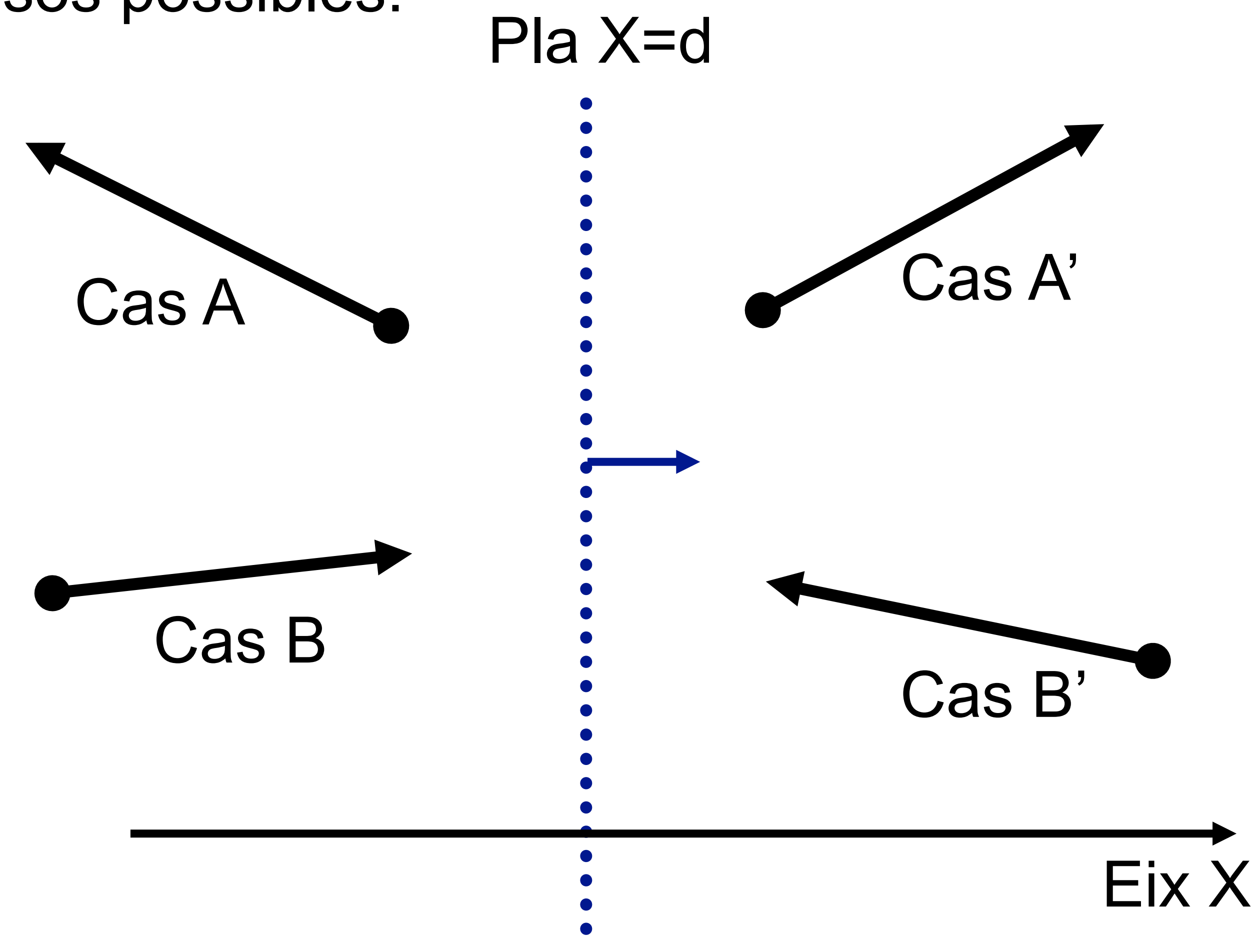


Representació BSP

```
class BSPNode : public Surface
{
    virtual bool hit(ray, λmin, λmax, hitRecord);
    ...
    Surface *left, *right;  // apuntadors als fills/objectes
    Eix eix;                // Orientació del pla (0=X, 1=Y, 2=Z)
    float d;                // coeficient d del pla
};
```

Intersecció raig-BSP

Quatre casos possibles:



Intersecció raig-BSP

Pla X=d

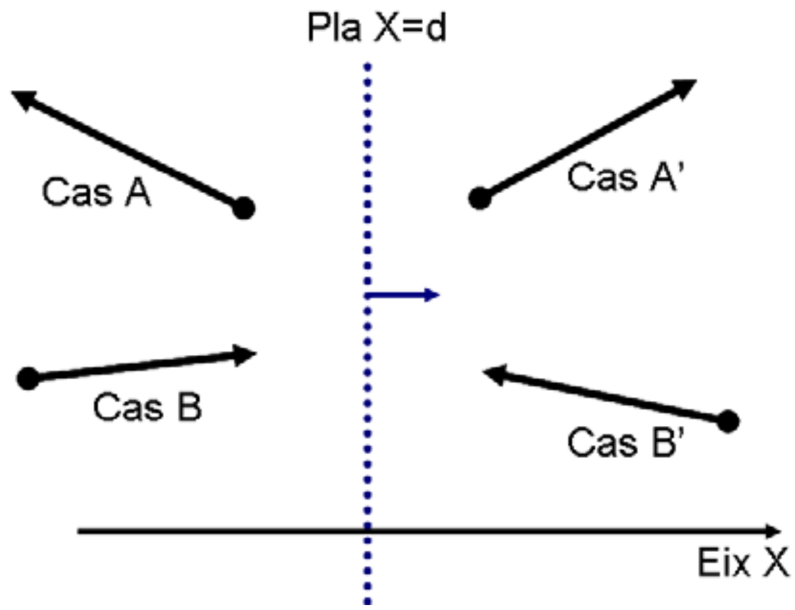
Raig P+λ**v**

Cas A: P.x<d ^ v.x < 0

Cas A' : P.x>d ^ v.x > 0

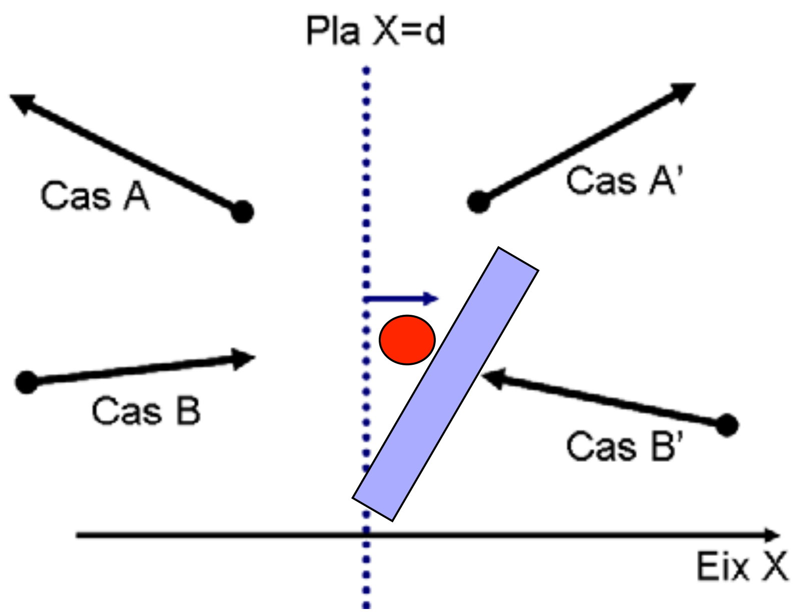
Cas B: P.x<d ^ v.x > 0

Cas B' : P.x>d ^ v.x < 0

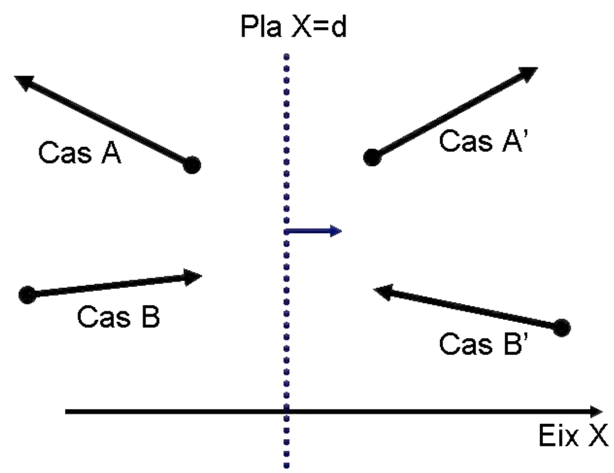


Intersecció raig-BSP

- Casos A i A’ : només cal comprovar la intersecció amb un dels dos fills (esquerra en A, dret en A’).
- Casos B i B’ : hem de comprovar la intersecció amb el primer fill, i si no hi ha intersecció, amb el segon fill.



Intersecció raig-BSP




```
funció BSPNode::hit(P, v, λmin, λmax, hit)
  si Px < d llavors  // casos A i B
    si vx < 0 llavors  // cas A
      retorna (left!=NULL ^ left→hit(raig, λmin, λmax, hit)
    fsi
    // cas B
    λ=<intersecció raig - pla>
    si λ > λmax llavors  // només cal comprobar amb un fill
      retorna (left!=NULL ^ left→hit(raig, λmin, λmax, hit)
    // cal comprobar els dos fills
    si (left!=NULL ^ left→hit(raig, λmin, λmax, hit) llavors retorna cert fsi
    retorna (right!=NULL ^ right→hit(raig, λmin, λmax, hit)
  altrament  // codi semblant pels casos A' i B'
  fsi
ffuncio
```