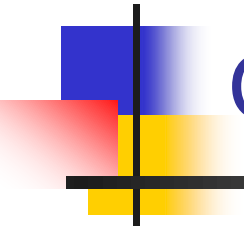


Graphics en Game Technologie

Robert G. Belleman
Universiteit van Amsterdam
Email: R.G.Belleman@uva.nl (zet a.u.b. “Graphics” in de subject)
Office: C3.139

Onderwerpen komende colleges

7. Solid modeling
8. Texture mapping
9. Data visualization
10. Interactieve graphics
11. Graphics hardware
12. Animatie en gaming

Let op:

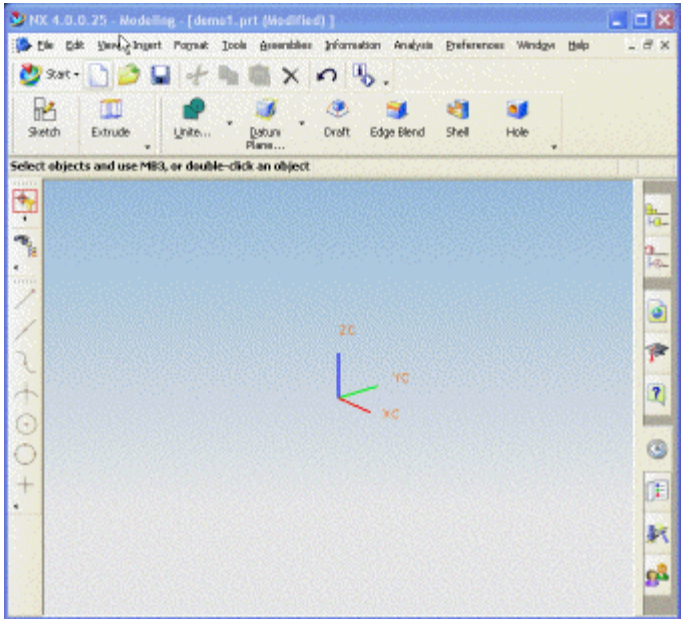
- Niet al deze onderwerpen staan in FCG van Shirley/Marschner!
- De sheets komen beschikbaar op Blackboard, vlak voor/na het college, maar:
- De sheets bieden *onvoldoende* informatie om het onderwerp te begrijpen.
- Kom naar het college! Maak aantekeningen! Stel vragen!

Graphics en Game Technologie

Solid modeling: 3D modellering van voorwerpen

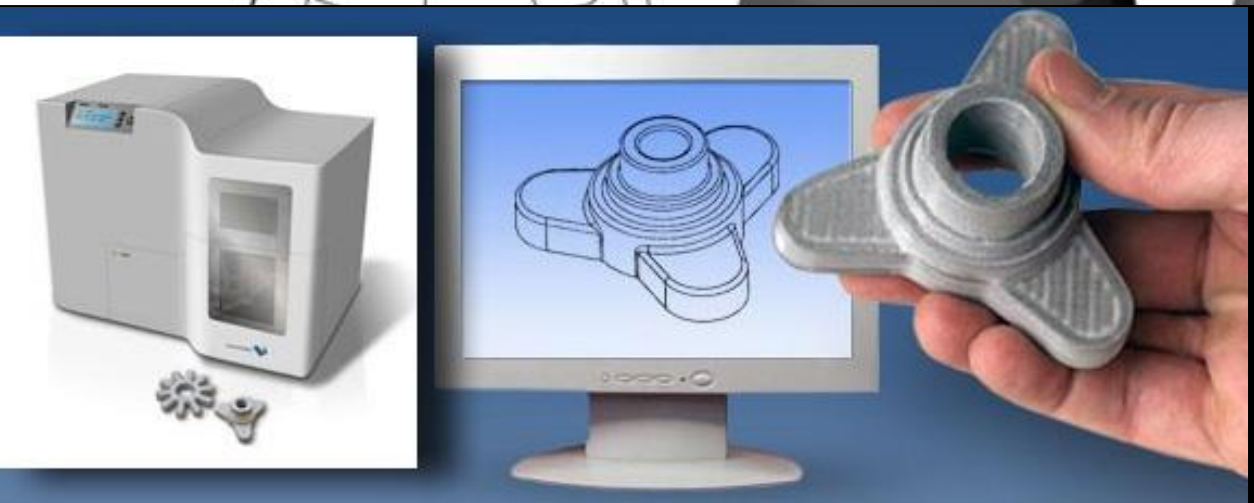
Robert G. Belleman
Universiteit van Amsterdam

FCG2: 8.1, 10.9.4, 10.10, 11.3, 13, 17.2
FCG4: 12, 13.3, 22, 17.7

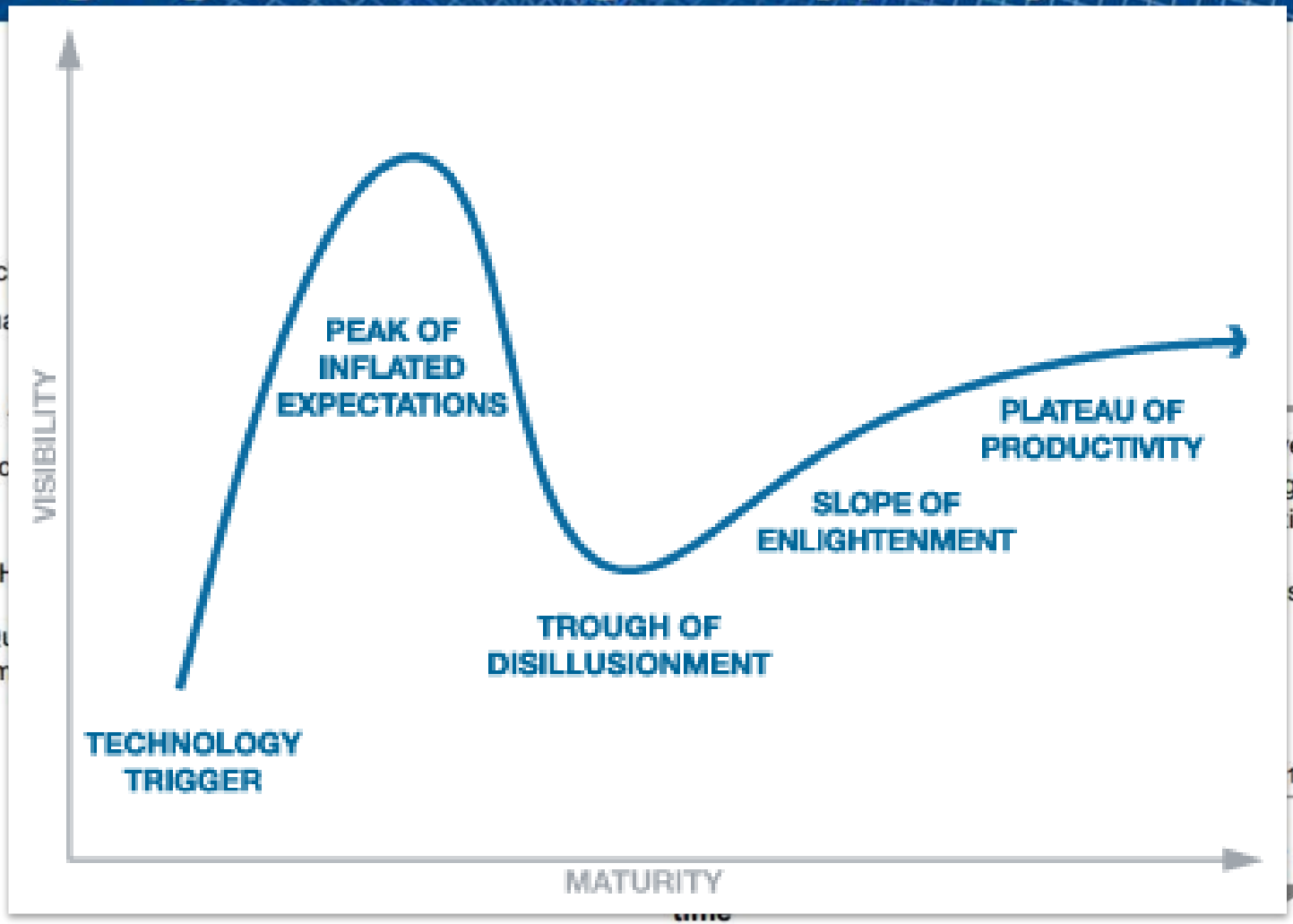


Overzicht

- Wat is een “voorwerp”?
- Object representatie methoden
- Datastructuren



Emerging Technologies Hype Cycle 2012

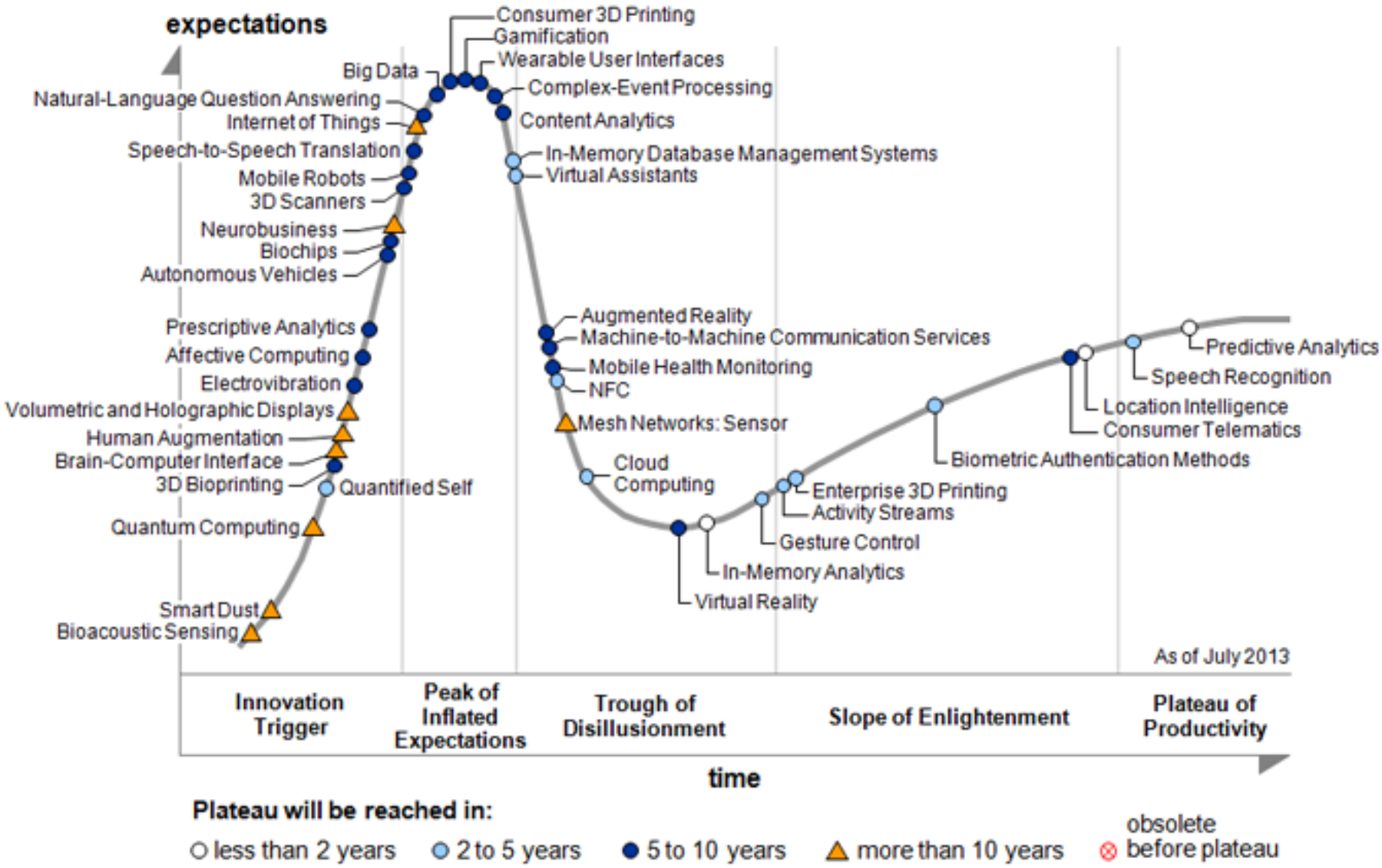


Plateau will be reached in:

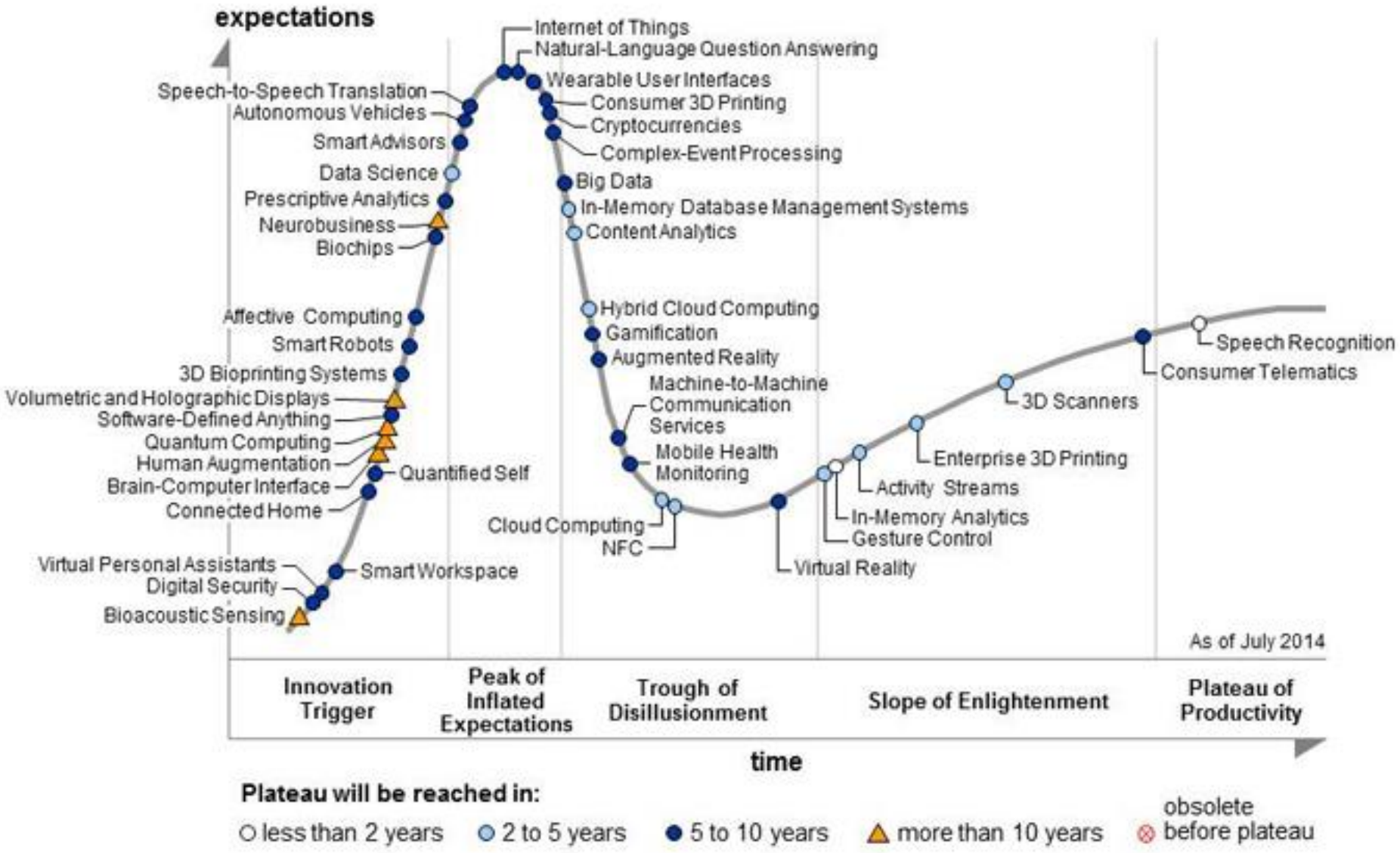
○ less than 2 years ◎ 2 to 5 years ● 5 to 10 years △ more than 10 years ⊗ obsolete before plateau

Gartner

Emerging Technologies Hype Cycle 2013 ~~2012~~

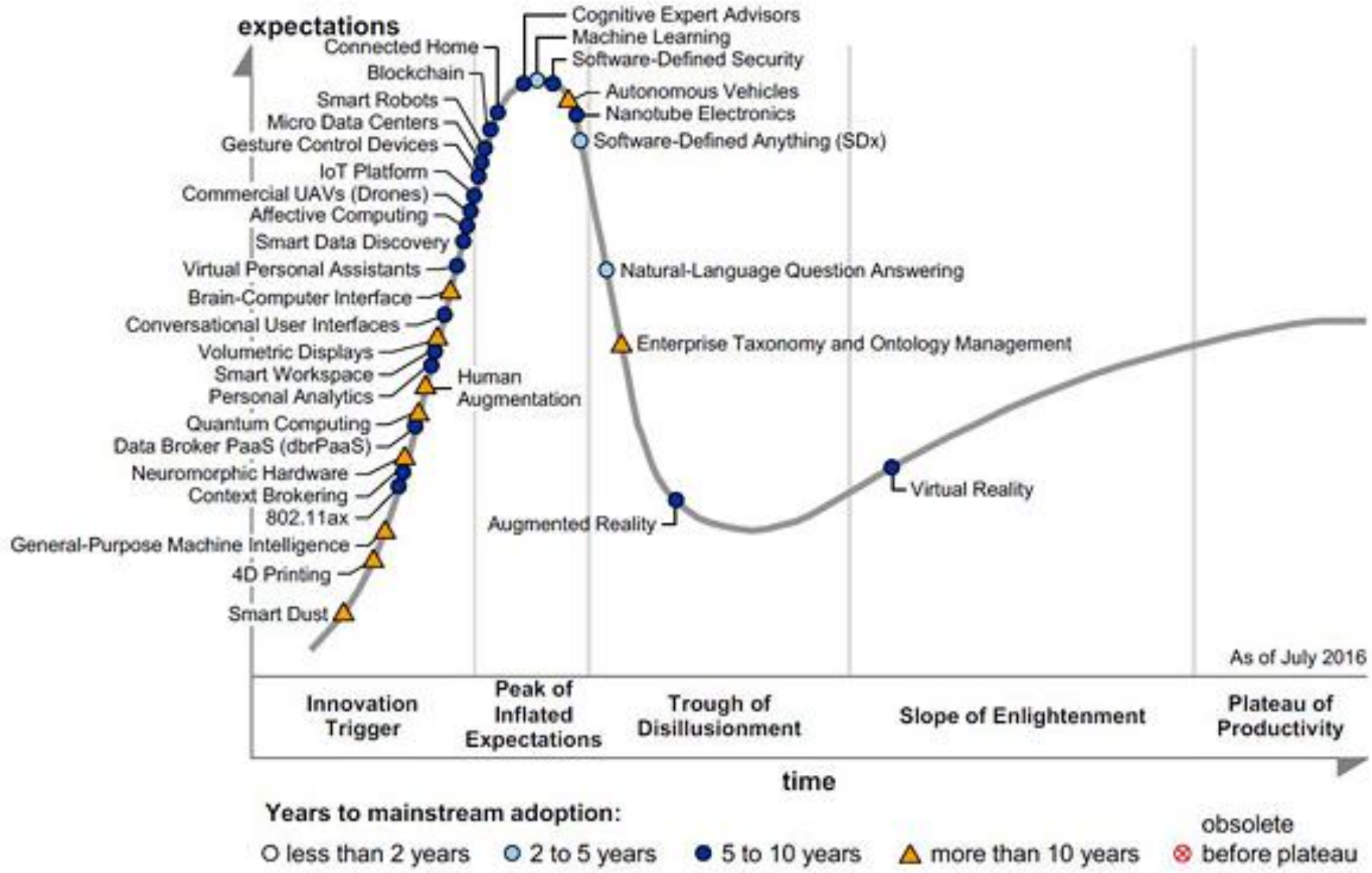


Emerging Technologies Hype Cycle ~~2012~~ 2014



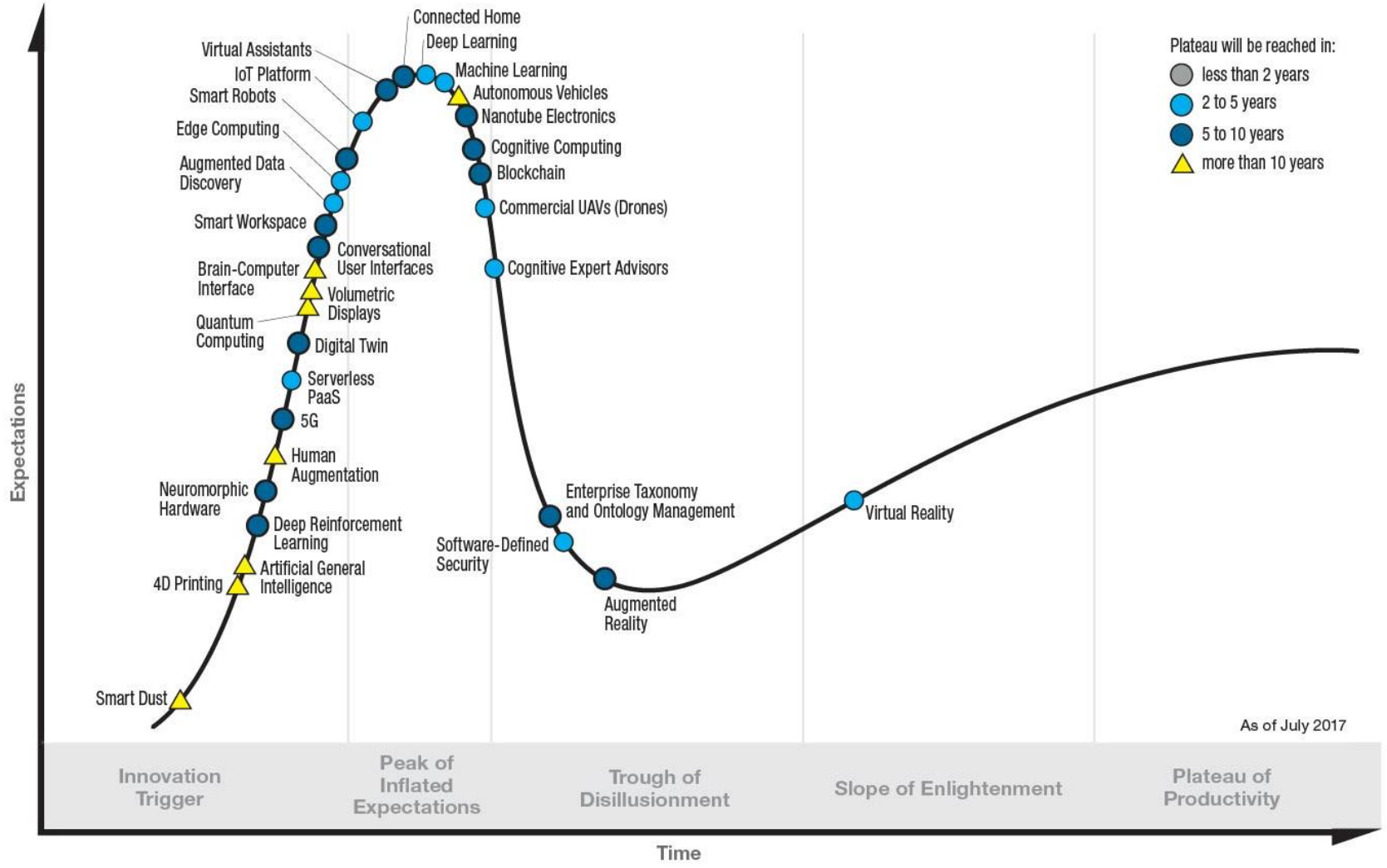
Emerging Technologies Hype Cycle 2015 ~~2012~~

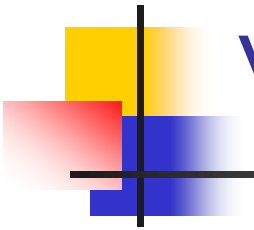
Emerging Technologies Hype Cycle ~~2012~~ 2016



Source: Gartner (July 2016)

Emerging Technologies Hype Cycle 2017 ~~2012~~





Voorwerpen

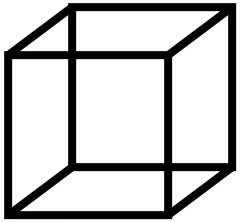
Tot nu toe: voorwerpen gerepresenteerd door lijnen, krommen, polygonen en oppervlakken in 2D en 3D. Maar deze hoeven niet een volumetrisch voorwerp te omsluiten, met een binnenkant (interieur) en buitenkant (exterieur)

- Hoe representeer je een voorwerp (volume, object, “solids”)?
- Hoe representeer je omgevingen (“scenes”) bestaande uit voorwerpen?

Toepassingen:

- Games, virtuele simulatie omgevingen
- CAD/CAM (prototyping, prosthetics)
- Fysische simulatie systemen
  - Eindige elementen methode (FEM/FVM)

Representeren van voorwerpen

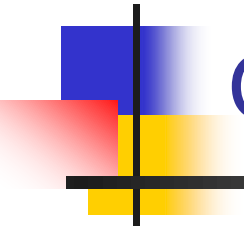


Gewenste eigenschappen:

1.	*Volledig*	Elk denkbaar voorwerp kan worden gerepresenteerd;
2.	*Ondubbelzinnig*	Een object kan niet anders worden geïnterpreteerd dan is bedoeld;
3.	*Uniek*	Een object kan maar op één manier gemaakt worden (“complete”);
4.	*Correct*	Het is niet mogelijk een incorrect voorwerp te maken, het moet eenvoudig zijn een correct voorwerp te maken;
5.	*Nauwkeurig*	Het voorwerp is geen benadering maar exact;
6.	*Gesloten*	Een voorwerp is na transformatie (translatie, rotatie, schaling, boolean operatoren, etc.) nog steeds een correct voorwerp;
7.	*Compact*	Het voorwerp kan compact gerepresenteerd worden;
8.	*Efficient*	Het voorwerp kan makkelijk/snel worden gemanipuleerd (aanpassen, verwijderen) en geïnspecteerd (interactie).

Het maken van een voorwerp met al deze eigenschappen is lastig.

Vaak is een compromis nodig.

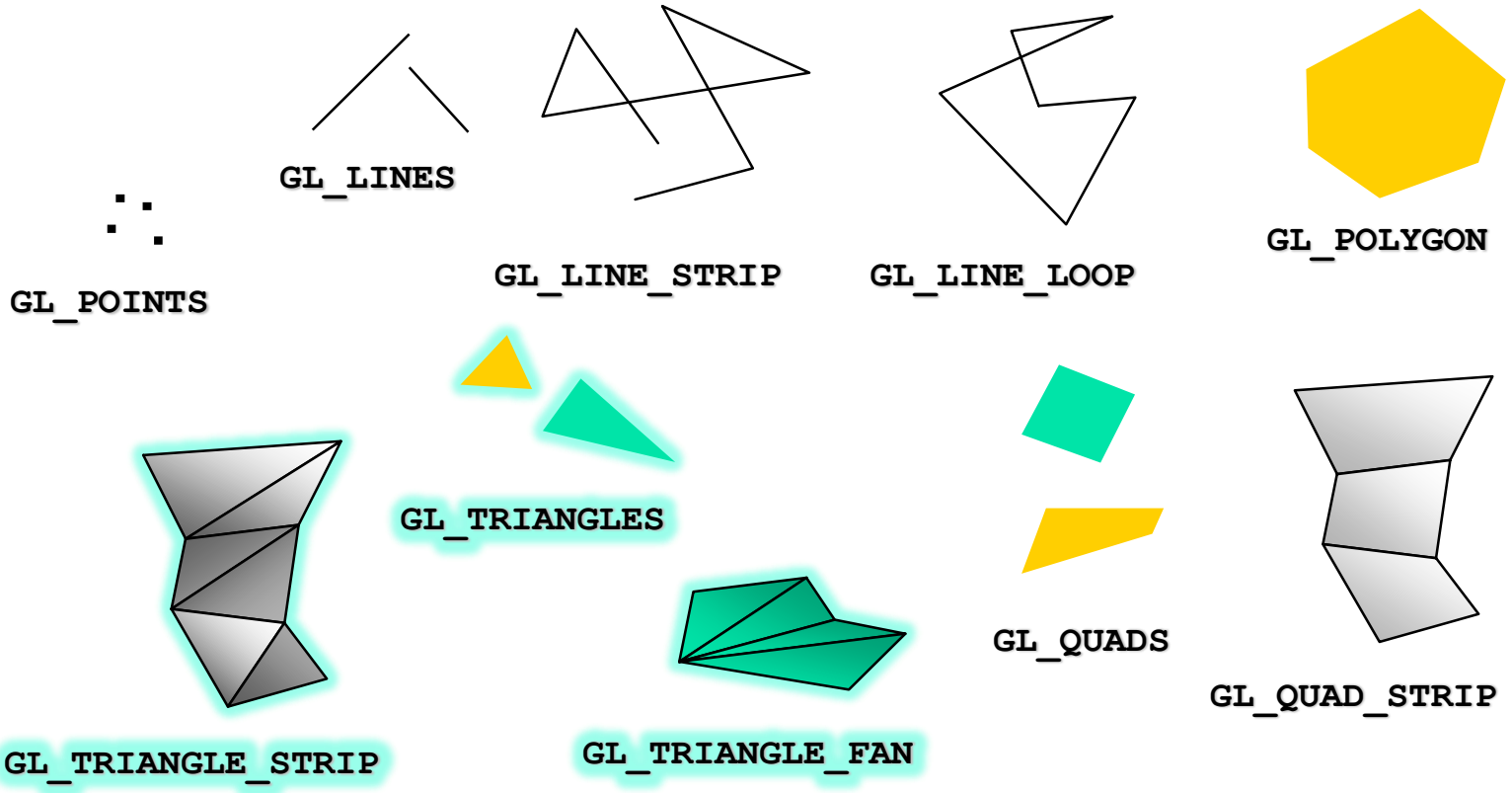


Object representatie methoden

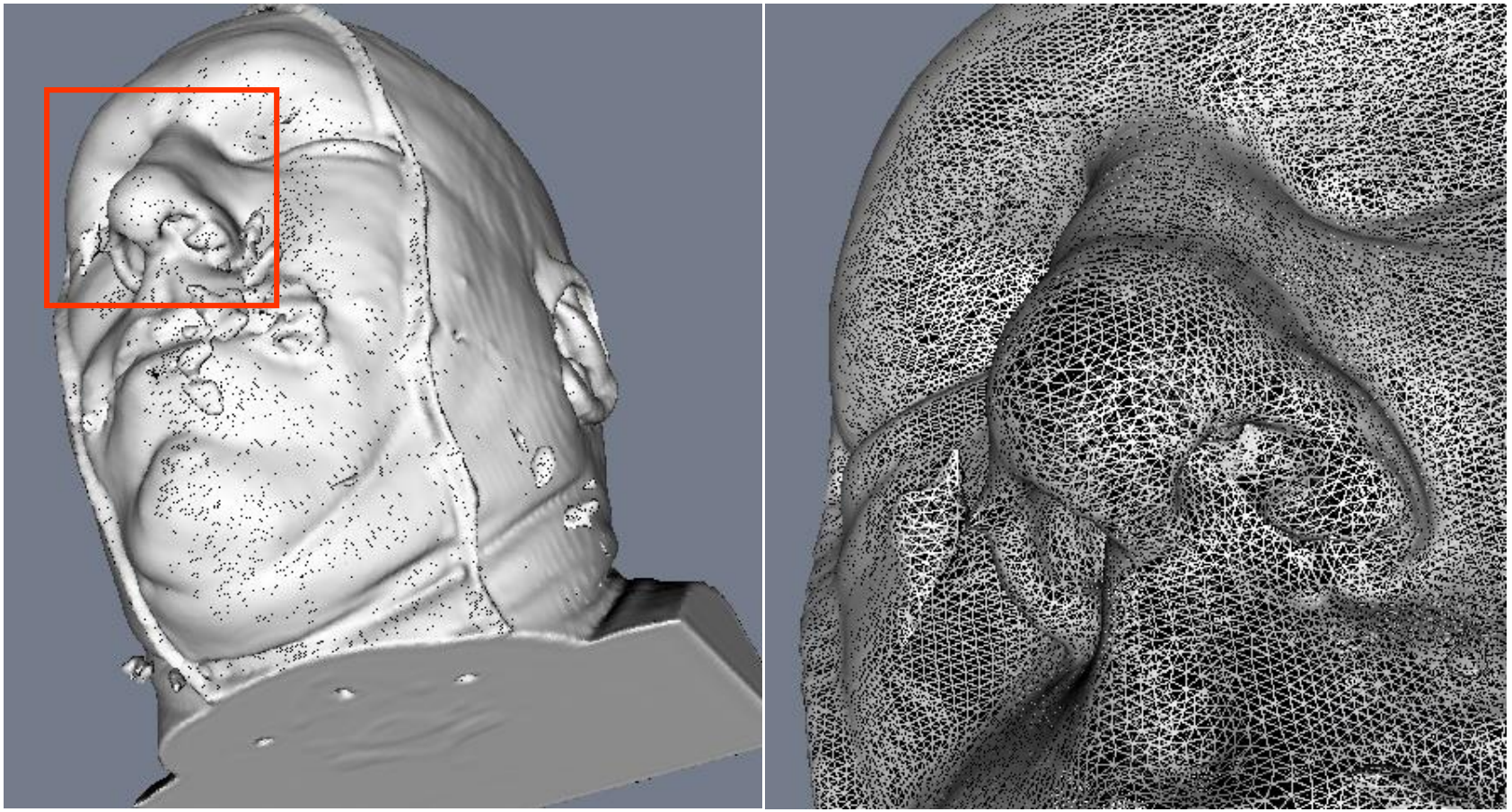
1. Triangle mesh
2. Extrusie methoden
3. Quadratische oppervlakken
4. Constructieve methoden
5. Omhullingen

1. Triangle mesh

- Verzameling verbonden driehoeken
  - Algemener: verzameling van polygonen
- Drie punten (vertices) per driehoek



1. Triangle mesh

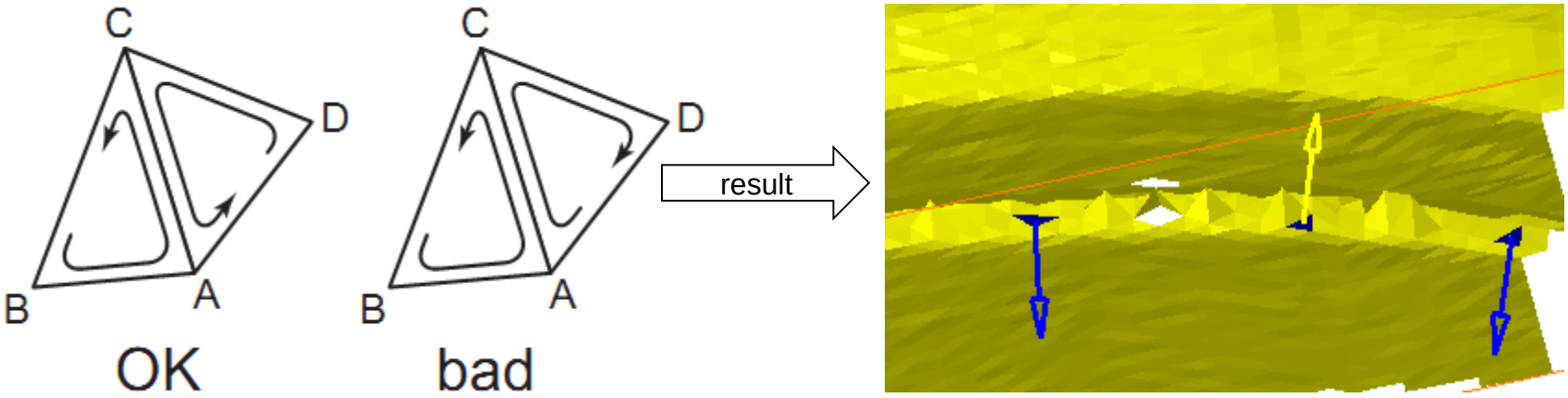


Visible male dataset, NIH

Triangle orientation

- Onderscheid “binnen” en “buiten” wordt gemaakt op basis van volgorde van punten:
  - Clockwise of counterclockwise

- Belangrijk dat dit consistent gebeurt:

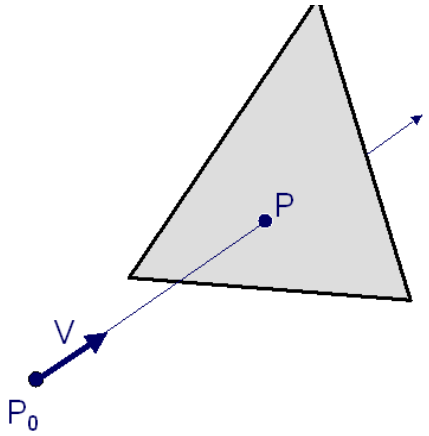


Triangle mesh

- Voordelen
  - Makkelijke manier om objecten mee te beschrijven
  - Driehoeken zijn efficient te renderen
- Nadelen
  - Mogelijkheid om binnen/buitenkant van objecten aan te geven is ambigu
  - Moeilijk om objecten te combineren ($\cap$, $\cup$, -)
  - Voor complexe modellen zijn veel driehoeken nodig

    → renderen kan lang duren

    → interactie met een model is niet altijd efficient



2. Extrusie: “sweeps”

Methode:

1. Neem een 2D voorwerp
2. Definieer een pad
3. Beweeg het 2D voorwerp loodrecht over het pad
4. Het resulterend volume definieert het voorwerp

Soorten:

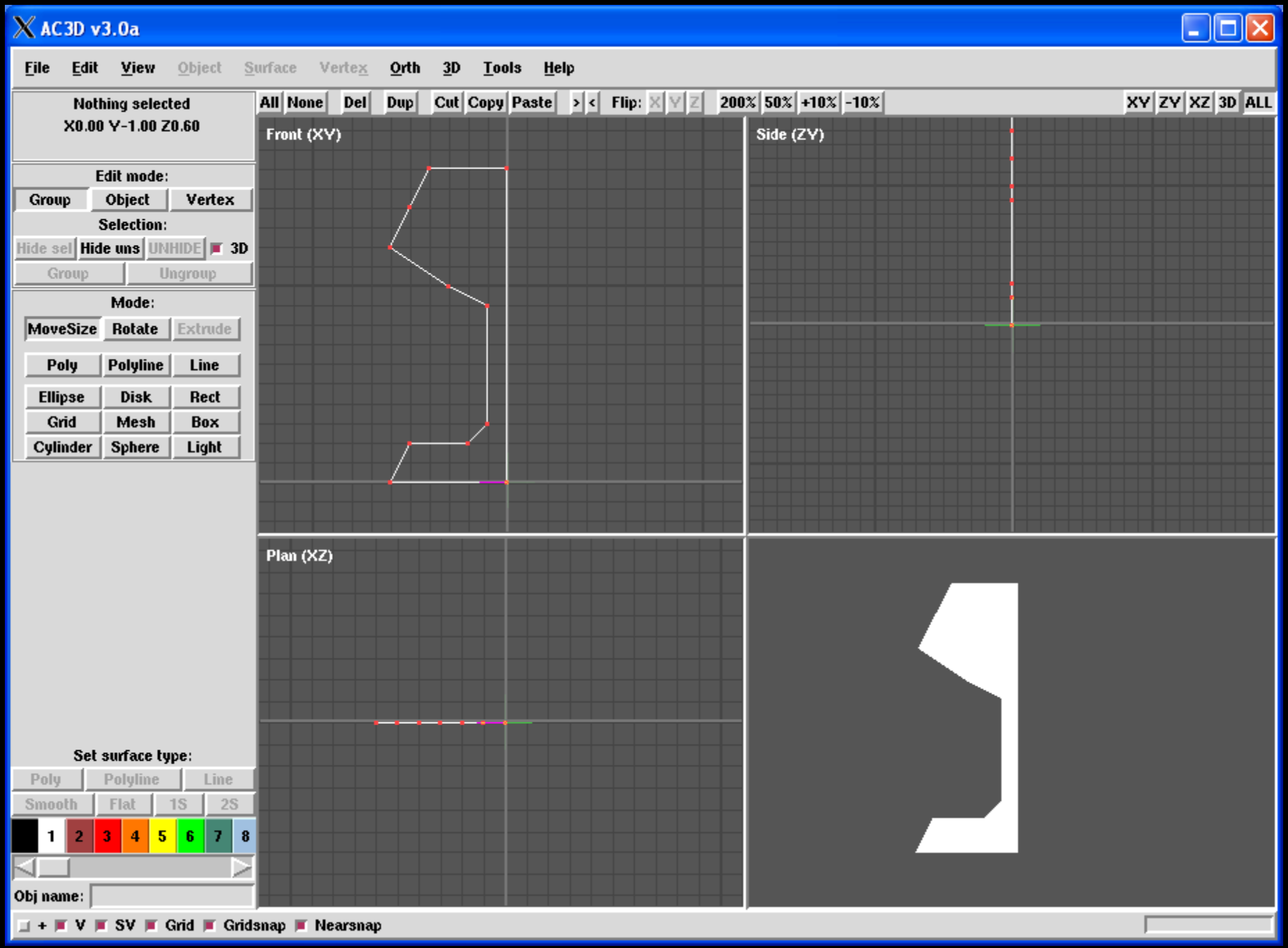
- Translational sweeps (extrusion)
- Rotational sweeps (revolvements)

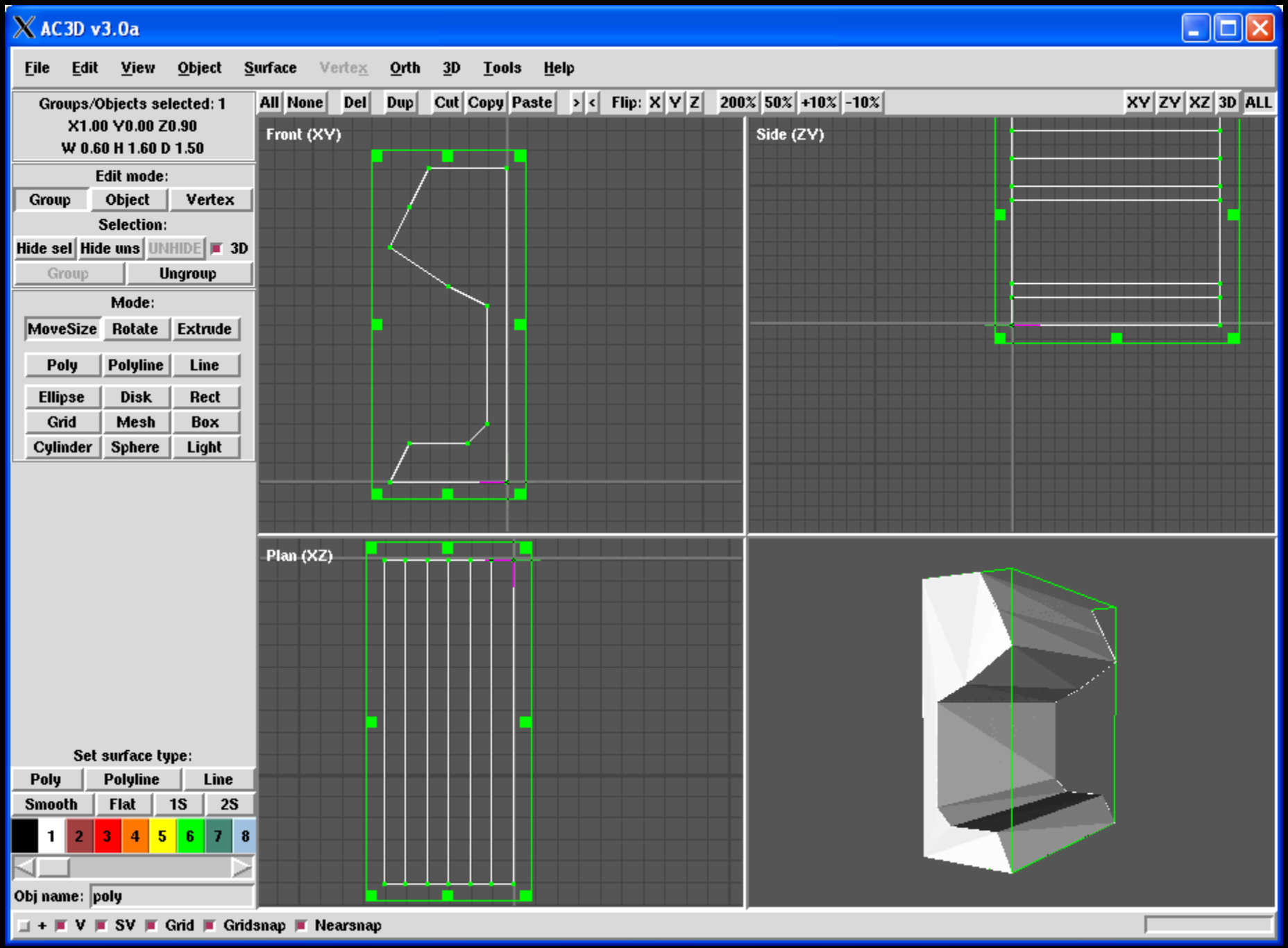
Voordelen:

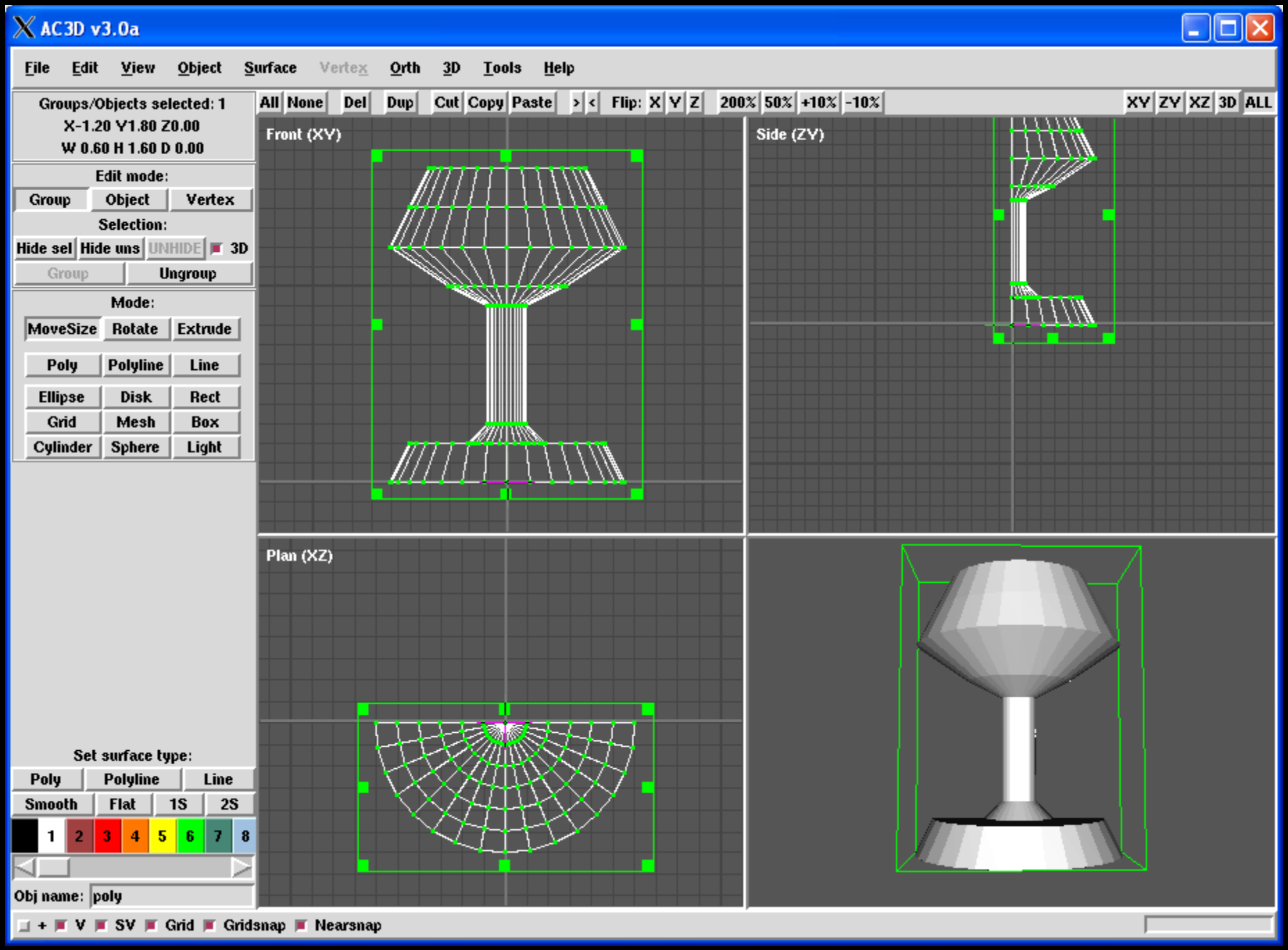
- Conceptueel eenvoudig
- Compact

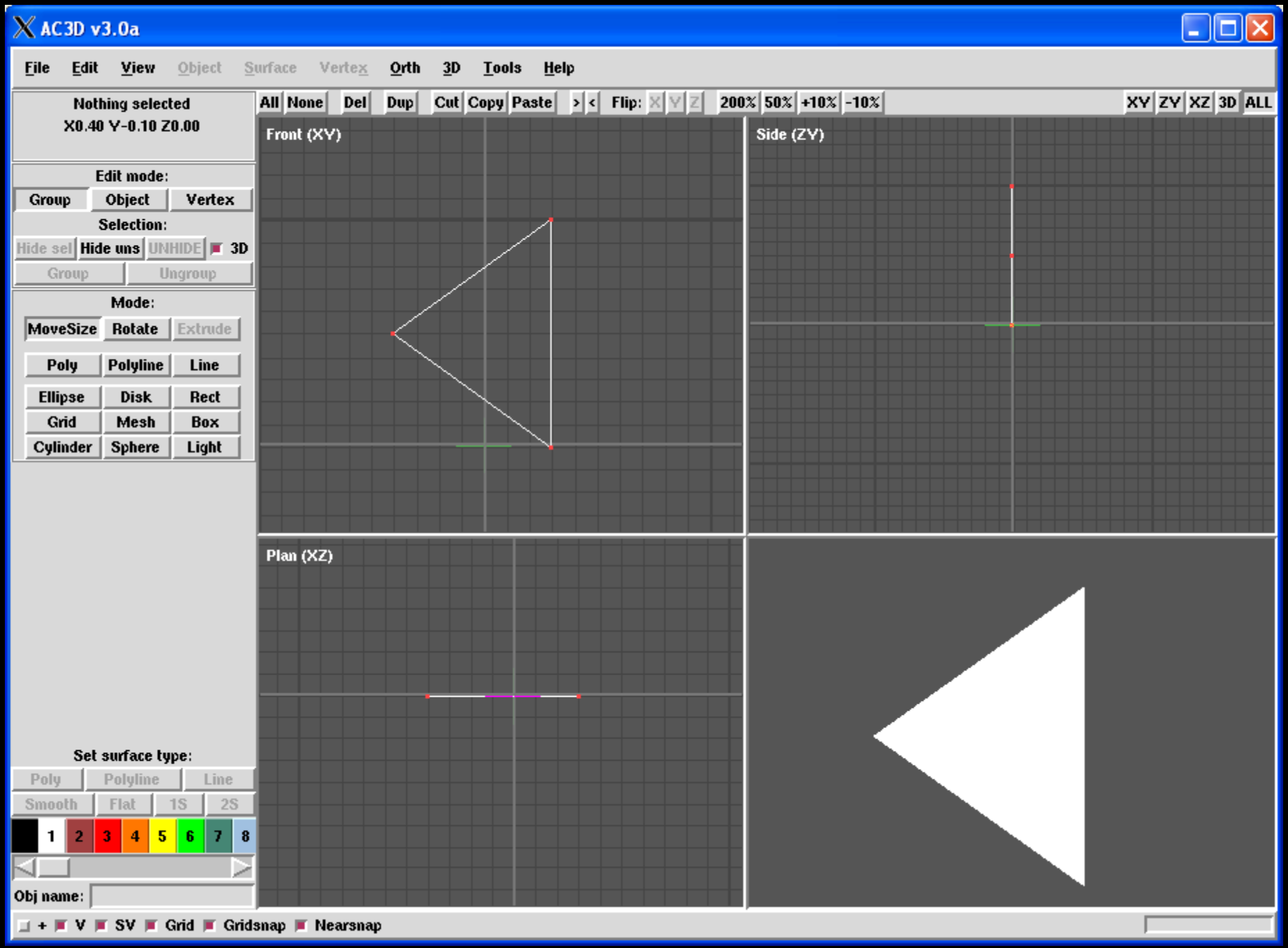
Nadelen:

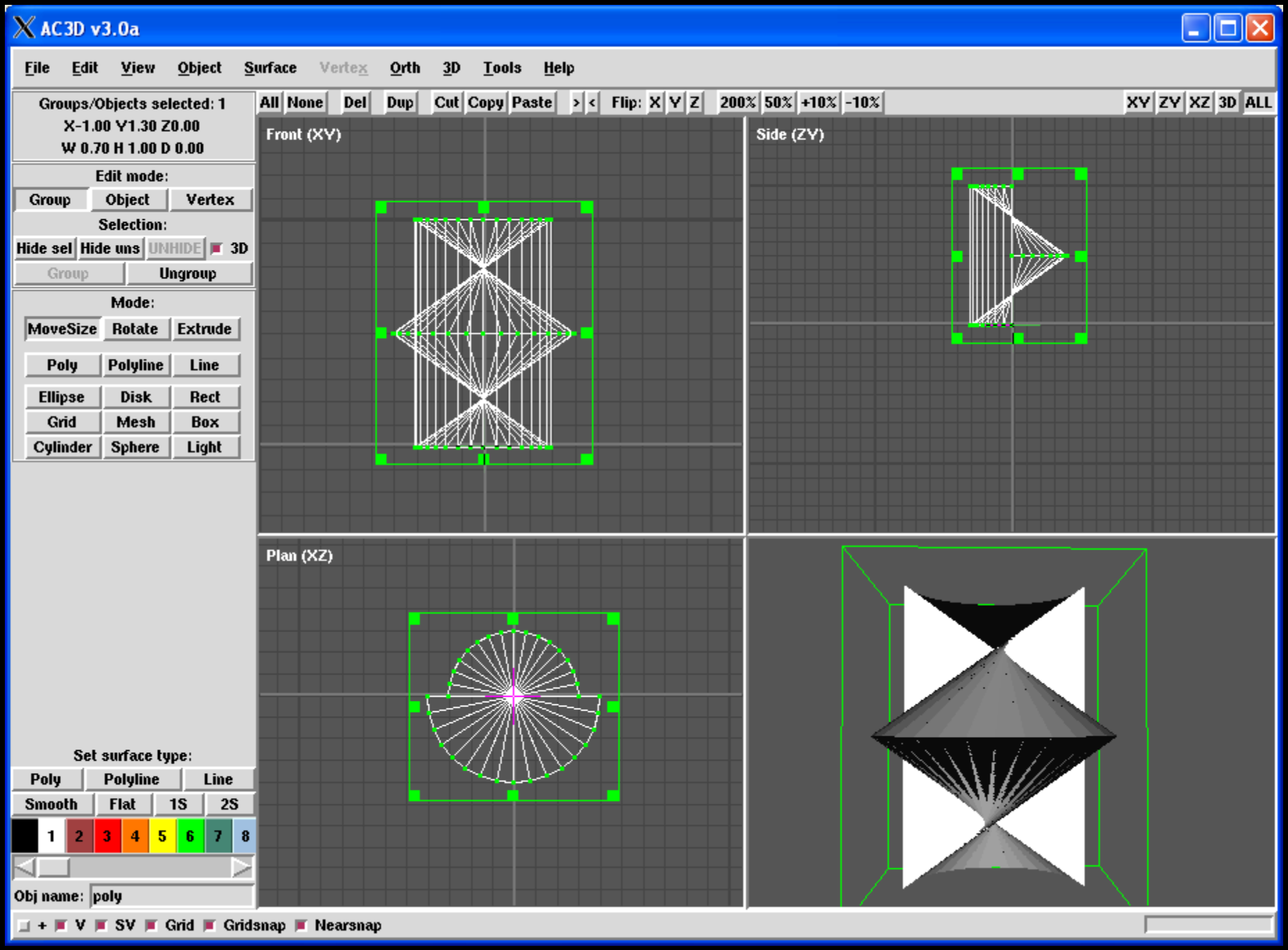
- Moeilijk om efficiënt te implementeren
- Kan resulteren in incorrecte objecten
- Sweeps zijn niet gesloten onder de “regulerende boolse operatoren”
- Nauwkeurigheid afhankelijk van resolutie 2D polygoon en pad


AC3D v3.0a
File Edit View Object Surface Vertex Orth 3D Tools Help
Nothing selected
X0.00 Y-1.00 Z0.60
Edit mode:
Group Object Vertex
Selection:
Hide sel Hide uns UNHIDE 3D
Group Ungroup
Mode:
MoveSize Rotate Extrude
Poly Polyline Line
Ellipse Disk Rect
Grid Mesh Box
Cylinder Sphere Light
Set surface type:
Poly Polyline Line
Smooth Flat 1S 2S
1 2 3 4 5 6 7 8
Obj name:
All None Del Dup Cut Copy Paste > < Flip: X Y Z 200% 50% +10% -10%
XV ZY XZ 3D ALL
Front (XY)
Side (ZY)
Plan (XZ)
+ V SV Grid Gridsnap Nearsnap


AC3D v3.0a
File Edit View Object Surface Vertex Orth 3D Tools Help
Groups/Objects selected: 1
X1.00 Y0.00 Z0.90
W 0.60 H 1.60 D 1.50
Edit mode:
Group Object Vertex
Selection:
Hide sel Hide uns UNHIDE 3D
Group Ungroup
Mode:
MoveSize Rotate Extrude
Poly Polyline Line
Ellipse Disk Rect
Grid Mesh Box
Cylinder Sphere Light
Set surface type:
Poly Polyline Line
Smooth Flat 1S 2S
1 2 3 4 5 6 7 8
Obj name: poly
All None Del Dup Cut Copy Paste > < Flip: X Y Z 200% 50% +10% -10%
XY ZY XZ 3D ALL
Front (XY)
Side (ZY)
Plan (XZ)
+ V SV Grid Gridsnap Nearsnap


AC3D v3.0a
File Edit View Object Surface Vertex Orth 3D Tools Help
Groups/Objects selected: 1
X-1.20 Y1.80 Z0.00
W 0.60 H 1.60 D 0.00
Edit mode:
Group Object Vertex
Selection:
Hide sel Hide uns UNHIDE 3D
Group Ungroup
Mode:
MoveSize Rotate Extrude
Poly Polyline Line
Ellipse Disk Rect
Grid Mesh Box
Cylinder Sphere Light
Set surface type:
Poly Polyline Line
Smooth Flat 1S 2S
1 2 3 4 5 6 7 8
Obj name: poly
All None Del Dup Cut Copy Paste > < Flip: X Y Z 200% 50% +10% -10%
XY ZY XZ 3D ALL
Front (XY)
Side (ZY)
Plan (XZ)
+ V SV Grid Gridsnap Nearsnap


AC3D v3.0a
File Edit View Object Surface Vertex Orth 3D Tools Help
Nothing selected
X0.40 Y-0.10 Z0.00
Edit mode:
Group Object Vertex
Selection:
Hide sel Hide uns UNHIDE 3D
Group Ungroup
Mode:
MoveSize Rotate Extrude
Poly Polyline Line
Ellipse Disk Rect
Grid Mesh Box
Cylinder Sphere Light
Set surface type:
Poly Polyline Line
Smooth Flat 1S 2S
1 2 3 4 5 6 7 8
Obj name:
All None Del Dup Cut Copy Paste > < Flip: X Y Z 200% 50% +10% -10%
XY ZY XZ 3D ALL
Front (XY)
Side (ZY)
Plan (XZ)
+ V SV Grid Gridsnap Nearsnap

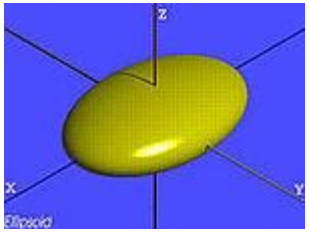

AC3D v3.0a
File Edit View Object Surface Vertex Orth 3D Tools Help
Groups/Objects selected: 1
X-1.00 Y1.30 Z0.00
W 0.70 H 1.00 D 0.00
Edit mode:
Group Object Vertex
Selection:
Hide sel Hide uns UNHIDE 3D
Group Ungroup
Mode:
MoveSize Rotate Extrude
Poly Polyline Line
Ellipse Disk Rect
Grid Mesh Box
Cylinder Sphere Light
Set surface type:
Poly Polyline Line
Smooth Flat 1S 2S
1 2 3 4 5 6 7 8
Obj name: poly
All None Del Dup Cut Copy Paste > < Flip: X Y Z 200% 50% +10% -10%
XY ZY XZ 3D ALL
Front (XY)
Side (ZY)
Plan (XZ)
+ V SV Grid Gridsnap Nearsnap

3. Quadric surfaces

- Voorwerp voorgesteld door middel van een kwadratische vergelijking
  - cirkel geroteerd om zijn middelpunt: bol

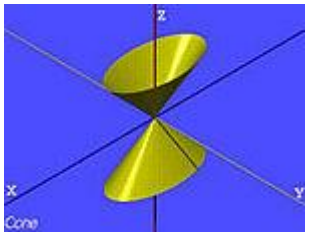
$$\frac{x^2}{a^2} + \frac{y^2}{b^2} + \frac{z^2}{c^2} = 1$$

(bol als a = b = c)



  - lijn met een punt op de rotatie-as: kegel

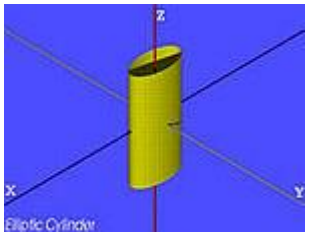
$$\frac{x^2}{a^2} + \frac{y^2}{b^2} - \frac{z^2}{c^2} = 0$$

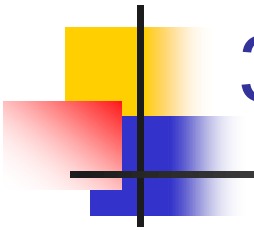


  - Lijn parallel aan de rotatie-as: cilinder

$$\frac{x^2}{a^2} + \frac{y^2}{b^2} = 1$$

(circulair als a = b)





3. Quadric surfaces

Voordelen:

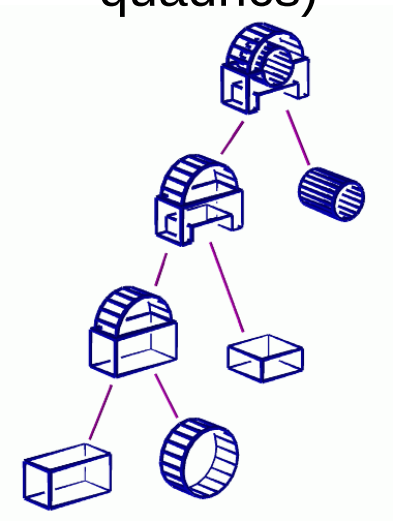
- Zeer nauwkeurig
- Zeer compact

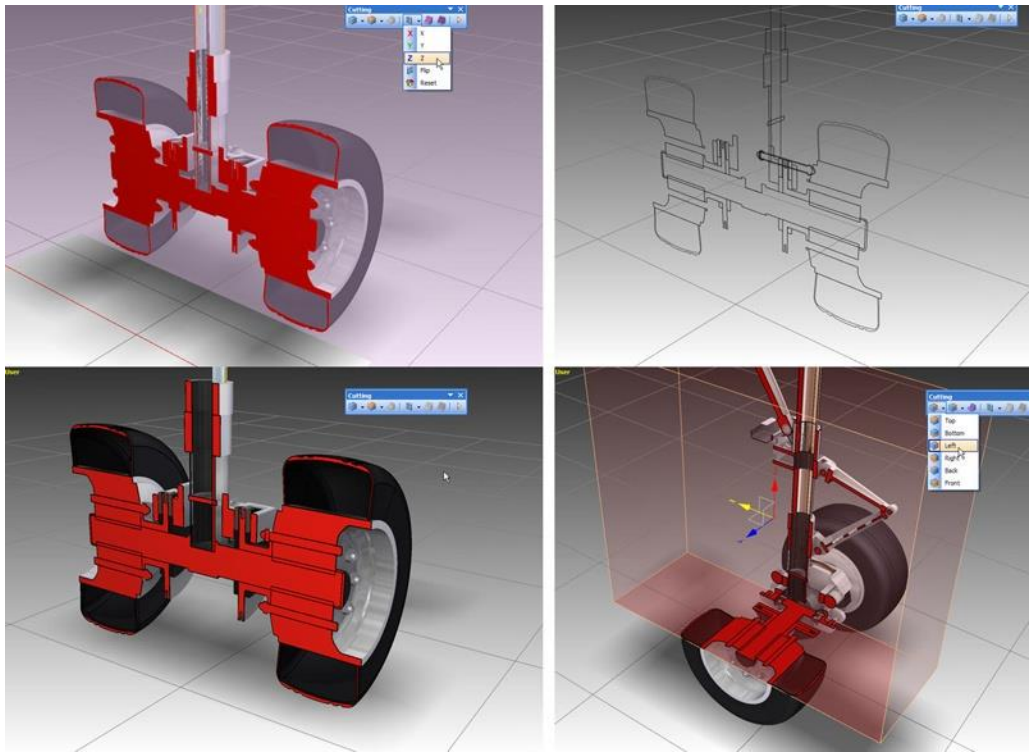
Nadelen:

- Gelimiteerd toepassingsgebied?
- Lastig om onregelmatige objecten mee te modeleren

4. Constructieve methoden

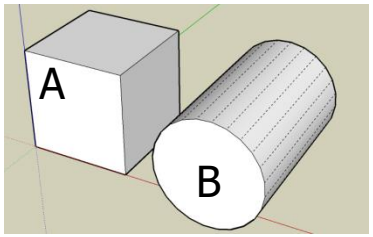
- De meest populaire methode om complexe objecten te modeleren is door het combineren van primitieven
  - Blokken, bollen, cylinders, kegels, wiggen, etc. (vaak quadrics)

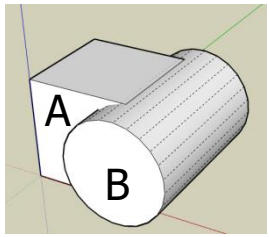




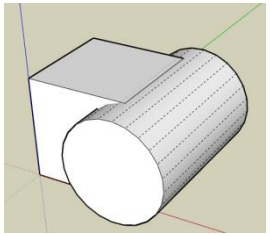
Boolse verzameling operaties

- Eenvoudige voorwerp combineren tot complex voorwerp :

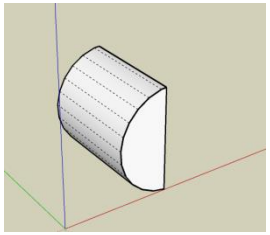




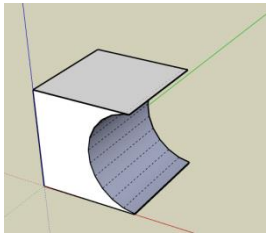
- Vereniging, doorsnede en verschil:



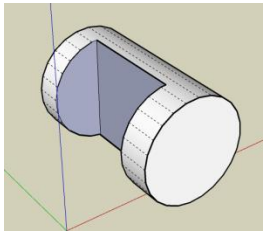
A ∪ B



A ∩ B



A - B

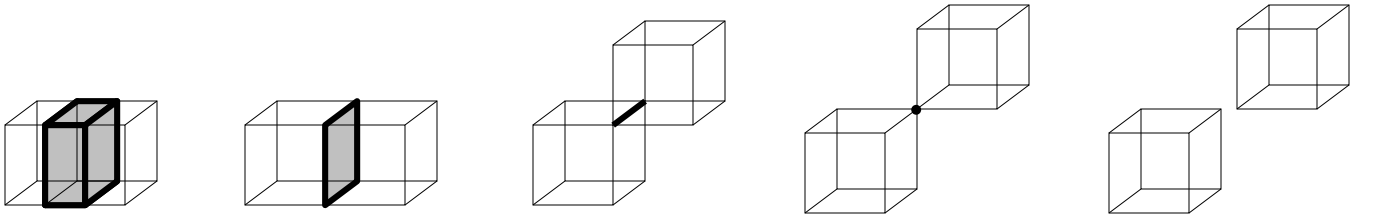


B - A

- Metafoor: “aan elkaar plakken”, “uitsnijden”

Boolse verzameling operaties

- Let op: dit levert niet altijd weer een voorwerp op!
  Neem de boolse doorsnede van twee kubussen:

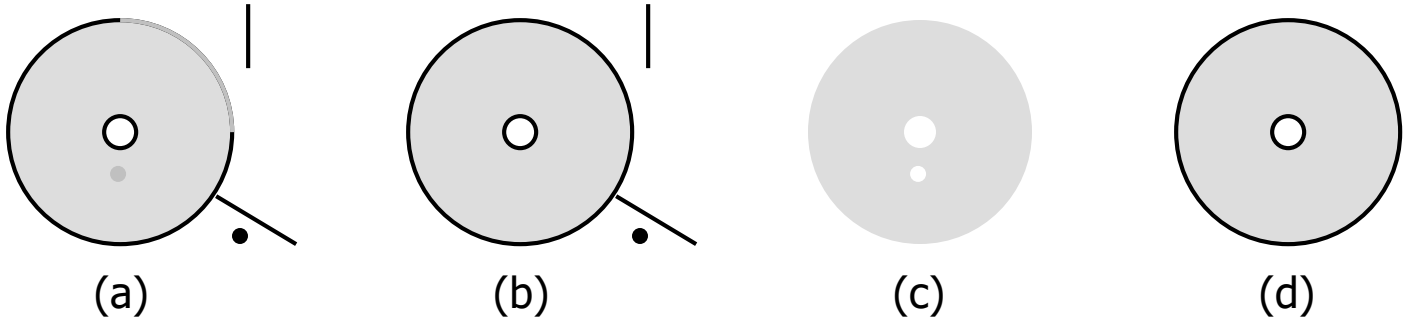


voorwerp vlak lijn punt leeg voorwerp

- Regulerende boolse verzameling operatoren:
  $\cap^*$, $\cup^*$ en $-^*$

- Zo is $A \cap^* B$ of een voorwerp, of leeg

“Reguleren” van een voorwerp



a) Neem een ongereguleerd voorwerp gedefinieerd door:
   - Interne punten (lichtgrijs)
   - Randpunten die deel uitmaken van het object (zwart)
   - Randpunten die niet deel uitmaken van het object (donkergrijs)
   - Bengelende en onverbonden lijn en punt en intern randpunt

b) Hiervan is het “gesloten” voorwerp (“closure”):
   - Alle randpunten van het voorwerp
   - Interne punten van het voorwerp

c) De “binnenkant” van het voorwerp (“interior”):
   - Alle bengelende en losse lijnen/punten verwijderd

d) Gereguleerde voorwerp = gesloten binnenkant van voorwerp

Reguleren van een voorwerp

Een gereguleerd voorwerp kan:

- geen randpunten bevatten die naast een intern punt liggen
- geen ‘bengelende’ punten bevatten

Regulerende boolse verzameling operatoren als gewone boolse operatoren:

A op B = closure(interior(A op B))*

Een regulerende boolse verzameling operator neemt twee gereguleerde voorwerpen en produceert een gereguleerd voorwerp.

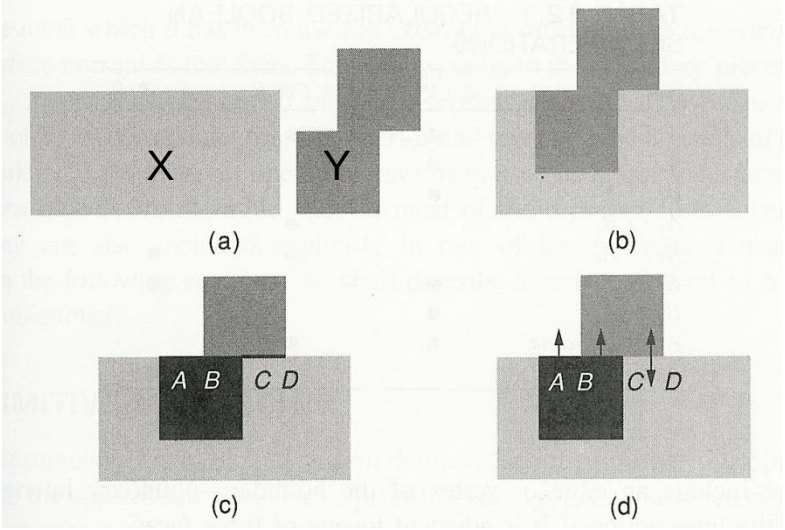
Verschil gewone en regulerende boolse operaties

a) 2 voorwerpen X en Y

b) Positie voor $\cap$

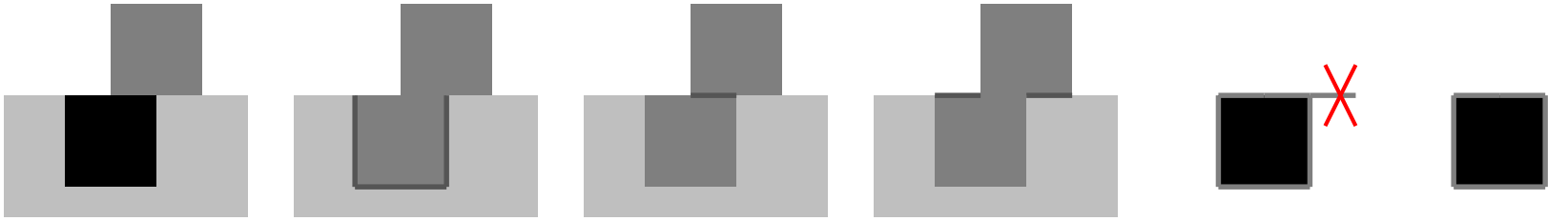
c) X$\cap$Y bevat AB, BC en CD
   - (X_i en Y_b) $\cap$ (Y_i en X_b)

d) X$\cap$*Y bevat AB en BC maar niet CD
   - ($X_i \cap Y_i$, $Y_b \cap X_i$, $X_b \cap Y_i$ en een stukje van $X_b \cap Y_b$



X_i : interieur van X

X_b : boundary van X



$X_i \cap Y_i$ $Y_b \cap X_i$ $X_b \cap Y_i$ $X_b \cap Y_b$ $X \cap Y$ $X \cap^* Y$

Verschil gewone en regulerende boolse operaties

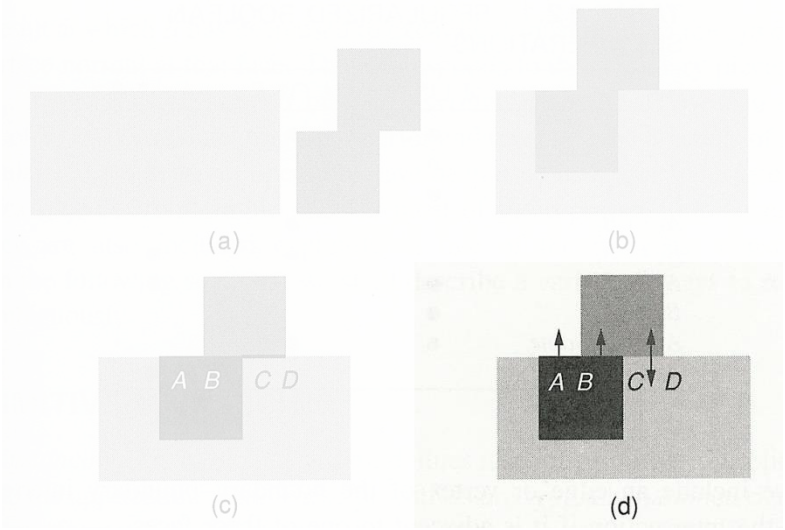
De rand zit wel in $\cap^*$ als:

- Binnenkant van beide voorwerpen zit aan dezelfde kant
  - Zie de pijlen in figuur (d): lijnstuk AB
- Interne punten in doorsnede
  - lijnstuk BC

De rand zit niet in $\cap^*$ als:

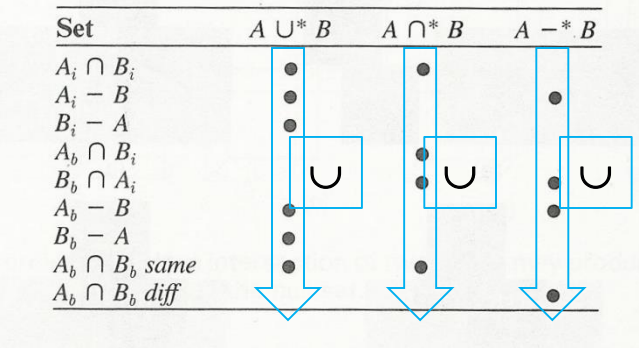
- Binnenkant van beide voorwerpen zit aan tegenovergestelde kant
  - lijnstuk CD
- Interne punten niet in doorsnede

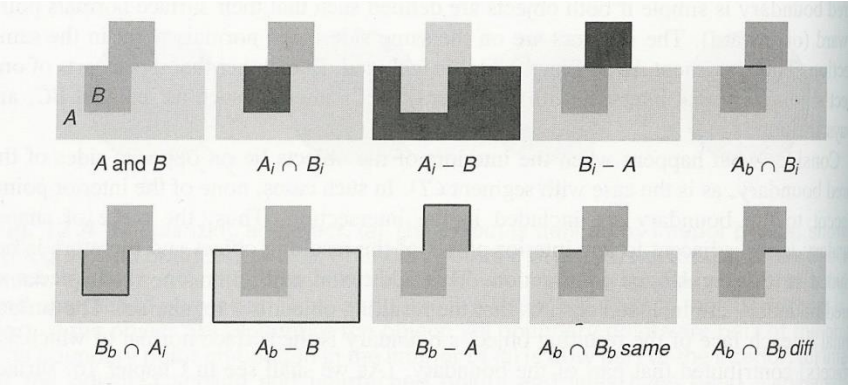
De restrictie op de randpunten zorgt voor een gereguleerd voorwerp.



Verschil gewone en regulerende boolse operaties

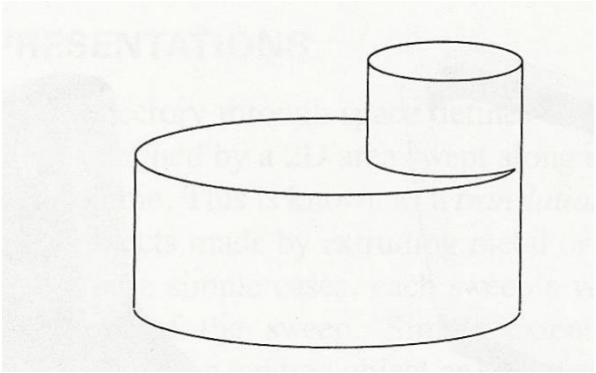
- Regulerende boolse operatoren gedefinieerd als gewone boolse operaties op basis van boundaries (b) en interiors (i):

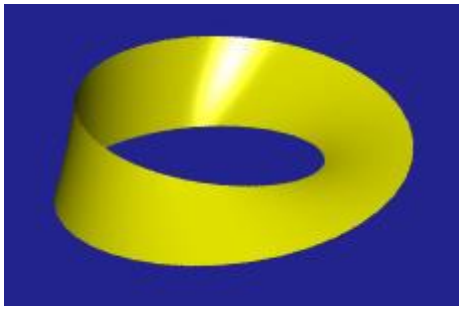




5. Boundary representations

- Ook wel “B-reps” genoemd
- Voorwerpen worden beschreven in termen die hun *omhullende* beschrijven:
  - punten (*vertices*), randen (*edges*) en zijden (*faces*)
  - De omhullende begrenst het interieur van het exterieur van het voorwerp
- Het kan wel eens lastig zijn om te bepalen wat een zijde is:

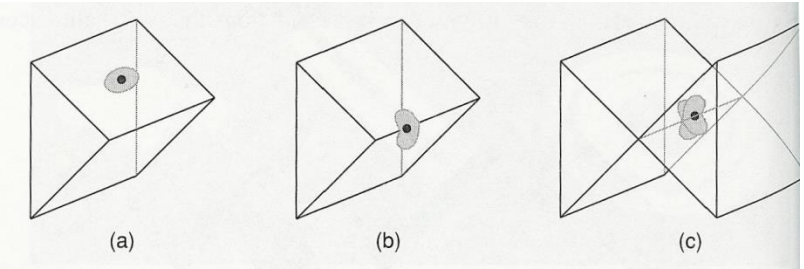




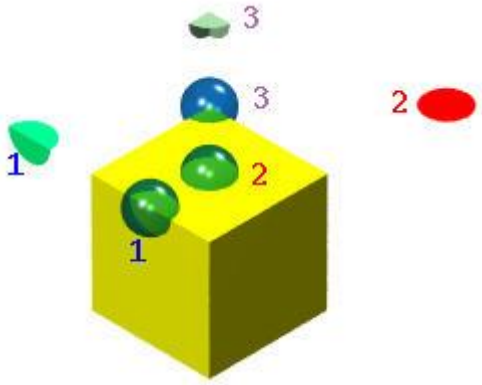
Hoeveel zijden?

5. Boundary representations

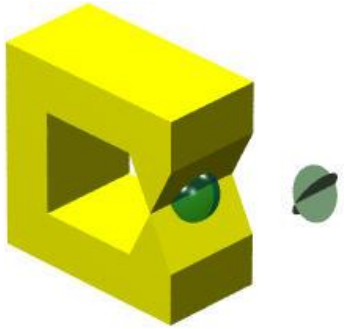
- Veel B-rep systemen laten alleen volumes toe waarvan de boundaries “2-manifold” zijn:



- Elk punt in een 2-manifold object heeft een omgeving die de vorm heeft van (of teruggevouwen kan worden tot) een vlak zonder overlappende stukken
  - (a) en (b) zijn wel 2-manifold
  - (c) is niet 2-manifold (meer dan twee zijden hebben een gemeenschappelijke rand)



Wel 2-manifold

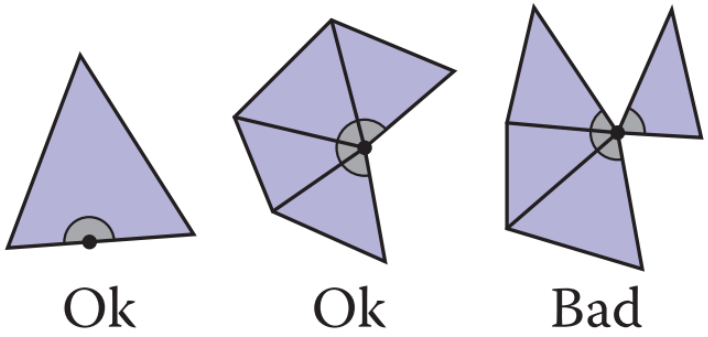


Niet 2-manifold

5. Boundary representations

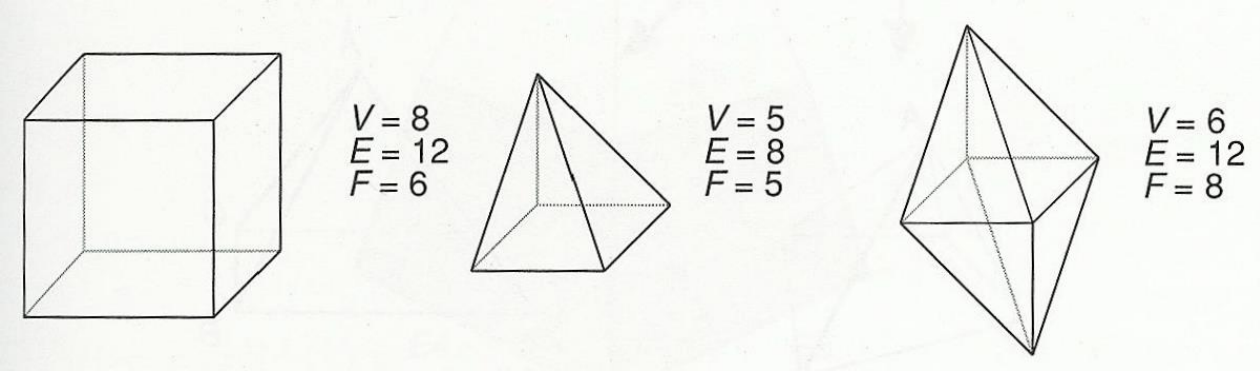
2-manifold: hoe controleer je dat?

- Elke kant wordt door één of twee driehoeken gebruikt
- Elk punt is verbonden met één enkele verzameling driehoeken die aan de rand zijn verbonden



B-rep van polyhedra

- Een polyhedron (veelvlak) is een volume dat is begrensd door een verzameling polygonen waarvan elke rand een even aantal polygonen heeft (precies 2 in het geval van een 2-manifold)
  - Een eenvoudige polyhedron (geen gaten) kan vervormd worden tot een bol

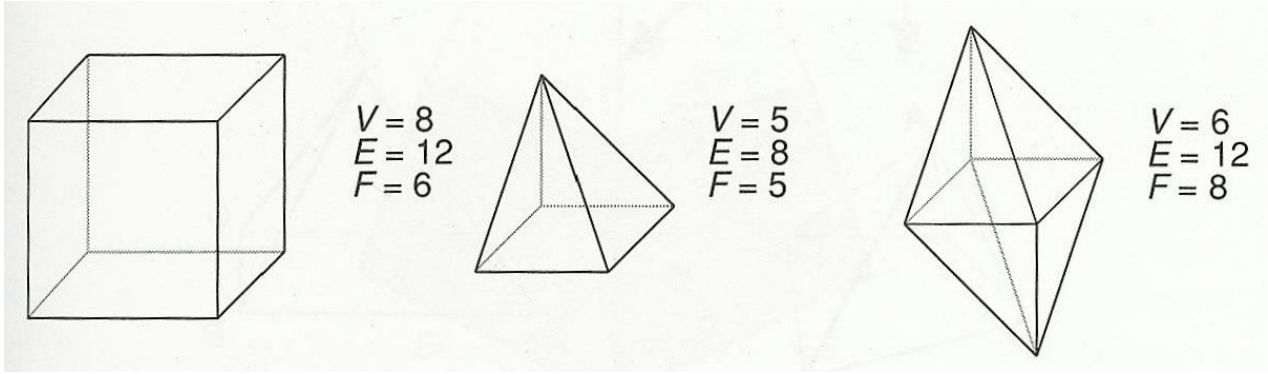


B-rep van polyhedra

- Om een voorwerp met polyhedra te beschrijven moet het volgende waar zijn:
  - Elke rand verbindt twee punten en wordt door precies twee zijden gedeeld
  - Tenminste drie randen komen samen in elk punt
  - Zijden mogen elkaar niet doorsnijden
- Euler: elk polyhedron voldoet aan de volgende relatie:

$$V - E + F = 2$$

waar V (*Vertices*): aantal punten, E (*Edges*): aantal randen,
F (*Faces*): aantal zijden

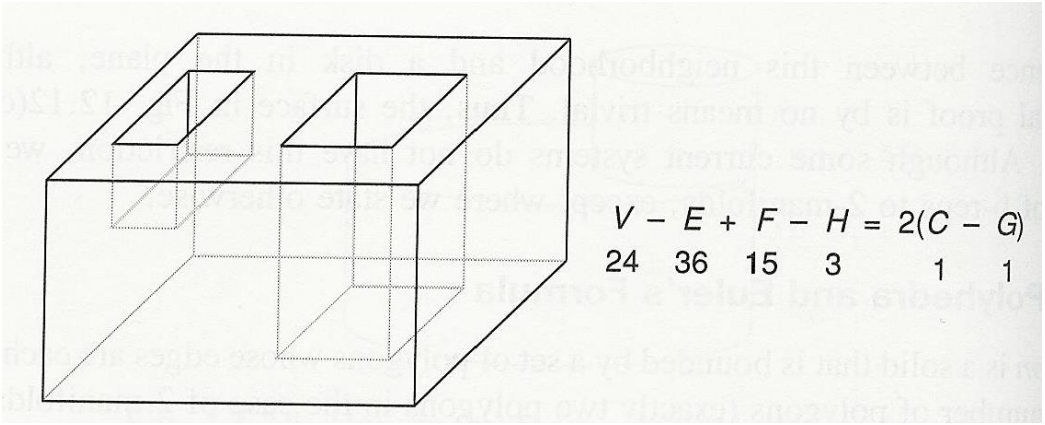


B-rep van polyhedra

- Er bestaat een generalisatie van Euler's formule voor 2-manifolds met zijden met gaten:

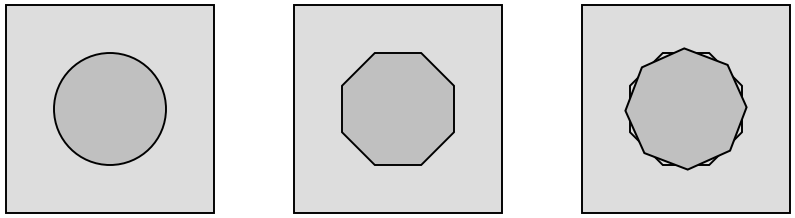
$$V - E + F - H = 2(C - G)$$

waar H (*Holes*): aantal gaten in zijden,
G (*Genus*): aantal gaten door het voorwerp,
C (*Components*): aantal aparte componenten in het voorwerp



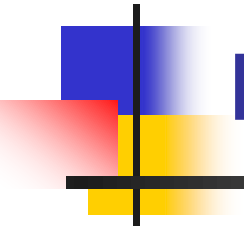
Non-polyhedron b-reps

- Polyhedra kunnen niet-polyhedra oppervlakken alleen benaderen
- Cylinder gerepresenteerd door polyhedron:



(a) (b) (c)

a) Doorsnede van cylinder in een rond gat
b) Polygoon benadering van gat en cylinder
c) Cylinder gedraaid: polyhedron wordt doorsneden: mag niet

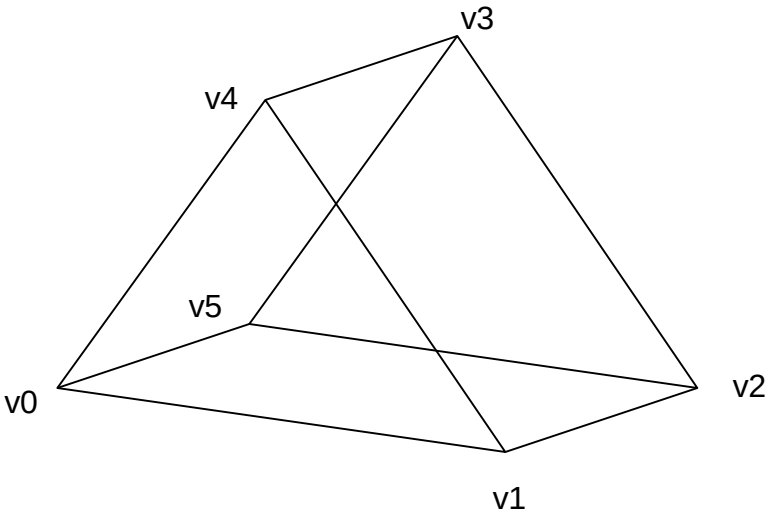


Datastructuren

1. Polygoon meshes
2. Winged-edge
3. Spatial partitioning representations

1. Polygoon meshes

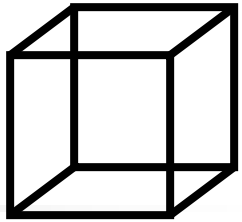
- De envoudigste manier om een verzameling van verbonden polygonen weer te geven is door punten (vertices) en zijden (faces)
- Datastructuur bevat twee lijsten:
  - Lijst van alle punten (geometrie)
  - Lijst van alle zijden (topologie)



v0 v1 v4 = F0
v5 v3 v2 = F1
v1 v2 v3 v4 = F2
v0 v4 v3 v5 = F3
v0 v5 v2 v1 = F4

Let op de volgorde!
Hier: tegen de klok in.

Kubus in OpenGL

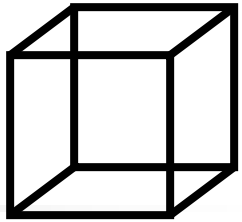


```
glBegin(GL_QUADS);

  /* top of cube */
  glVertex3f( 1.0f,  1.0f, -1.0f);
  glVertex3f(-1.0f,  1.0f, -1.0f);
  glVertex3f(-1.0f,  1.0f,  1.0f);
  glVertex3f( 1.0f,  1.0f,  1.0f);

  /* bottom of cube */
  glVertex3f( 1.0f, -1.0f,  1.0f);
  glVertex3f(-1.0f, -1.0f,  1.0f);
  glVertex3f(-1.0f, -1.0f, -1.0f);
  glVertex3f( 1.0f, -1.0f, -1.0f);

  /* front of cube */
  glVertex3f( 1.0f,  1.0f,  1.0f);
  glVertex3f(-1.0f,  1.0f,  1.0f);
  glVertex3f(-1.0f, -1.0f,  1.0f);
  glVertex3f( 1.0f, -1.0f,  1.0f);

  /* back of cube. */
  glVertex3f( 1.0f, -1.0f, -1.0f);
  glVertex3f(-1.0f, -1.0f, -1.0f);
  glVertex3f(-1.0f,  1.0f, -1.0f);
  glVertex3f( 1.0f,  1.0f, -1.0f);

  /* left of cube */
  glVertex3f(-1.0f,  1.0f,  1.0f);
  glVertex3f(-1.0f,  1.0f, -1.0f);
  glVertex3f(-1.0f, -1.0f, -1.0f);
  glVertex3f(-1.0f, -1.0f,  1.0f);

  /* Right of cube */
  glVertex3f( 1.0f,  1.0f, -1.0f);
  glVertex3f( 1.0f,  1.0f,  1.0f);
  glVertex3f( 1.0f, -1.0f,  1.0f);
  glVertex3f( 1.0f, -1.0f, -1.0f);

glEnd()
```

Hier: expliciete definitie van elk punt en elk vlak.

Merk op: van alle vlakken zijn de punten tegen de klok in geörienteerd.

Kubus in OpenGL

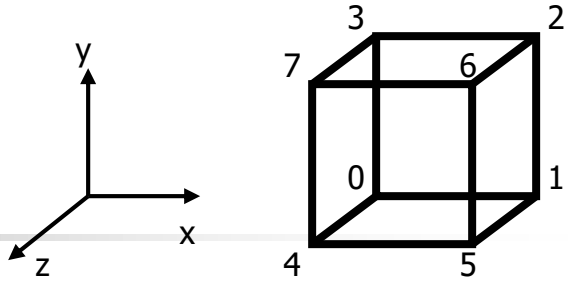


```
face() {
   glBegin(GL_QUADS);
   glVertex3f(-1.0, -1.0f,  1.0f);
   glVertex3f( 1.0, -1.0f,  1.0f);
   glVertex3f( 1.0,  1.0f,  1.0f);
   glVertex3f(-1.0,  1.0f,  1.0f);
   glEnd();
}

cube() {
  face();
  glRotatef(90.0, 0.0, 1.0, 0.0);
  face();
  glRotatef(90.0, 0.0, 1.0, 0.0);
  face();
  glRotatef(90.0, 0.0, 1.0, 0.0);
  face();
  glRotatef(90.0, 0.0, 1.0, 0.0);

  glRotatef(90.0, 1.0, 0.0, 0.0);
  face();
  glRotatef(180.0, 1.0, 0.0, 0.0);
  face();
}
```

Hier: één vlak gedefinieerd, kubus wordt opgespannen door zes getransformeerde vlakken.

Merk op: van alle vlakken zijn de punten tegen de klok in geörienteerd.

Kubus in OpenGL




```
// Geometry:
float vertices[8][3] = {
  {-1.0, -1.0, -1.0 },
  { 1.0, -1.0, -1.0 },
  { 1.0,  1.0, -1.0 },
  {-1.0,  1.0, -1.0 },
  {-1.0, -1.0,  1.0 },
  { 1.0, -1.0,  1.0 },
  { 1.0,  1.0,  1.0 },
  {-1.0,  1.0,  1.0 }
};

// Topology of a face:
face(int a, int b, int c, int d) {
  glBegin(GL_QUADS);
  glVertex3fv(vertices[a]);
  glVertex3fv(vertices[b]);
  glVertex3fv(vertices[c]);
  glVertex3fv(vertices[d]);
  glEnd();
}

// Topology of a cube:
Cube() {
  face(0,3,2,1);
  face(2,3,7,6);
  face(0,4,7,3);
  face(1,2,6,5);
  face(4,5,6,7);
  face(0,1,5,4);
}
```

Hier: geometrie en topologie zijn gescheiden.

Merk op: van alle vlakken zijn de punten tegen de klok in geörienteerd.

2. Winged-Edge

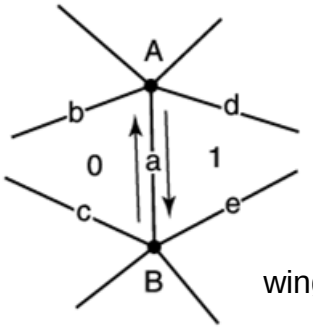
Voor toepassingen waar de polygoon mesh niet verandert is een eenvoudige datastructuur voldoende.

Maar wat nu als je de volgende vragen wilt beantwoorden:

- Wat zijn de drie naburige driehoeken van een driehoek?
- Welke twee driehoeken delen een rand?
- Welke zijden delen een punt?
- Welke randen delen een punt?
- ...

Winged edge datastructuur beschrijft geometrie en topologie van:

- edges (randen)
- vertices (punten)
- triangles (zijden)



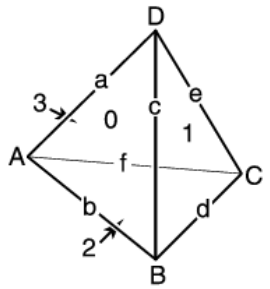
winged edge data structure

edge	vertex 1	vertex 2	face left	face right	pred left	succ left	pred right	succ right
a	B	A	0	1	c	b	d	e

2. Winged-Edge

- Zoeken door polygoon meshes is nu goedkoper
- Datastructuur is relatief klein

- Naast de “edge” tabel worden vaak twee extra tabellen gebruikt:
  - Vertices tabel
  - Faces tabel



edge	vertex 1	vertex 2	face left	face right	pred left	succ left	pred right	succ right
a	A	D	3	0	f	e	c	b
b	A	B	0	2	a	c	d	f
c	B	D	0	1	b	a	e	d
d	B	C	1	2	c	e	f	b
e	C	D	1	3	d	c	a	f
f	C	A	3	2	e	a	b	d

vertex	edge
A	a
B	d
C	d
D	e

face	edge
0	a
1	c
2	d
3	a

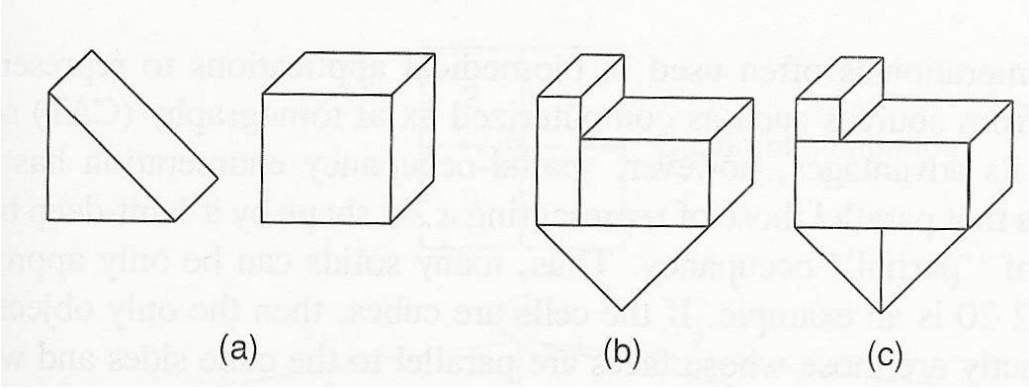
3. Spatial-partitioning representaties

Voorwerp wordt gerepresenteerd door aaneengrenzende, niet-overlappende volumes

1. Cell decomposition
2. Spatial-occupancy enumeration
3. Quadtrees, octrees
4. Binary space-partioning trees

3.1. Cell decomposition

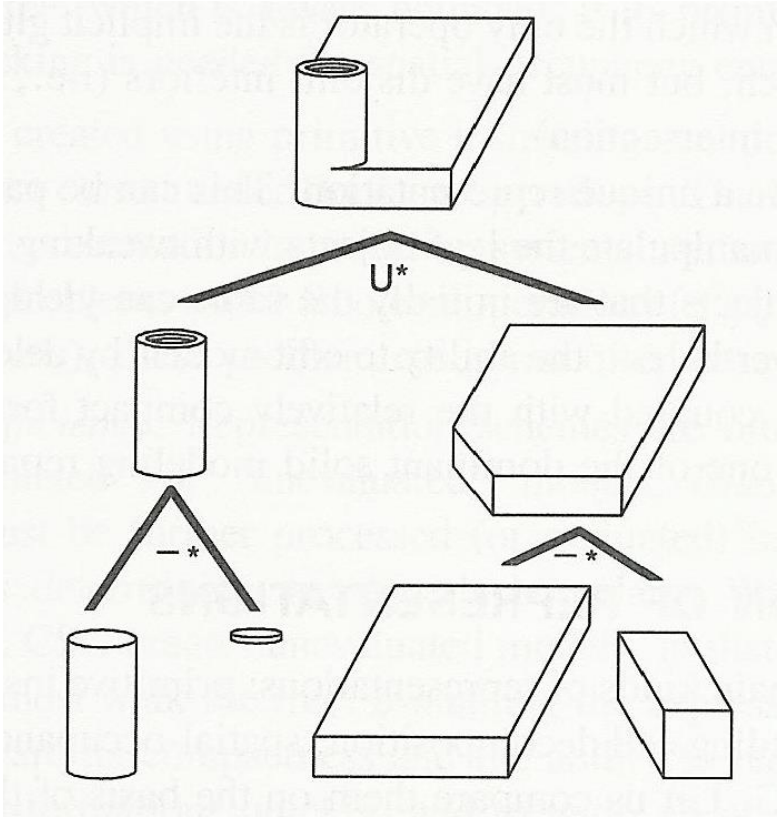
- Verzameling primitieven die aan elkaar gelijmd worden tot complexe objecten
  - Primitieven: blok, wig, cylinder, bol, ...
  - Elke twee primitieven delen een punt, rand of zijde
  - Objecten mogen niet overlappen (i.e. alleen $\cup$)
  - De representatie is ondubbelzinnig
  - De representatie is niet uniek
- Onder andere gebruikt in eindige elementen methode (Finite Element/Volume Methods, FEM/FVM)

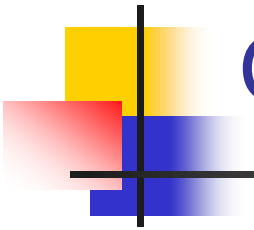


3.1. Cell decomposition

Constructive Solid Geometry (CSG)

- Combineren van primitieven met regulerende boolse operatoren
- Voorwerp is opgeslagen als een boom
  - Knoop bevat boolse operator of eenvoudige transformatie (translatie, rotatie, schaling)
  - Bladeren bevatten primitieven



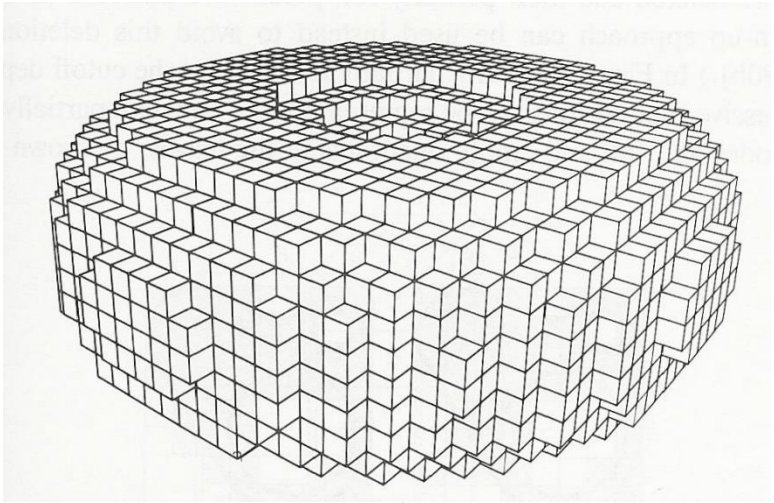


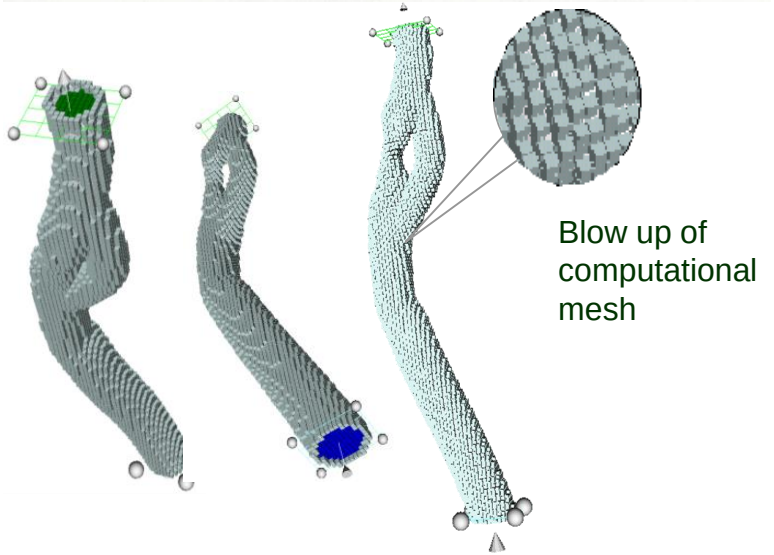
Constructive Solid Geometry

- Primitieven:
  - Kubussen, bollen, cylinders, etc: elke bewerking levert een regulier voorwerp op
  - Half-spaces; niet begrensde volumes
    - Kubus: doorsnede van 6 half-planes
    - Moeilijk te valideren
  - Cell-decomposition en spatial-occupancy enumeration zijn speciale gevallen van CSG met alleen $\cup$
  - CSG is niet uniek

3.2. Spatial-occupancy enumeration

- Cuberilles: cellen worden weergegeven door identieke cellen: blokjes (voxels) in een regelmatig rooster
- Alleen aangeven of cel *wel* of *niet* zichtbaar moet zijn
- Min-of-meer een 3D bitmap
  - Nadeel: nauwkeurigheid afhankelijk van cel grootte
- Wordt soms gebruikt in biomedische topassingen
  - CT, lattice Boltzman
- De representatie is ondubbelzinnig
- De representatie is uniek!

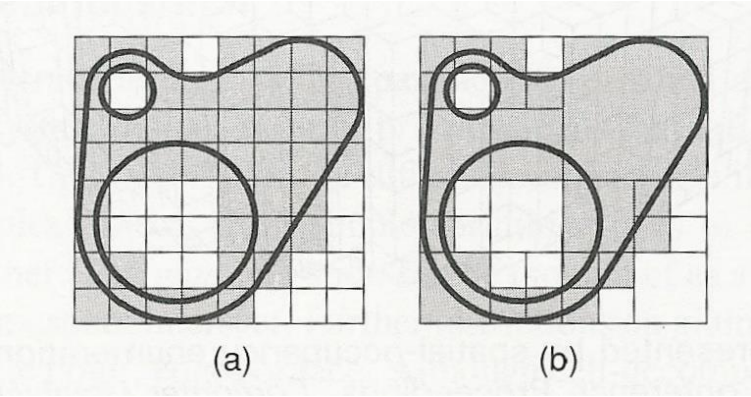




Blow up of computational mesh

3.3. Quadtrees/Octrees

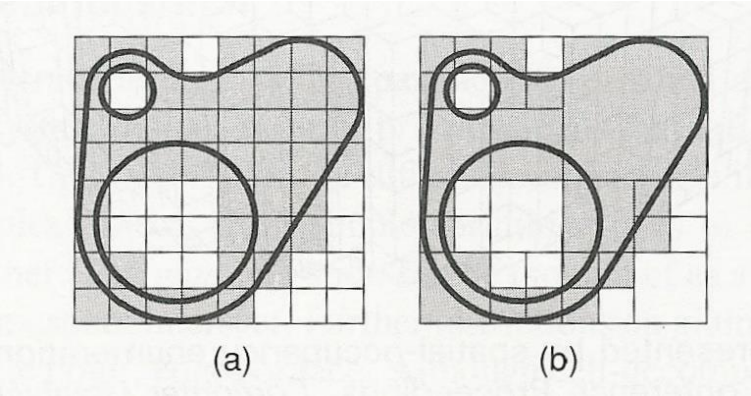
- Hierarchische datastructuur
  - 2D: quadtrees
  - 3D: octrees
- We gebruiken ter illustratie een 2D quadtree:

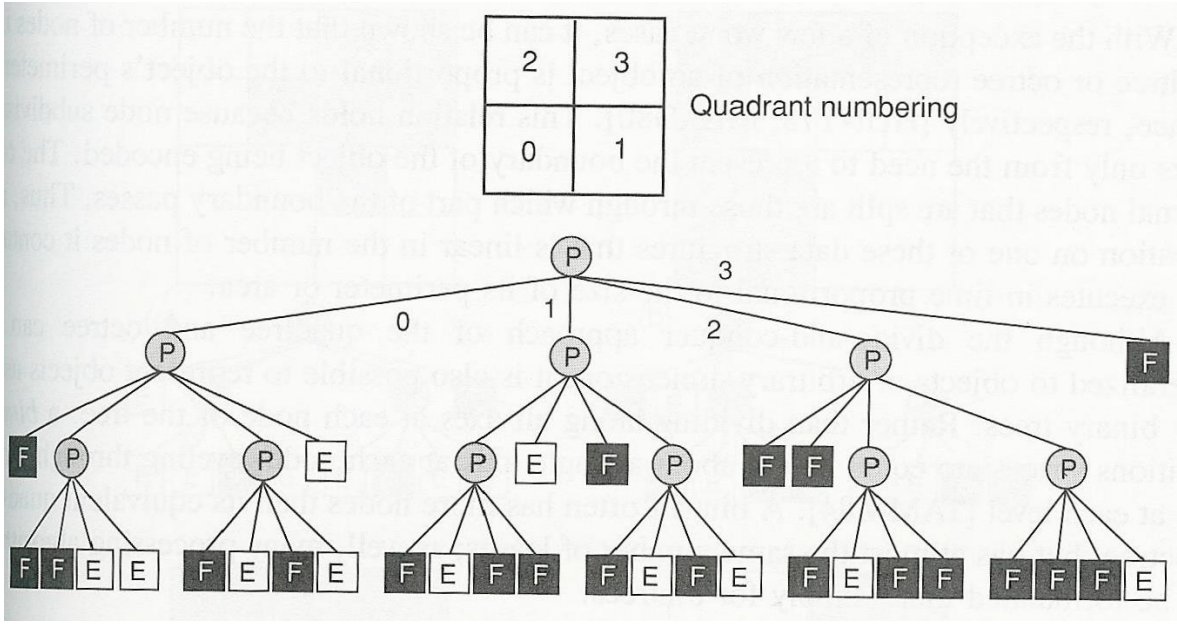


a) Spatial occupancy enumeration
b) Quadtree

- Vlak wordt verdeeld in kwadranten
  - Leeg of vol: klaar
  - Gemend: ga door
  - Recursie
- Elk kwadrant is een knoop in de boom
  - Als leeg of vol: eindpunt (blad)
  - Als gemengd: verder opgedeeld

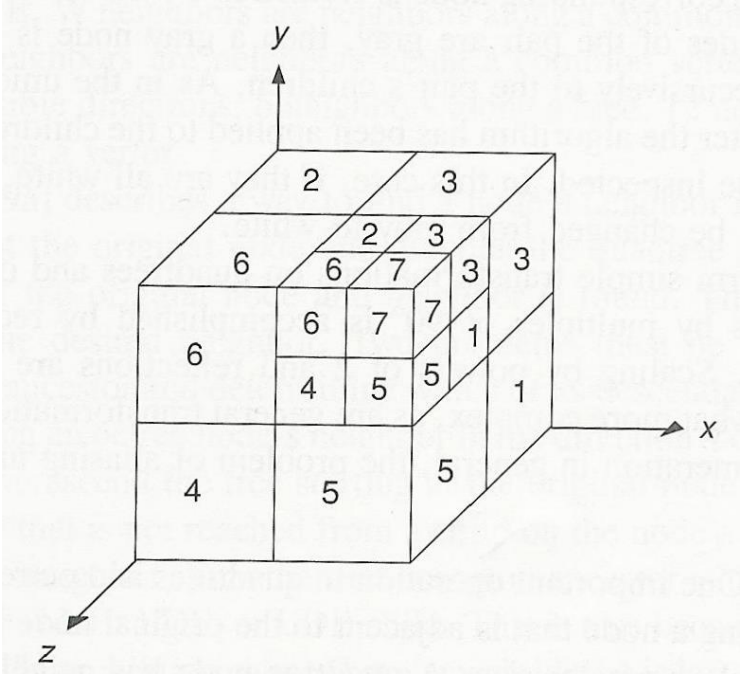
3.3. Quadtrees/Octrees





3.3. Quadtrees/Octrees

- Octree: 8 sub-blokken
  - Octant 0 is hier niet zichtbaar
- Efficient op te slaan en te verwerken
- Boolse verzameling operatoren zijn eenvoudig toe te passen

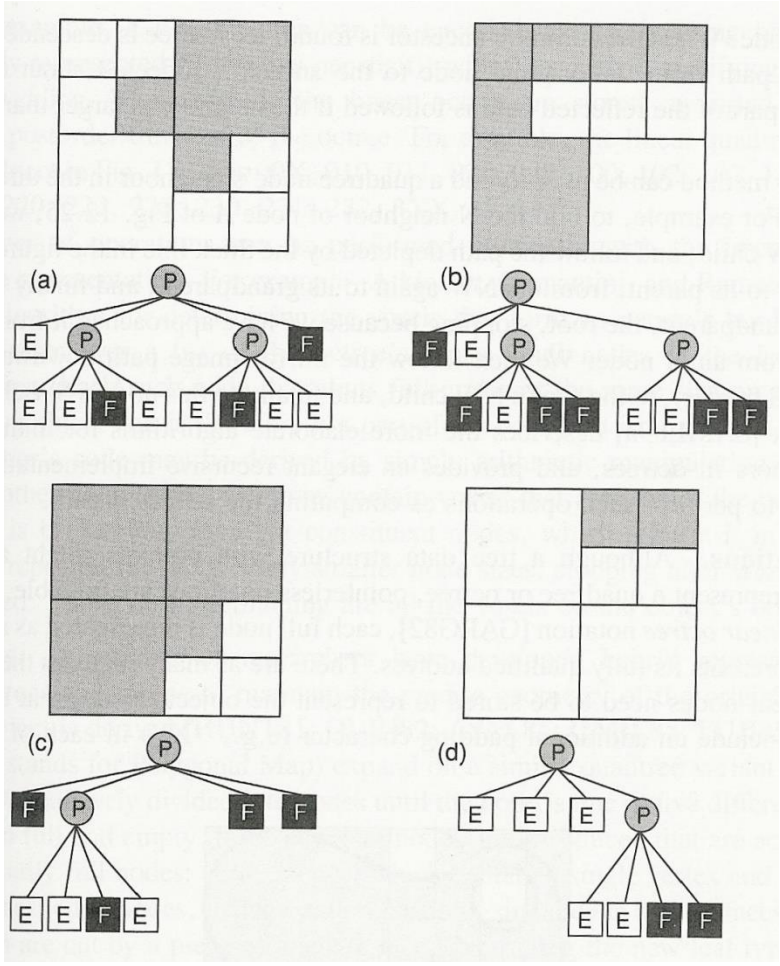


3.3. Quadtrees/Octrees

- Octree: 8 sub-blokken
  - Octant 0 is hier niet zichtbaar
- Efficient op te slaan en te verwerken
- Boolse verzameling operatoren zijn eenvoudig toe te passen:

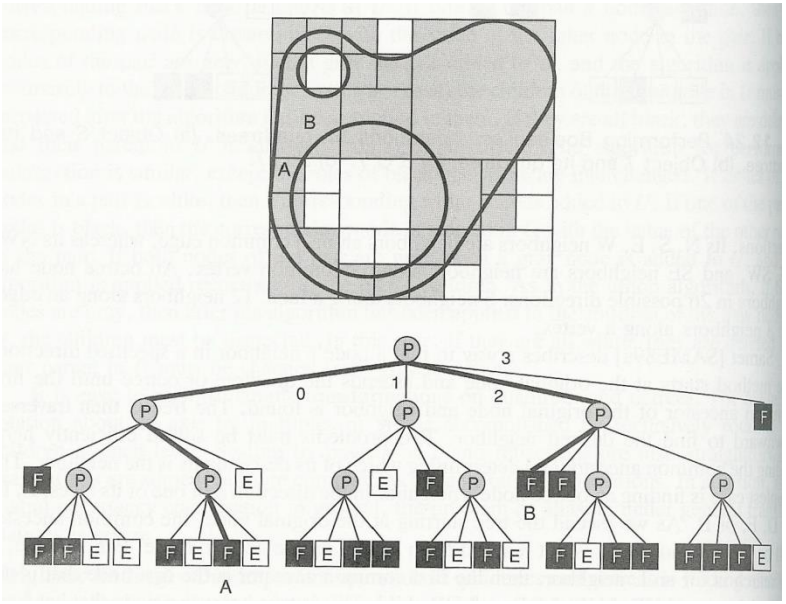
  Als voorbeeld op een quadtree:

  a) Voorwerp S
  b) Voorwerp T
  c) S ∪ T
  d) S ∩ T

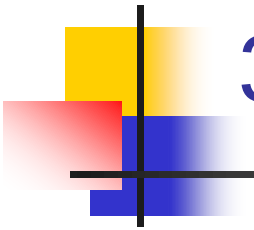


3.3. Quadtrees/Octrees

- Zoeken van de buur ten noorden van A
  - up up up (root) 2 0 (=leaf B)
- Het aantal nodes in een quadtree of octree is proportioneel met resp. de rand of de omhullende van het object:
  - omdat subdivisie alleen plaatsvindt om een begrensing interieur/exterieur weer te geven
  - een subdivisie vindt alleen plaats waar een omhullende een octant/quadrant doorsnijdt



3.4. Binary space-partioning trees

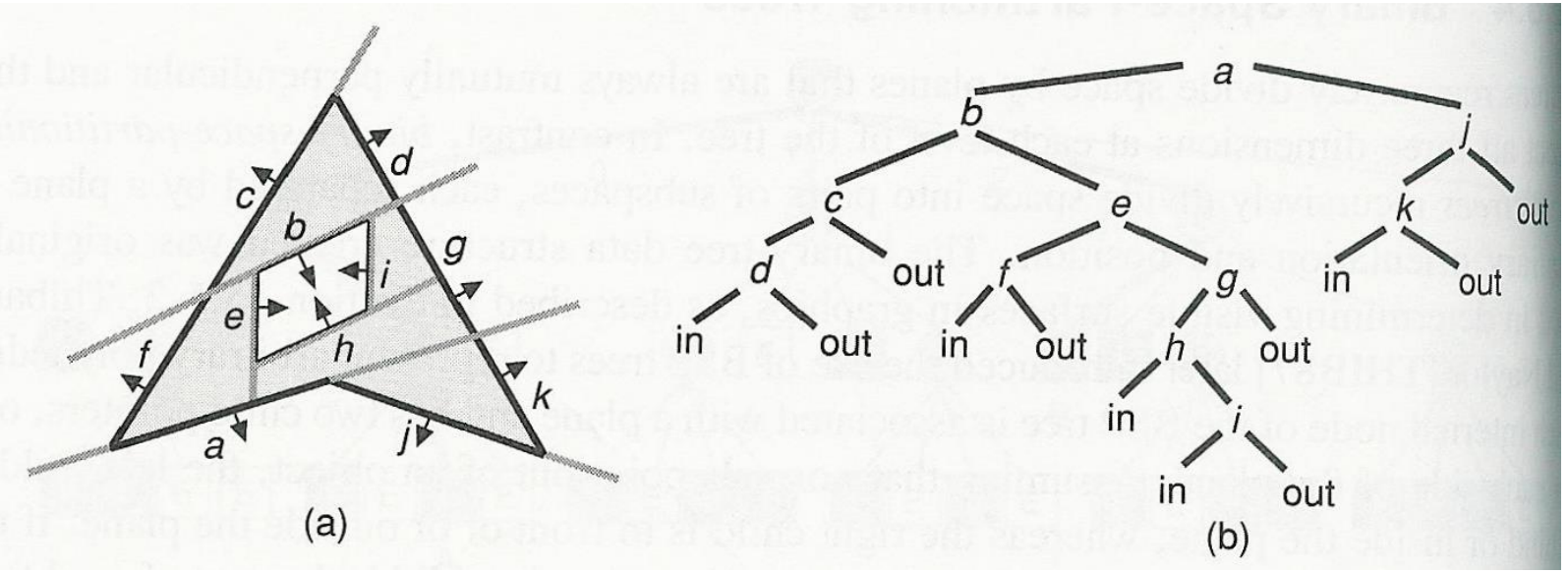


- BSP tree verdeelt voorwerp op in twee delen aan weerszijden van een vlak
  - Recursief
- Elke knoop representeert een vlak
- Linker kind representeert de “achterkant” van het vlak
- Rechter kind representeert de “voorkant” van het vlak
- Als een kind niet verder opgedeeld kan worden dan is het ofwel “binnen” (in) of “buiten” (out), anders is het de wortel van een subtree

- Een BSP tree kan gebruikt worden om een concaaf volume te representeren als de vereniging van convexe “in” gebieden.
  - “Tesselation”

3.4. Binary space-partioning trees

- Hier weergegeven in 2D (: opdeling d.m.v. lijnen i.p.v. vlakken)
  - Concaaf polygoon omsloten door zwarte lijnen
  - Donkergrijs: snijlijnen, normalen wijzen naar buiten
  - Lichtgrijze delen: “binnen” (in)
  - Elke snijlijn is een bisectie in de boom: de ene tak bevat alle elementen ‘voor’ de snijlijn, de andere tak alle elementen die er ‘achter’ liggen



Vergelijking

	Accuracy	*Domain*	*Uniqueness*	*Validity checking*	*Closure*
Primitive instancing	Yes	Limited	Possible	Easy	No
Sweeps	Yes	Limited	Possible	Easy	No
b-reps poly	Approximation	Limited	No	Difficult	Yes
b-reps non-poly	Yes	Any	No	Difficult	Yes
Cell decomposition	Approximation	Any	No	Difficult	Yes
Spatial occupancy enumeration	Approximation	Any	Yes	Easy	Yes
Octrees	Approximation	Any	Yes	Easy	Yes
BSP trees	Approximation	Any	No	Easy	Yes
CSG	Yes	Any	No	Easy	Yes

Software

Commercieel:

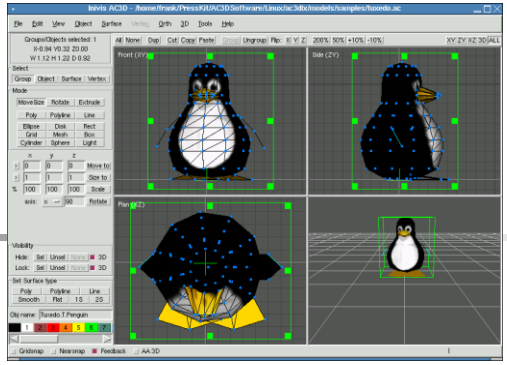
- Maya
- Softimage
- Lightwave
- 3D Studio Max
- ...

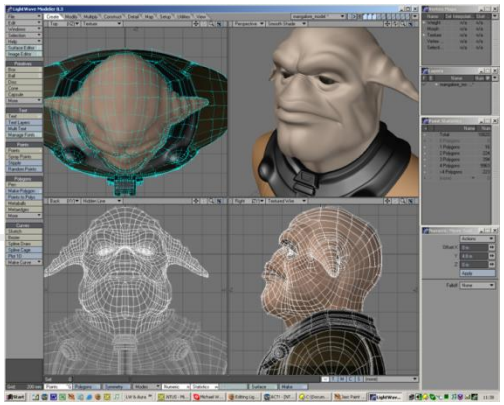
Cheap:

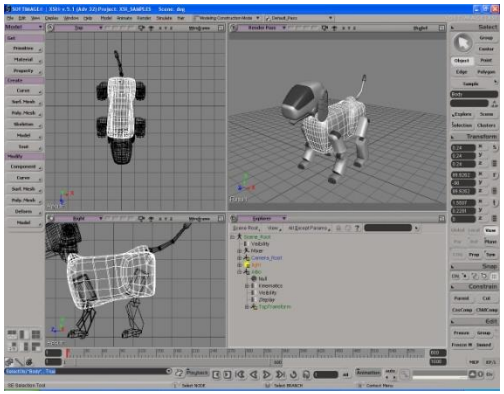
- AC3D
- Google SketchUp
- ...

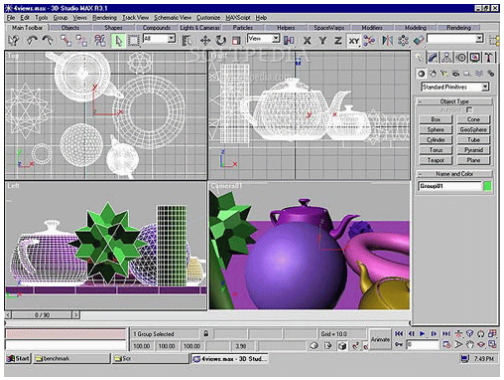
Free:

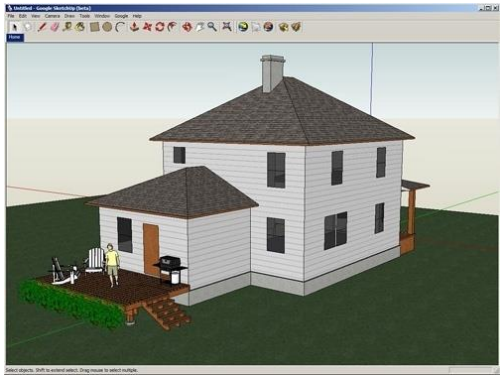
- Blender
- Wings3D
- ...

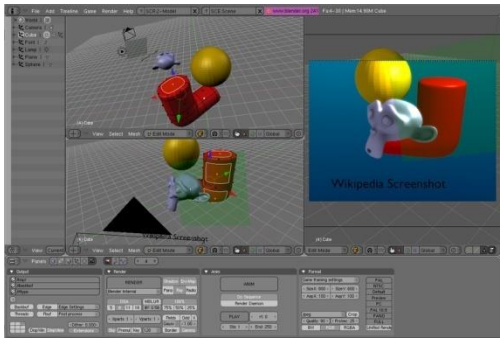


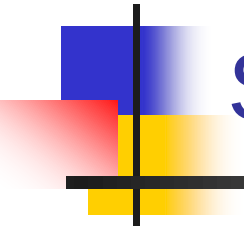










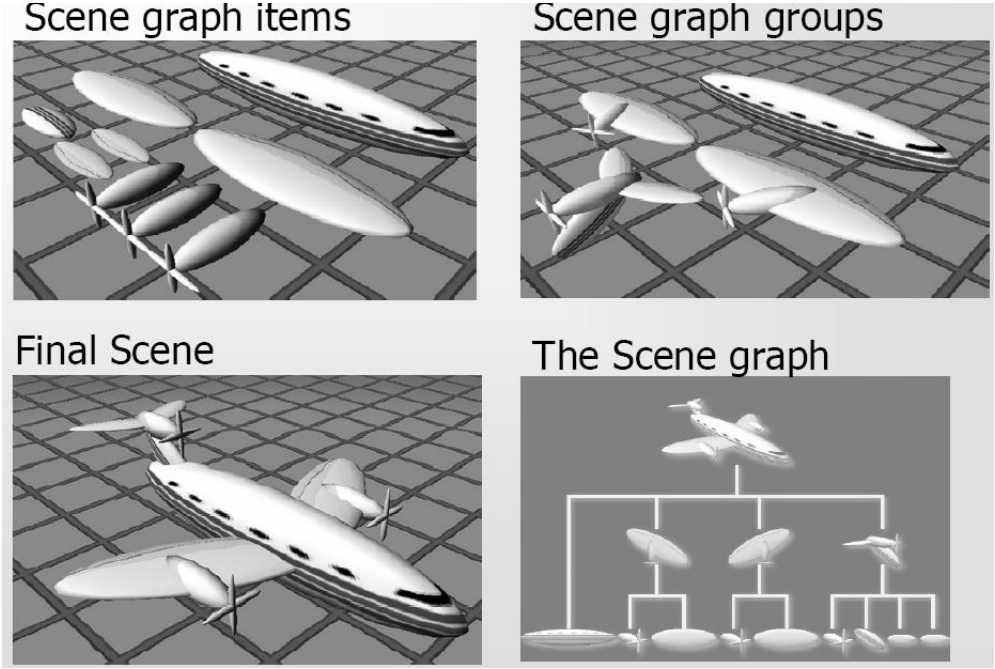


Scenegraphs

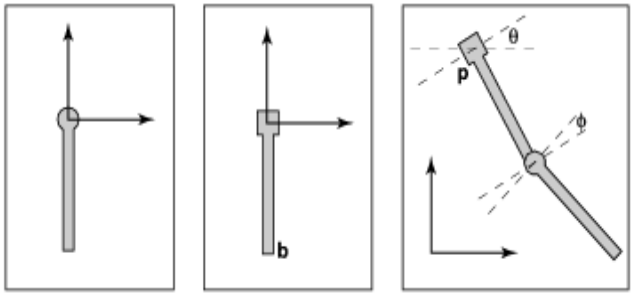
Werelden opgebouwd uit objecten

- “Scenegraphs” definieren een scene (wereld, omgeving)
  - Hierarchische beschrijving van één of meerdere voorwerpen
  - Directed Acyclic Graphs (DAG)

- Voorbeelden:
  - VRML
  - X3D
  - Java3D
  - OpenGL|Performer



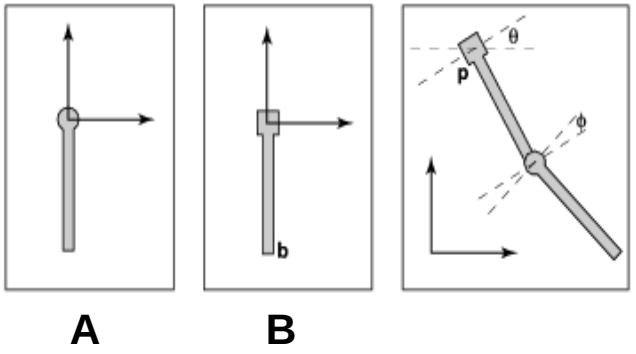
Scenegraphs



Voorbeeld:

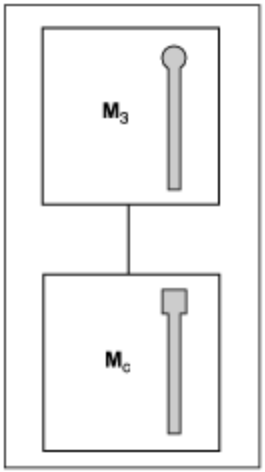
- Een slinger met twee scharnieren
  - Hoe tekenen we die?

Scenegraphs

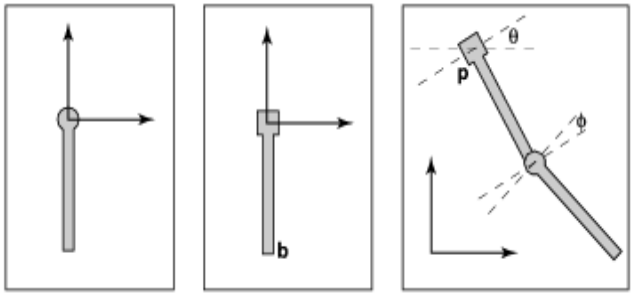


Het bovenste deel (**B**) kunnen we tekenen door:

- M_1=rotate(θ)
- M_2=translate(**p**)
- M_3=M_2M_1
- Pas M_3 toe op alle punten van het bovenste deel



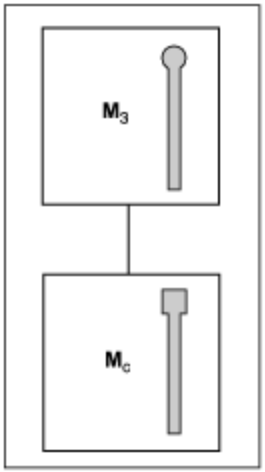
Scenegraphs



A **B**

Het onderste deel (**A**) kunnen we tekenen door:

- M_a=rotate(φ)
- M_b=translate(**b**)
- M_c=$M_b M_a$
- M_d=$M_3 M_c$
- Pas M_d toe op alle punten van het onderste deel

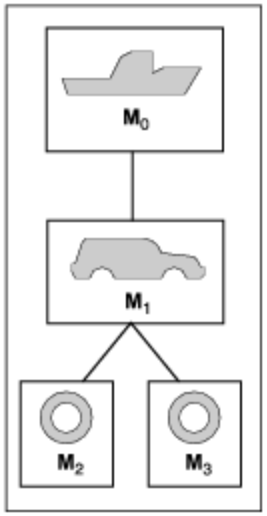


Scenegraphs

Ander voorbeeld:

- Een schip, met daarop een auto, met wielen, …

schip	M_0
carosserie	$M_0\, M_1$
wiel linksvoor	$M_0\, M_1\, M_2$
wiel linksachter	$M_0\, M_1\, M_3$
…	



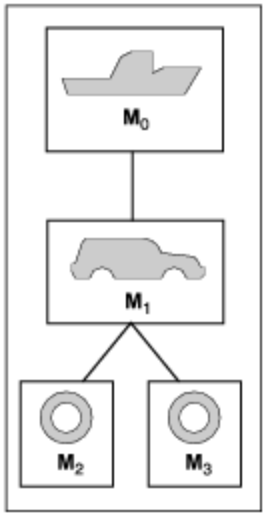
- Directed Acyclic Graph (DAG)
- Transformaties zijn cumulatief
  - Alle voorwerpen onder een transformatie worden beïnvloed

Scenegraphs

De matrix stack: biedt de mogelijkheid om aan de rechterkant van het matrixproduct matrices toe te voegen of te verwijderen

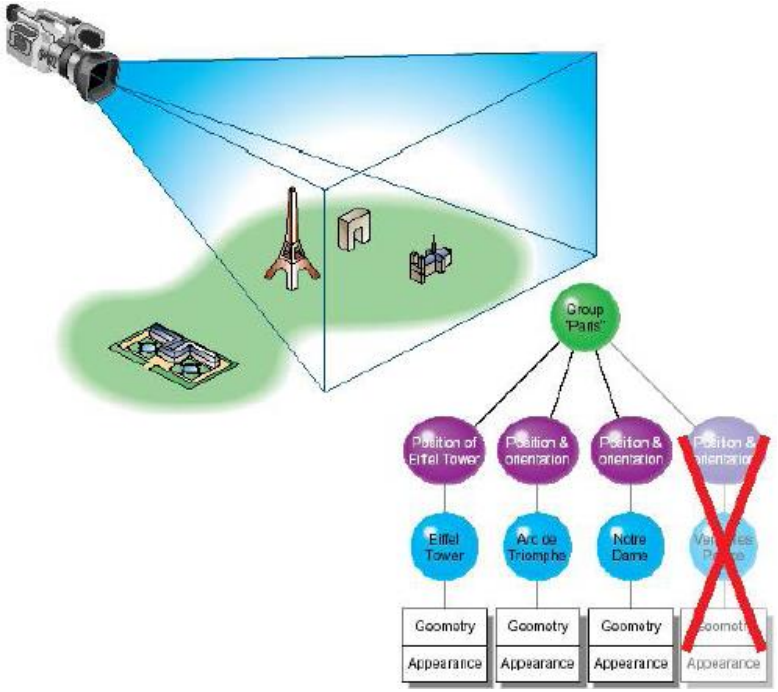
- push() en pop()

```
                                /* M=I */
push(M0)                        /* M=M0 */
    teken(schip)
    push(M1)                    /* M=M0 M1 */
      teken(carosserie)
      push(M2)                  /* M=M0 M1 M2 */
        teken(wiel)             /* linksvoor */
      pop()                     /* M=M0 M1 */
      push(M3)                  /* M=M0 M1 M3 */
        teken(wiel)             /* linksachter */
      pop()                     /* M=M0 M1 */
  pop()                         /* M=M0 */
pop()                           /* M=I */
```



Frustum culling

- Alle voorwerpen in de scenegraph tekenen kan veel werk zijn als de wereld uit veel voorwerpen bestaat
  - Dat kan nadelig zijn als de omgeving responsief en interactief moet zijn (bv. VR)
- Frustum culling: teken alleen die voorwerpen die zichtbaar zijn
  - Iets niet tekenen is goedkoper dan wel tekenen!



Horizon Zero Dawn, Guerrilla Games
Source: https://www.youtube.com/watch?v=A0eaGRcdwpo