Colleges: tweede helft

- Collegestof week 5 t/m 7:
  1. Solid modeling
  2. Texture mapping
  3. Data Visualization
  4. Graphics Hardware
  5. Interactive Graphics Applications
  6. Animation
- Online deeltoets in 8e week op 24 oktober
  - Cijfer moet minimaal 5.0 zijn (zo niet: hertentamen over de *hele* stof)

Verzin een tentamenvraag!

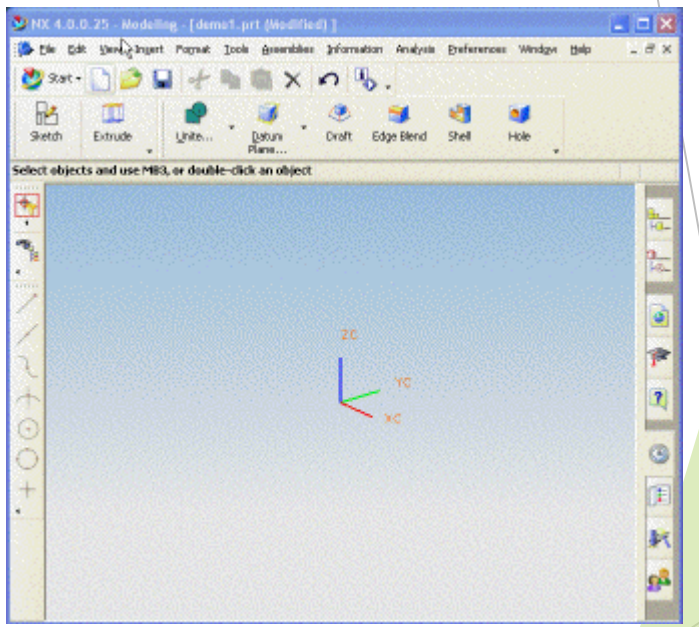
- Verzin een tentamenvraag n.a.v. dit college
- Dien in op dit formulier
- Als je vraag wordt gebruikt tijdens een (deel)tentamen ontvang jij een half punt extra op je eindcijfer voor dat (deel)tentamen (bv.: had je een 7, dan krijg je een 7,5)!*

*: Vragen die voorgaande jaren al eens op een (deel)tentamen voor dit vak zijn gesteld zijn van deze regeling uitgesloten.

Graphics en Game Technologie

7. Solid modeling

Modellering van 3D voorwerpen

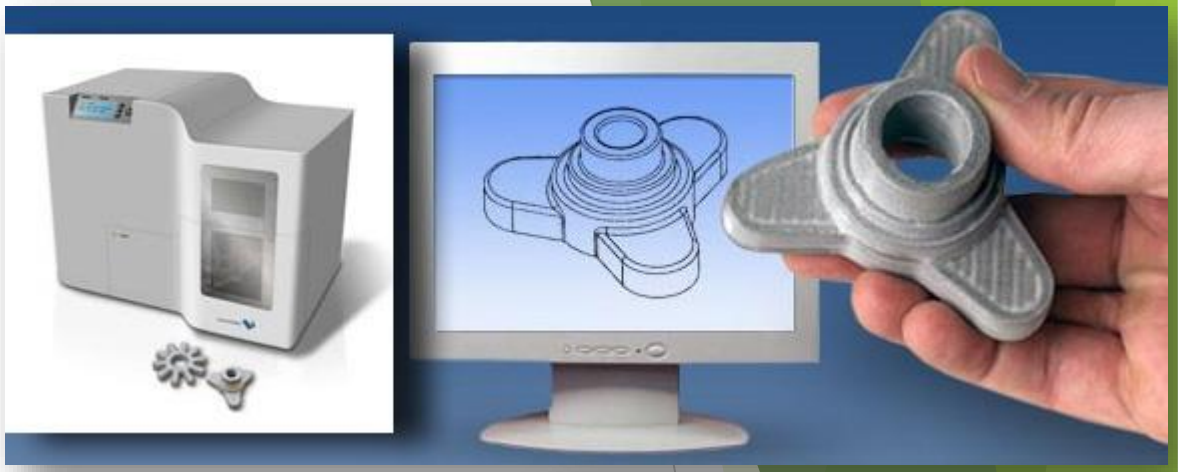


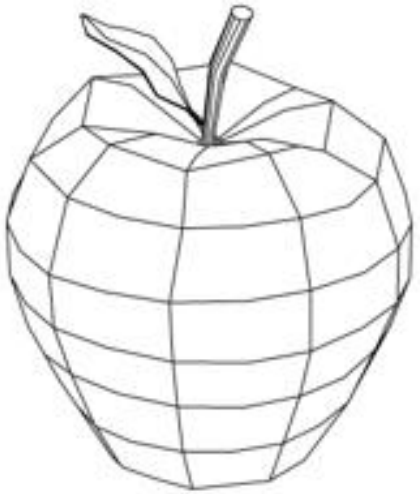
Robert Belleman
Computational Science Lab
Universiteit van Amsterdam
R.G.Belleman@uva.nl
(voeg a.u.b. “[GGT]” toe aan subject)

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

Overzicht

1. Wat is een “voorwerp”?
2. Object representatie methoden
3. Datastructuren
4. Scenegraphs





Voorwerpen

Tot nu toe: voorwerpen gerepresenteerd door lijnen, krommen, polygonen en oppervlakken in 2D en 3D. 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?

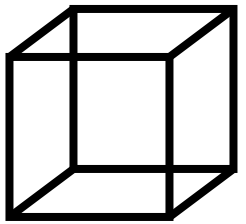
Toepassingen:

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

Representeren van voorwerpen

Gewenste eigenschappen:

- Volledig — Elk denkbaar voorwerp kan worden gerepresenteerd;
- Ondubbelzinnig — Een object kan niet anders worden geïnterpreteerd dan is bedoeld;
- Uniek — Een object kan maar op één manier gemaakt worden (“complete”);
- Correct — Het is niet mogelijk een incorrect voorwerp te maken, het moet eenvoudig zijn een correct voorwerp te maken;
- Nauwkeurig — Het voorwerp is geen benadering maar exact;
- Gesloten — Een voorwerp is na transformatie (translatie, rotatie, schaling, boolean operatoren, etc.) nog steeds een correct voorwerp;
- Compact — Het voorwerp kan compact gerepresenteerd worden;
- Efficiënt — 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.

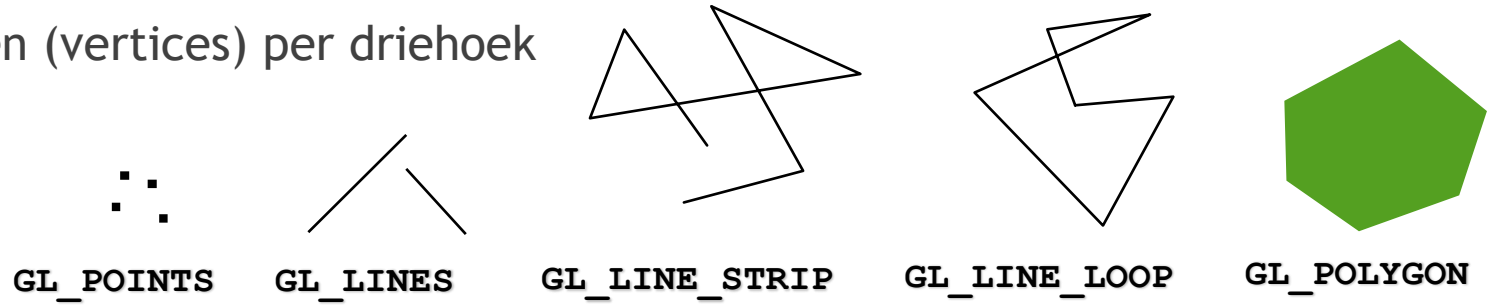
Object representatie methoden

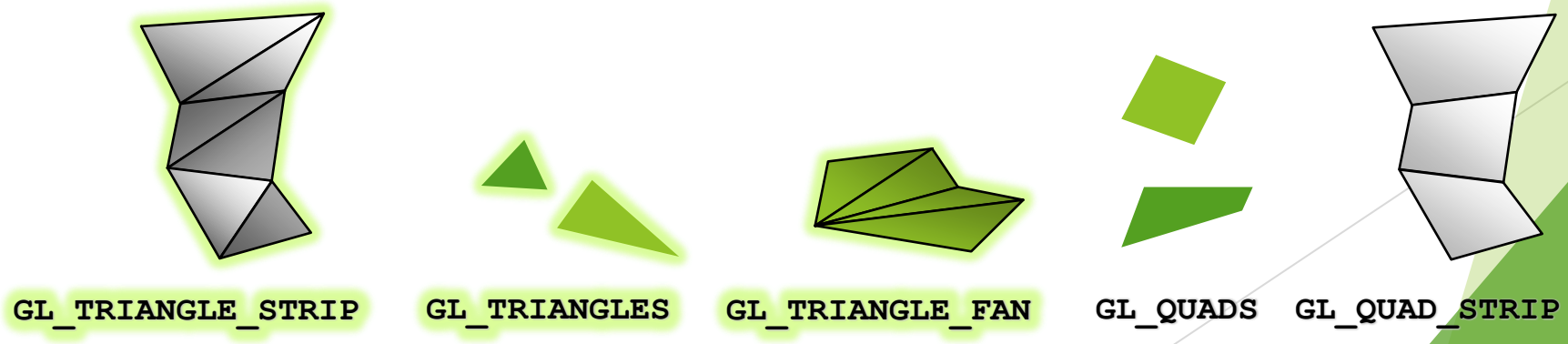
1. Triangle mesh
2. Extrusie methoden
3. Impliciete quadratische oppervlakken
4. Constructieve methoden
5. “Omhullingen” (boundary representations)

1. Triangle mesh

Verzameling verbonden driehoeken definieert een oppervlak

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





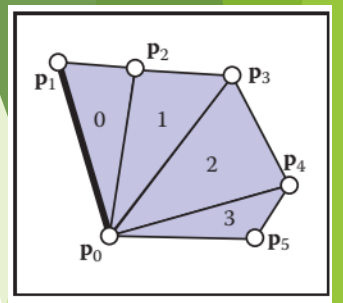


Figure 12.9. A triangle fan.

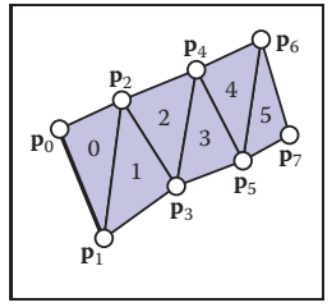
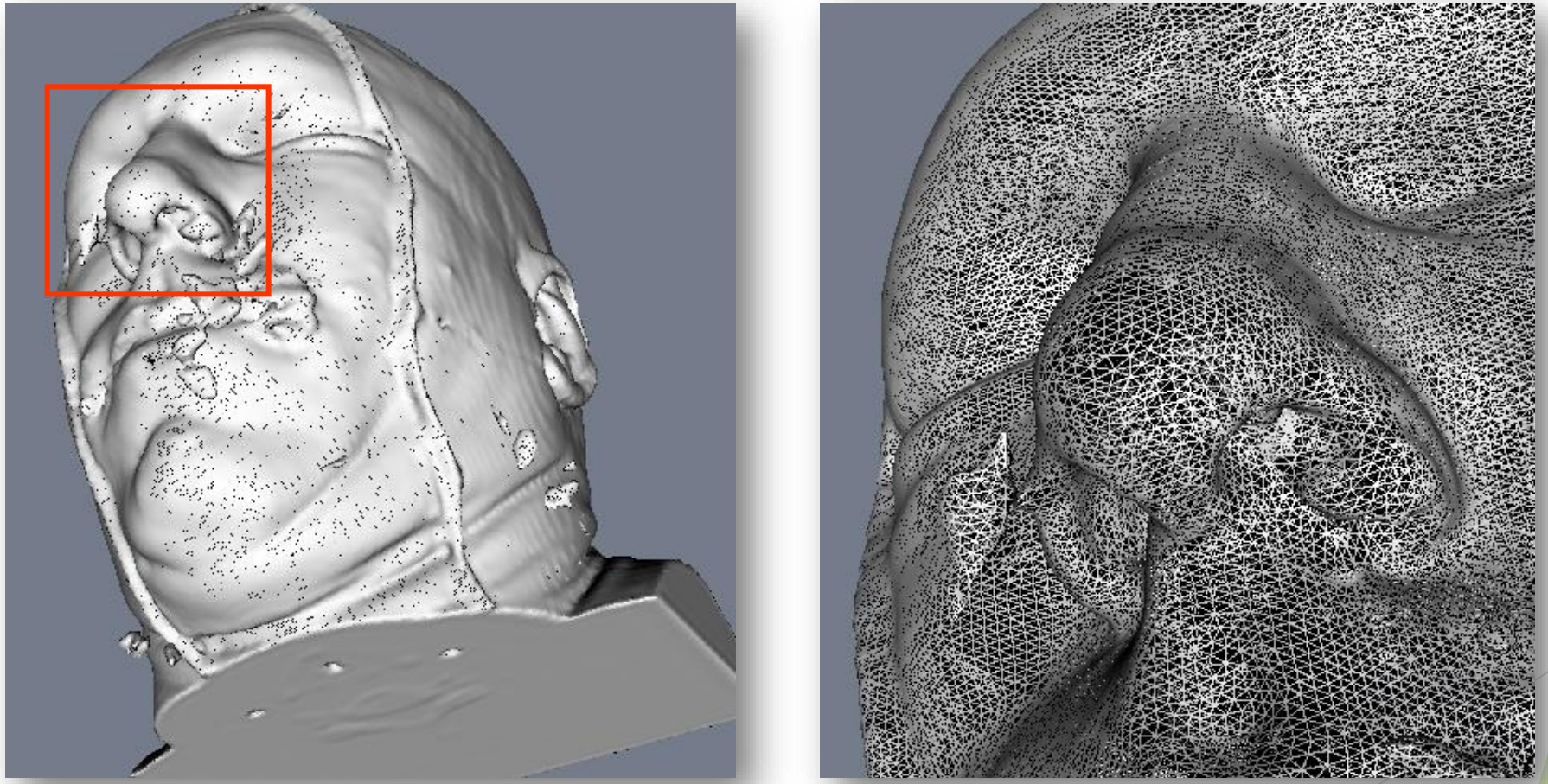


Figure 12.10. A triangle strip.

1. Triangle mesh



Isosurface representation of the Visible male dataset, NIH

1. Triangle mesh: triangle orientation

Onderscheid “binnen” en “buiten” wordt gemaakt op basis van normaalvector en dus op volgorde van punten:

- Clockwise of counterclockwise

Belangrijk dat dit consistent gebeurt:

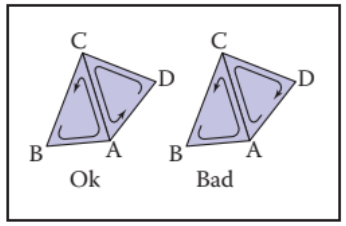
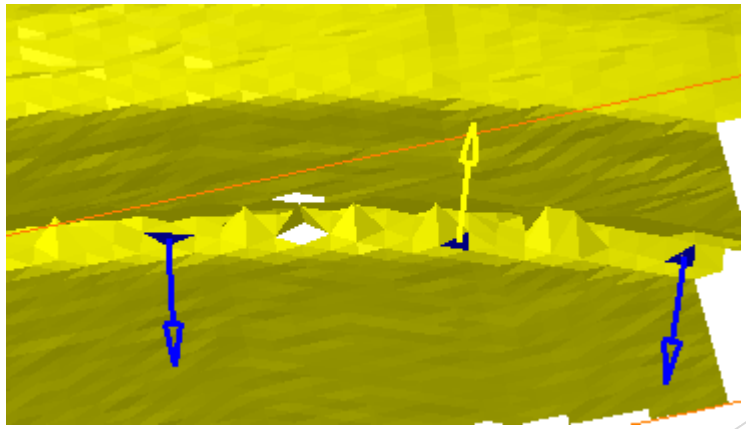


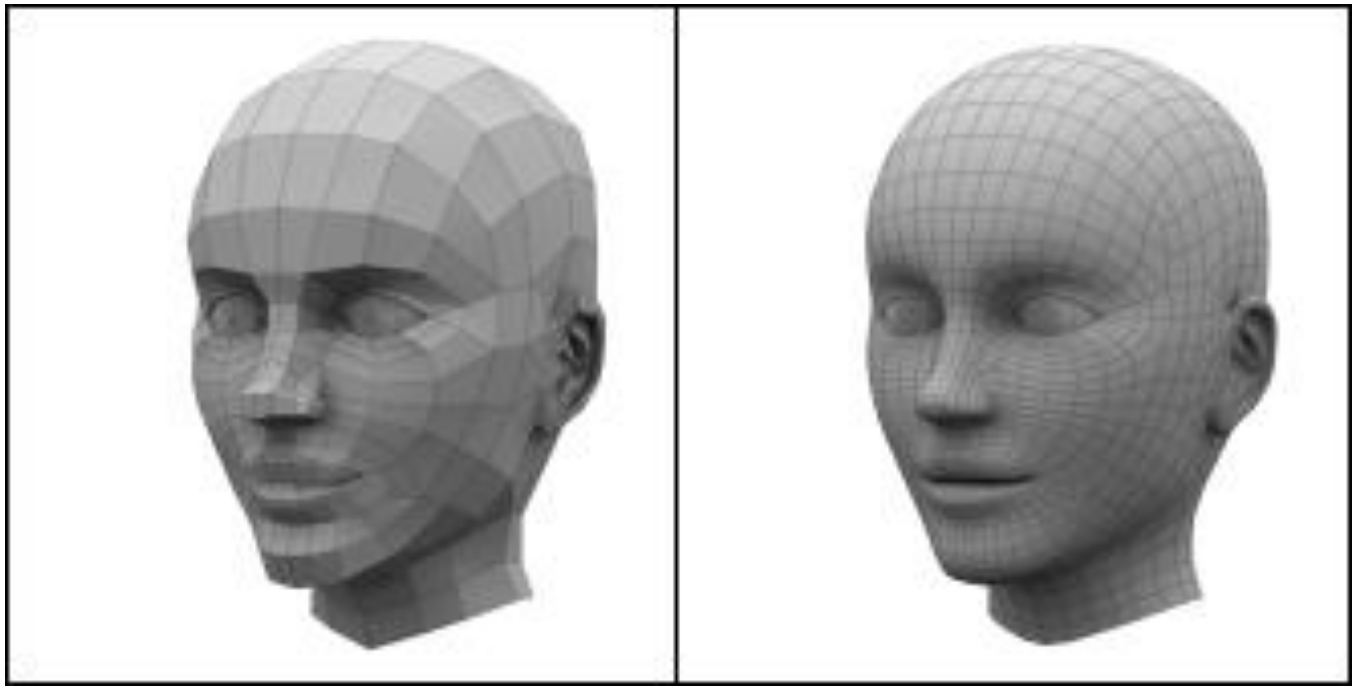
Figure 12.4. Triangles (B,A,C) and (D,C,A) are consistently oriented, whereas (B,A,C) and (A,C,D) are inconsistently oriented.

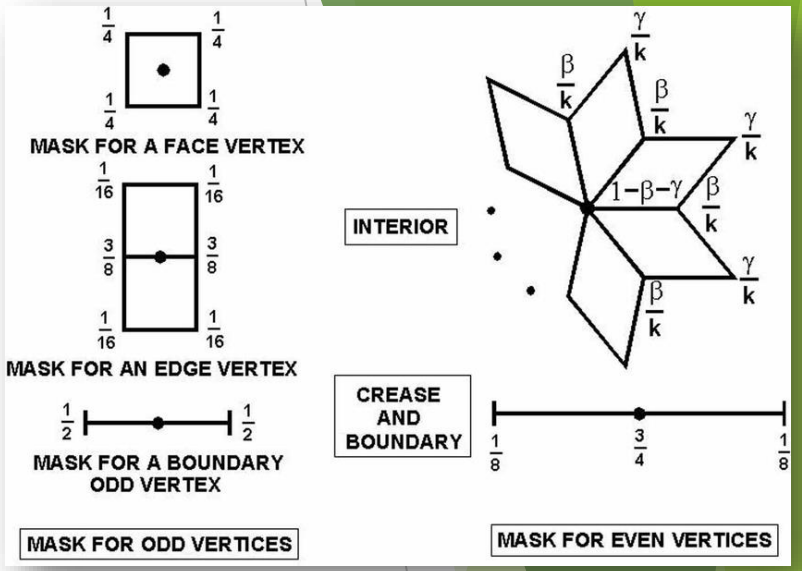


Mogelijk resultaat als inconsistent toegepast

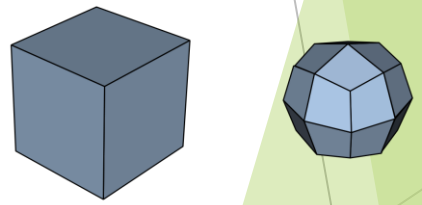
1. Triangle mesh: subdivision

- Modeling by subdivision: Catmull-Clark subdivision





Catmull-Clark subdivision scheme



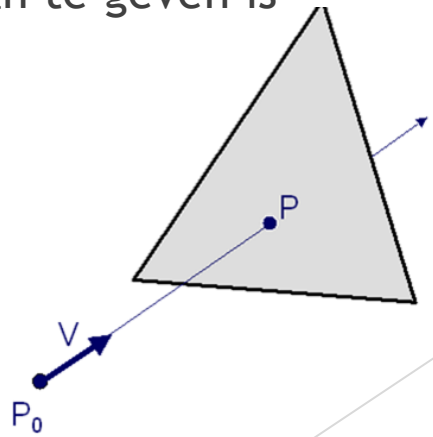
1. Triangle mesh

Voordelen:

- Makkelijke manier om objecten mee te beschrijven
- Driehoeken zijn efficiënt te renderen

Nadelen:

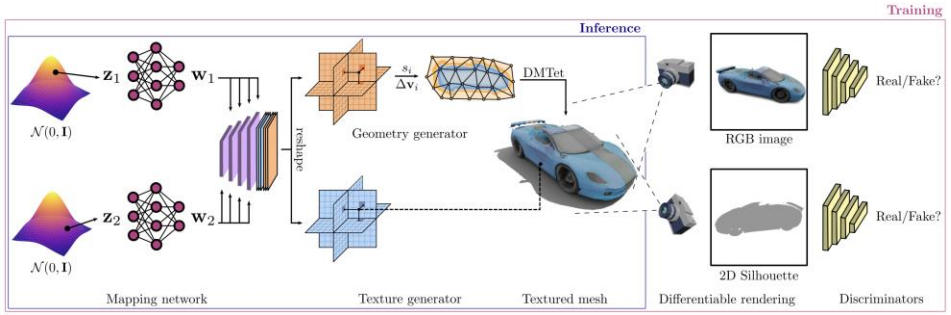
- Mogelijkheid om binnen/buitenkant van objecten aan te geven is ambigu
- Moeilijk om objecten te combineren ($\cap$, $\cup$, -)
- Voor complexe modellen zijn veel driehoeken nodig
  - renderen kan lang duren
  - interactie met een model is niet altijd efficiënt

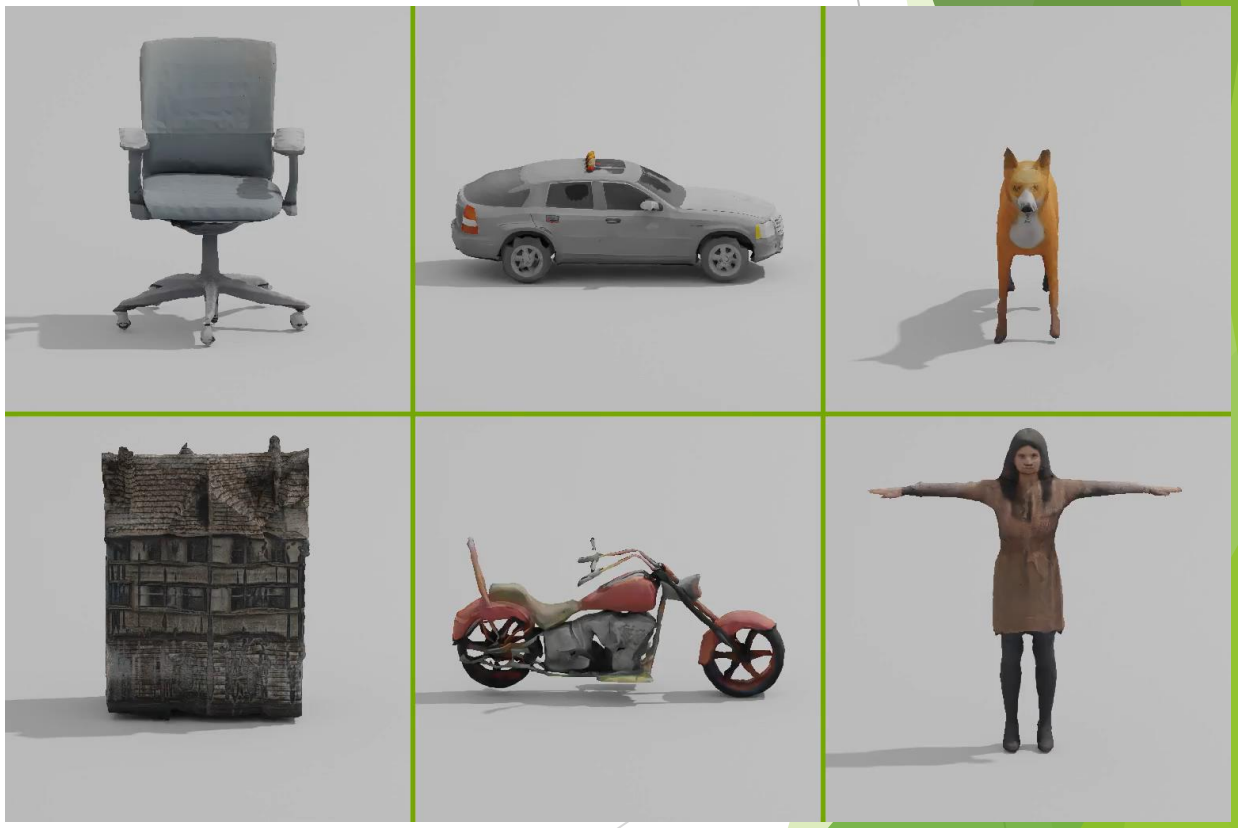


1. Triangle mesh

Bleeding-edge: 3D triangle meshes genereren o.b.v. beschrijvende tekst:

GET3D: A Generative Model of High Quality 3D Textured Shapes Learned from Images (2022)





Source

2. Extrusie: “sweeps”

Algoritme:

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:

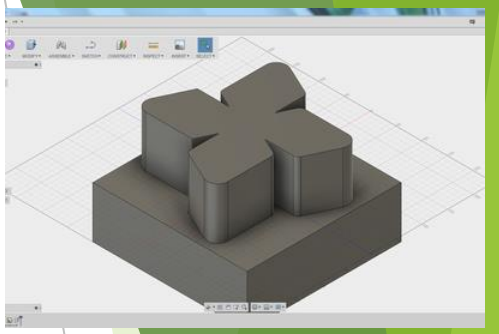
- Translational sweeps (extrusion)
- Rotational sweeps (revolvements)

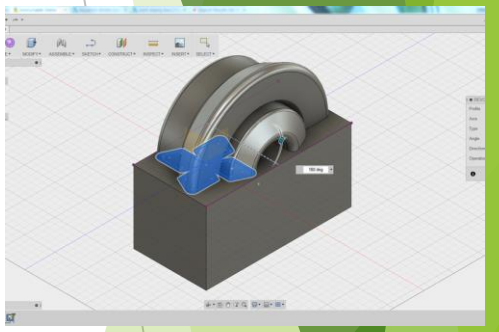
Voordelen:

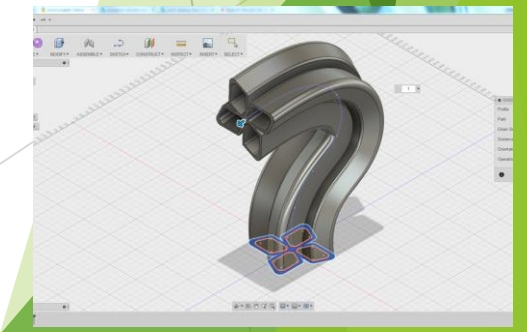
- Conceptueel eenvoudig
- Compact

Nadelen:

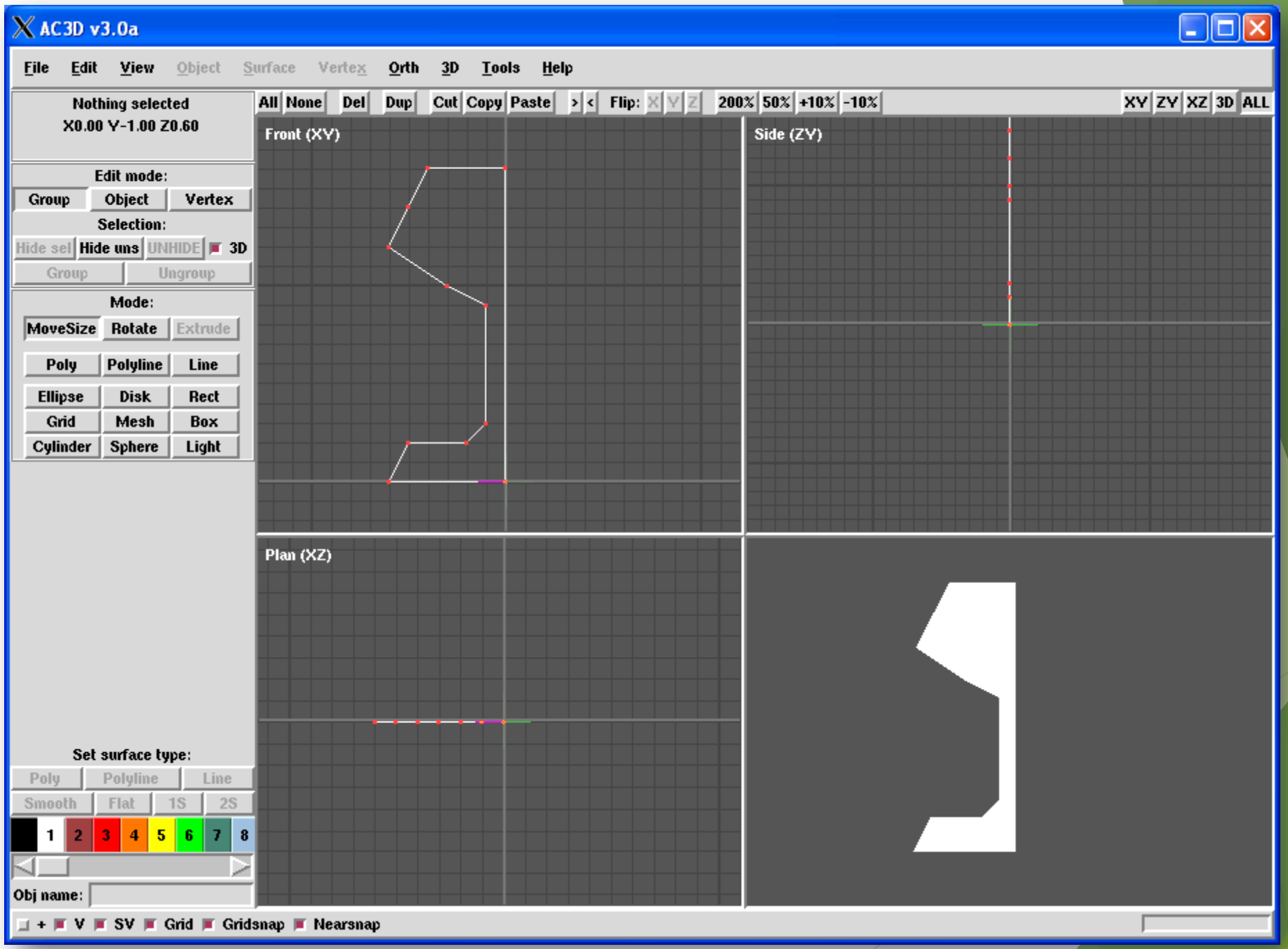
- Moeilijk efficiënt te implementeren
- Kan resulteren in incorrecte objecten
- Sweeps zijn niet gesloten onder de “regulerende boolse operatoren”
- Nauwkeurigheid afhankelijk van resolutie 2D polygoon en pad

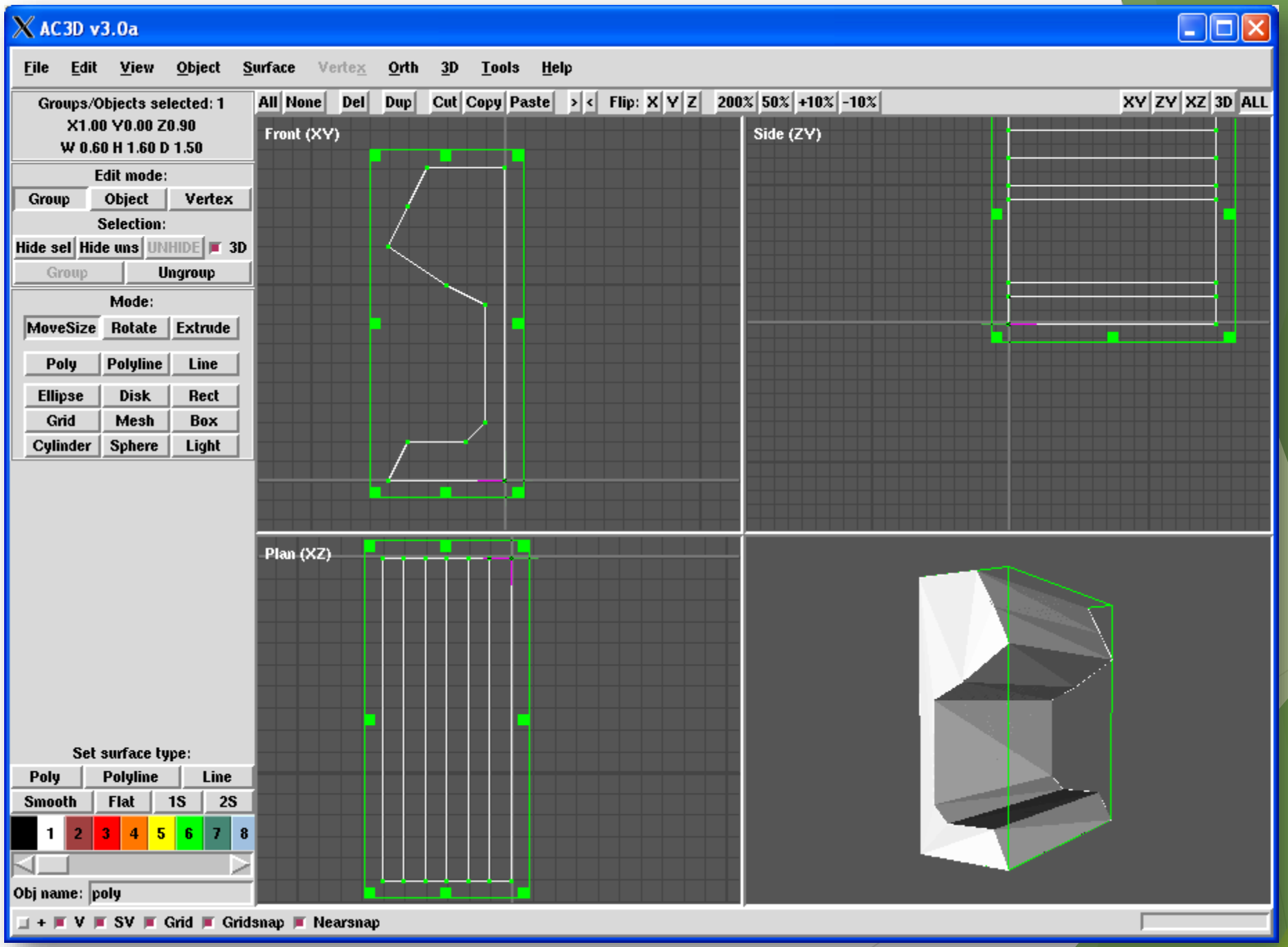


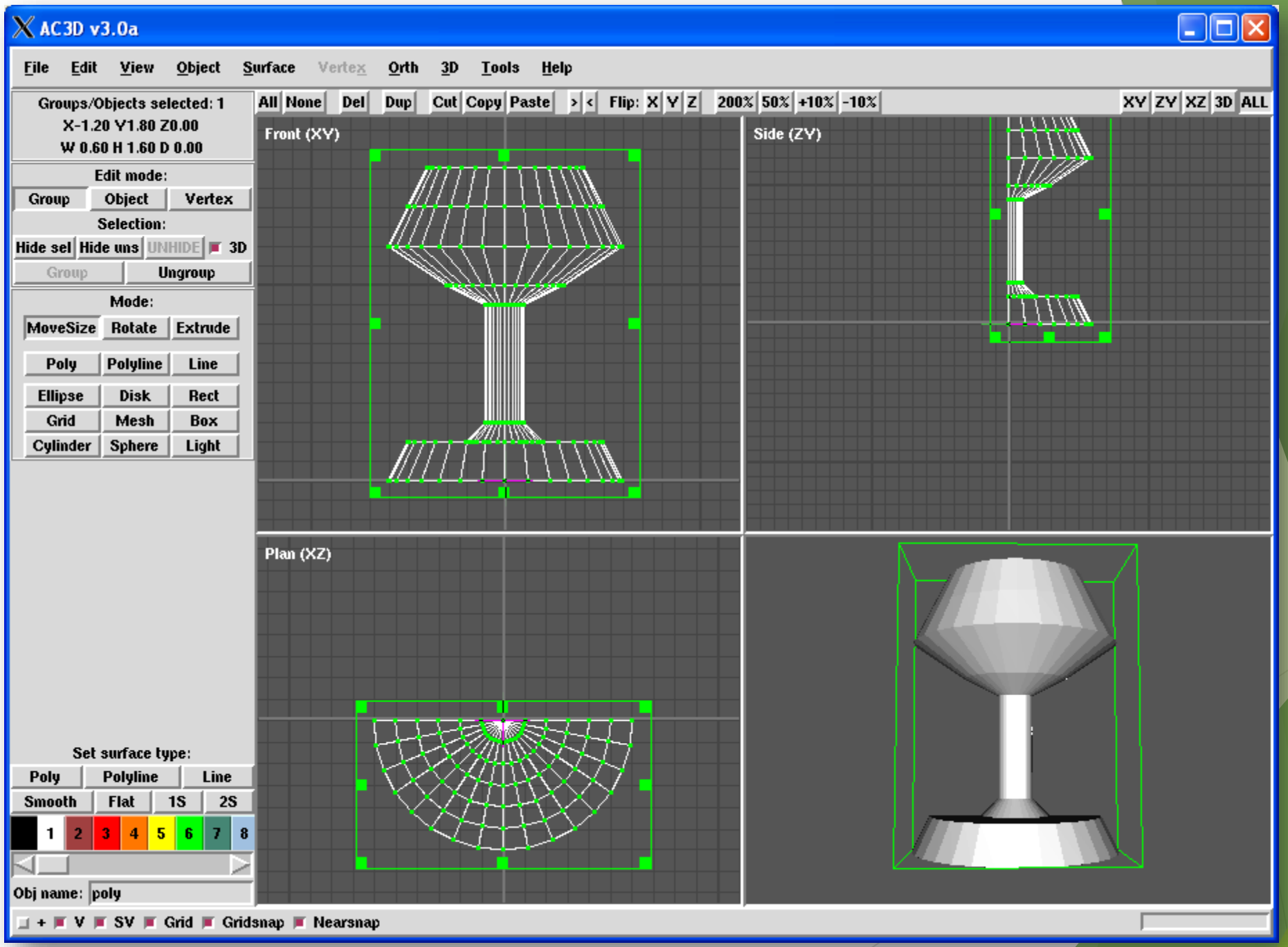


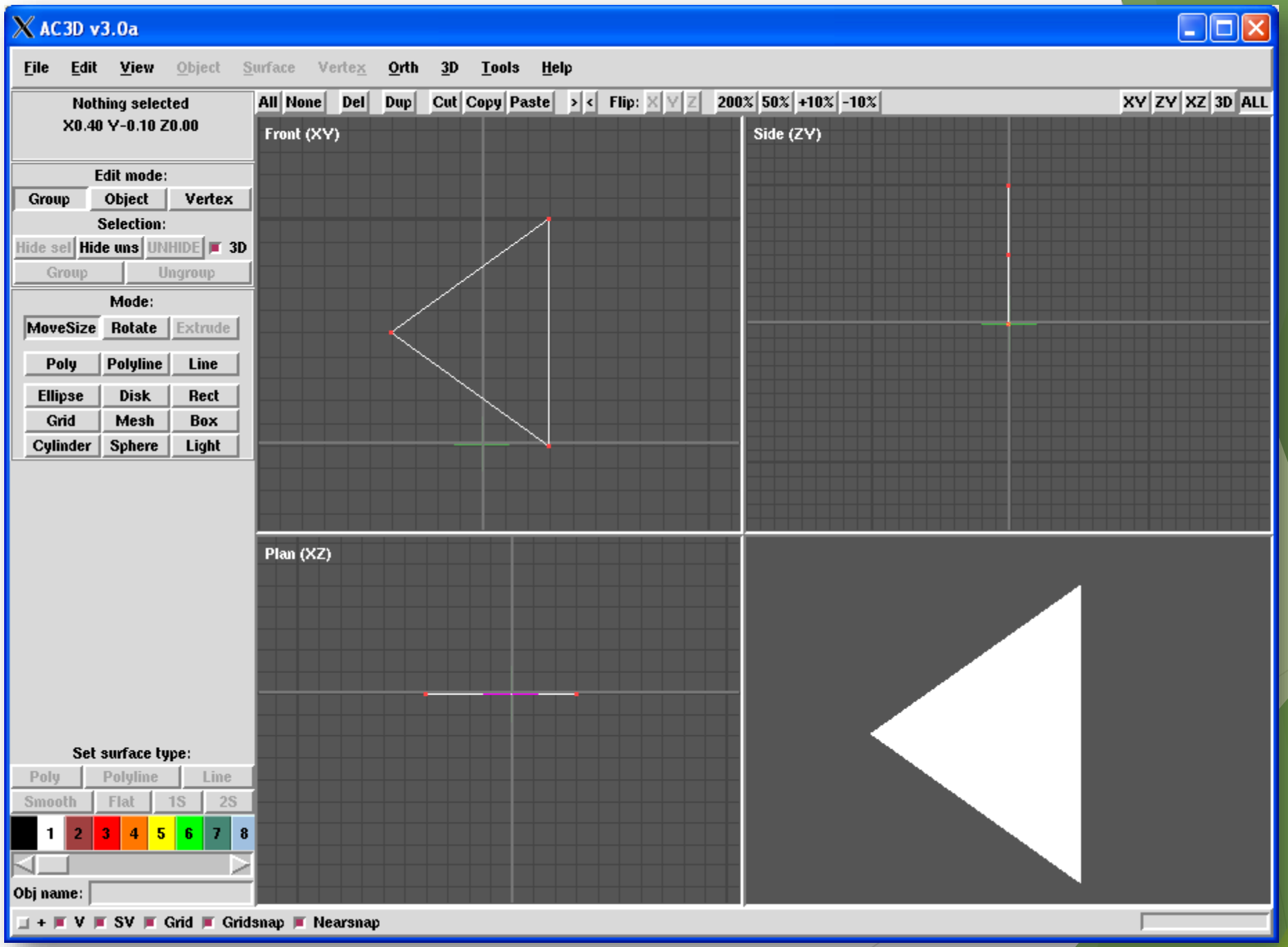


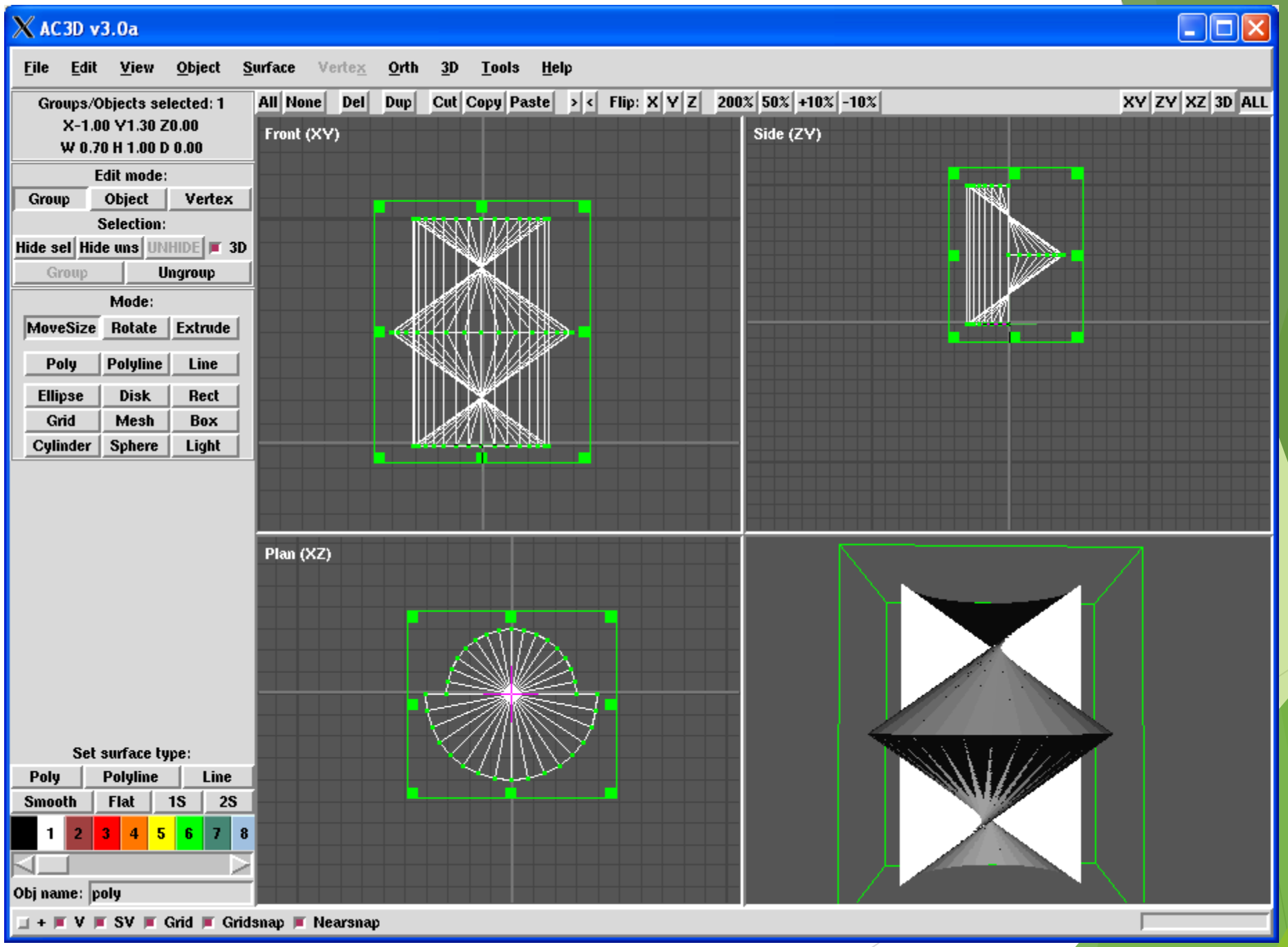
Source


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%
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
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

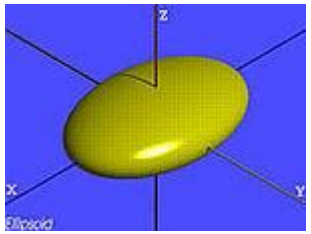

AC3D v3.0a
File Edit View Object Surface Vertex Orth 3D Tools Help
Groups/Objects selected: 1
X-1.00 Y1.30 Z0.00
W 0.70 H 1.00 D 0.00
Edit mode:
Group Object Vertex
Selection:
Hide sel Hide uns UNHIDE 3D
Group Ungroup
Mode:
MoveSize Rotate Extrude
Poly Polyline Line
Ellipse Disk Rect
Grid Mesh Box
Cylinder 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. Implicit Quadric surfaces

Voorwerp voorgesteld door middel van een kwadratische vergelijking

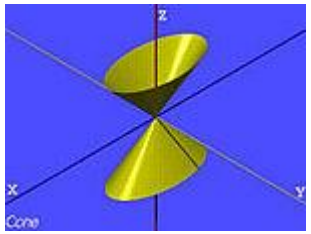
- cirkel geroteerd om zijn middelpunt: bol

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



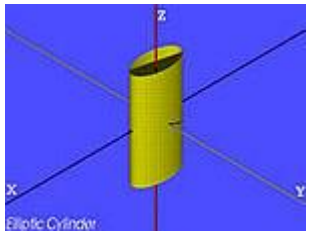
- lijn met een punt op de rotatie-as: kegel

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



- Lijn parallel aan de rotatie-as: cylinder

$$\frac{x^2}{a^2}+\frac{y^2}{b^2}=1$$ (circulair als a = b)



3. Implicit Quadric surfaces

Voordelen:

- Zeer nauwkeurig
- Zeer compact

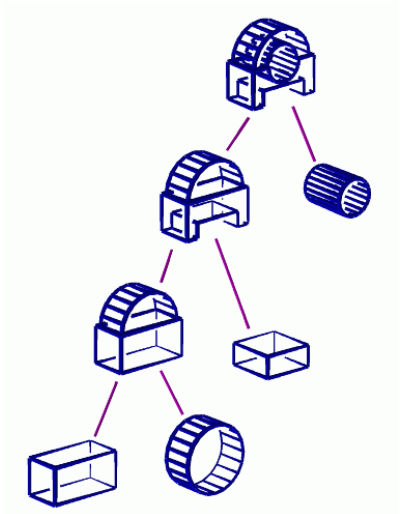
Nadelen:

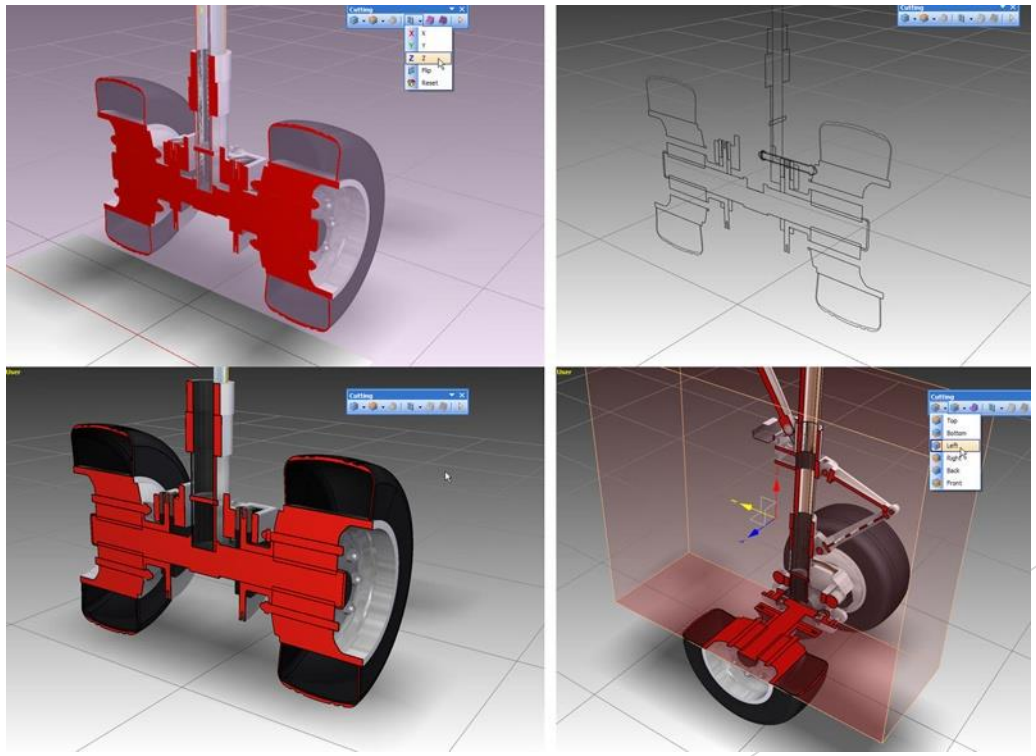
- Objecten impliciet gedefinieerd
- Gelimiteerd toepassingsgebied? (ray tracing)
- Lastig om onregelmatige objecten mee te modeleren

4. Constructieve methoden

Complexe objecten modeleren is door combineren van primitieven

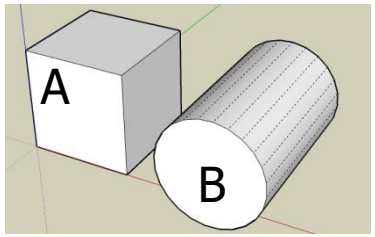
- Blokken, bollen, cylinders, kegels, wiggen, etc. (vaak quadrics)

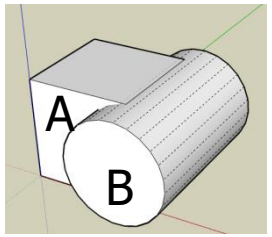




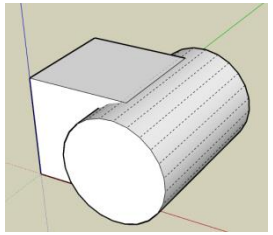
4. Constructieve methoden
Boolse verzameling operaties

Eenvoudige voorwerpen combineren tot complex voorwerp :

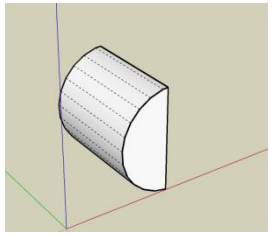




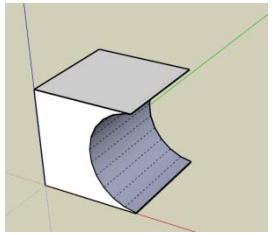
Verzameling operaties: vereniging, doorsnede en verschil:



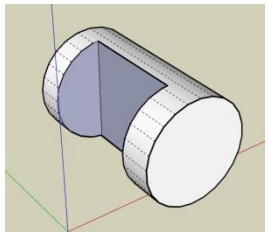
A ∪ B



A ∩ B



A - B

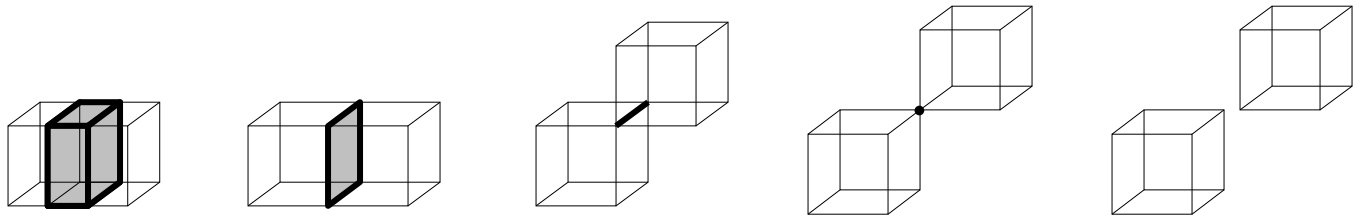


B - A

Metafoor: “aan elkaar plakken”, “uitsnijden”

4. Constructieve methoden
Boolse verzameling operaties

Dit levert niet altijd weer een voorwerp op!
Neem bv. de boolse doorsnede van twee kubussen:



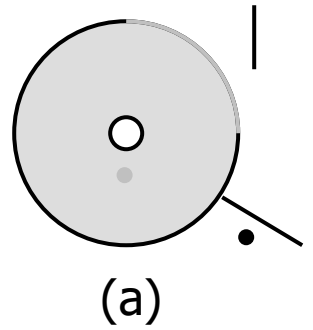
voorwerp vlak lijn punt leeg voorwerp

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

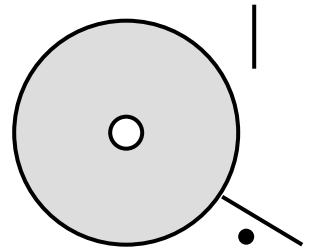
- Zodat bv. A $\cap^*$ B een voorwerp oplevert, of leeg is

4. Constructieve methoden
“Reguleren” van een voorwerp

- (a) is 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
- Van (a) is (b) het “gesloten” voorwerp (“closure”):
  - Alle randpunten van het voorwerp
  - Interne punten van het voorwerp
- Van (a) is (c) de “binnenkant” van het voorwerp (“interior”):
  - Alle bengelende en losse lijnen/punten verwijderd
- (d) is het gereguleerde voorwerp = gesloten binnenkant van voorwerp

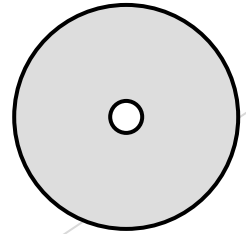


(a)



(b)

(c)



(d)

4. Constructieve methoden
“Reguleren” van een voorwerp

Een gereguleerd voorwerp:

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

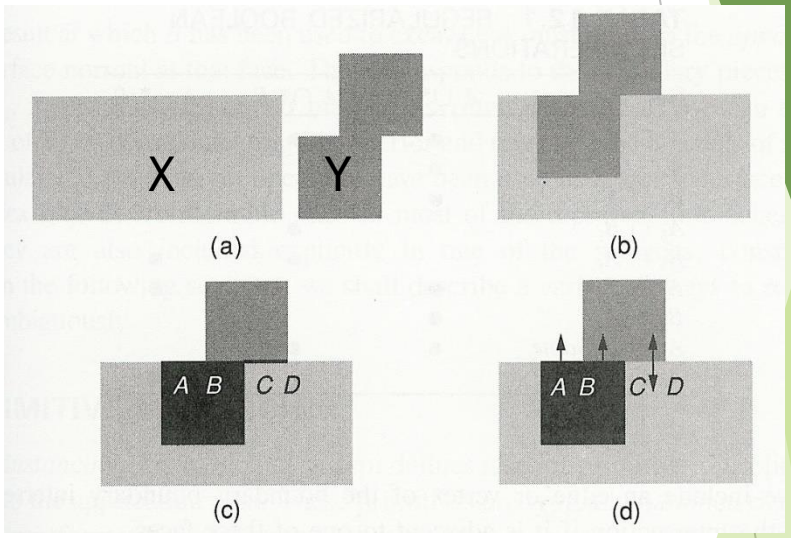
Regulerende boolse verzameling operatoren als gewone boolse operatoren:

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

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

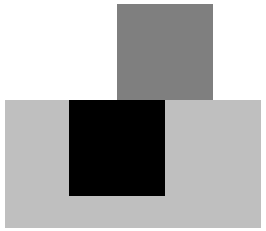
4. Constructieve methoden
Verschil gewone en regulerende boolse operaties

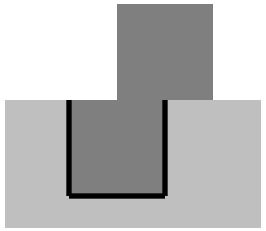
a) 2 voorwerpen X en Y
b) Positie voor ∩
c) X∩Y bevat AB, BC en CD
   - (X_i en Y_b) ∩ (Y_i en X_b)
d) X∩*Y bevat AB en BC maar niet CD
   - ($X_i \cap Y_i$, $Y_b \cap X_i$, $X_b \cap Y_i$ en een stukje van $X_b \cap Y_b$

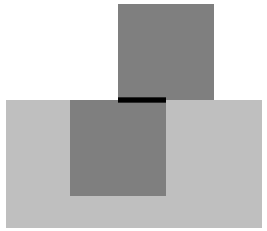


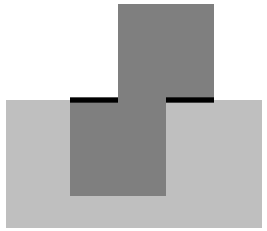
X_i : interieur van X
X_b : boundary van X

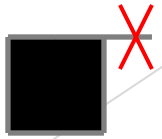
Source: Foley, van Dam et al. Computer Graphics: principles and practice, Addison-Wesley Professional, 1996.


$X_i \cap Y_i$


$Y_b \cap X_i$


$X_b \cap Y_i$


$X_b \cap Y_b$


$X \cap Y$


$X \cap^* Y$

4. Constructieve methoden
Verschil gewone en regulerende boolse operaties

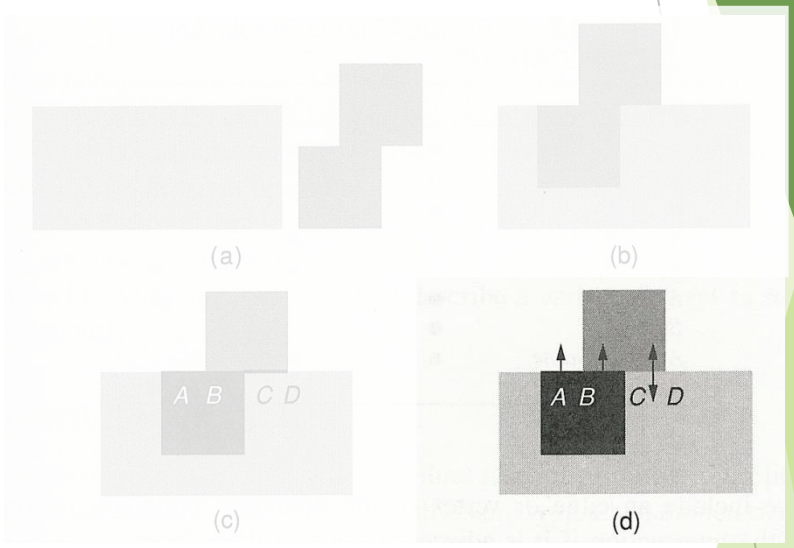
De rand zit wel in $\cap^*$ als:

- 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

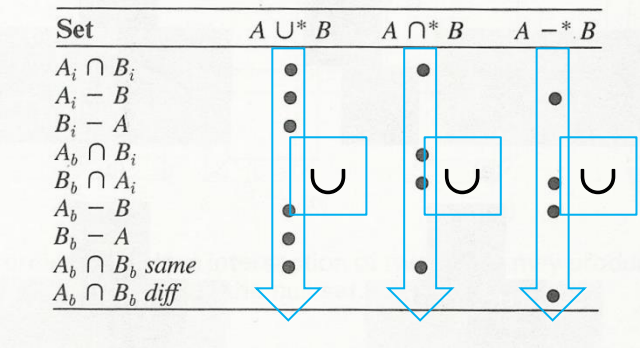
De restrictie op de randpunten zorgt voor een gereguleerd voorwerp.

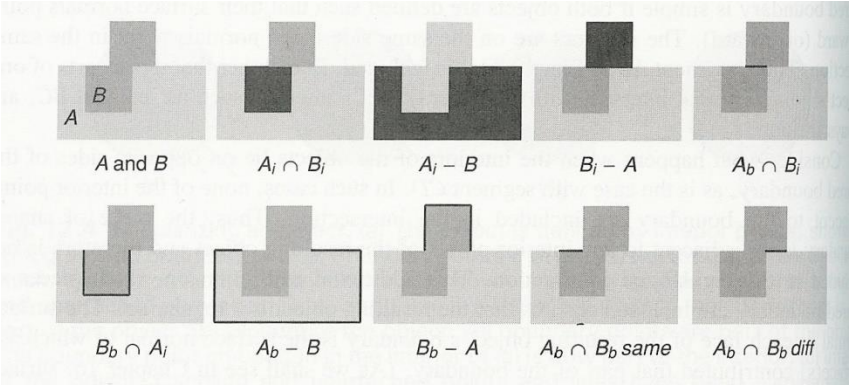


Source: Foley, van Dam et al. Computer Graphics: principles and practice, Addison-Wesley Professional, 1996.

4. Constructieve methoden
Verschil gewone en regulerende boolse operaties

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

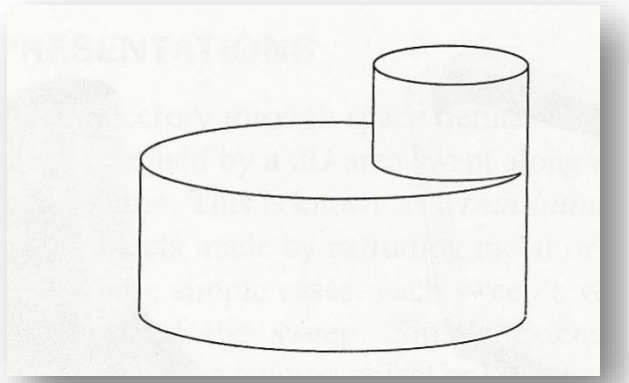




Source: Foley, van Dam et al. Computer Graphics: principles and practice, Addison-Wesley Professional, 1996.

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:



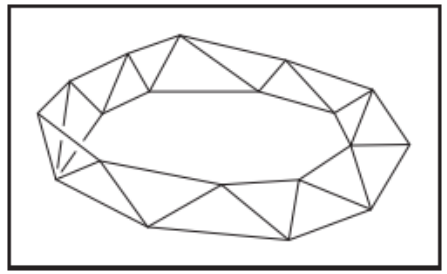


Figure 12.5. A triangulated Möbius band, which is not orientable.

Hoeveel zijden?

2-manifold

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

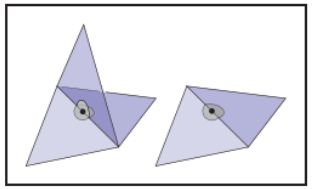


Figure 12.1. Non-manifold (left) and manifold (right) interior edges.

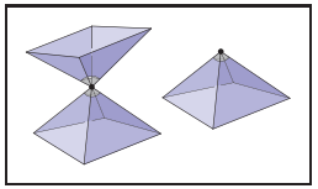
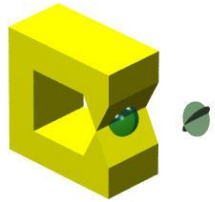
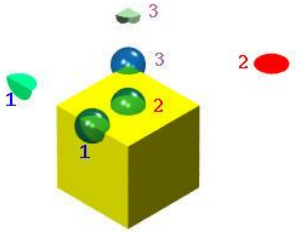


Figure 12.2. Non-manifold (left) and manifold (right) interior vertices.

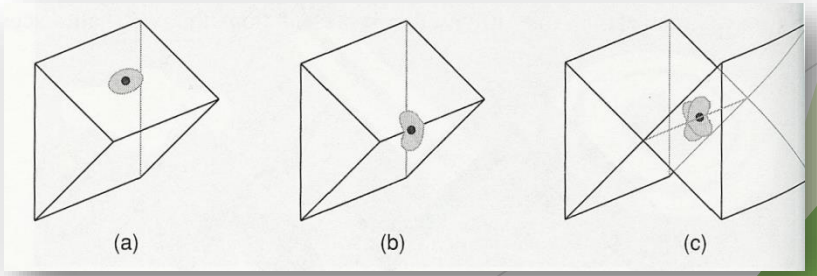


Niet 2-manifold



Wel 2-manifold

- 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)



2-manifold

2-manifold:

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

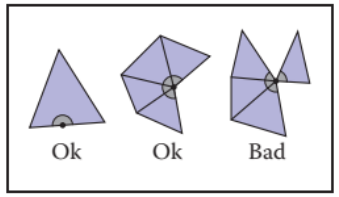
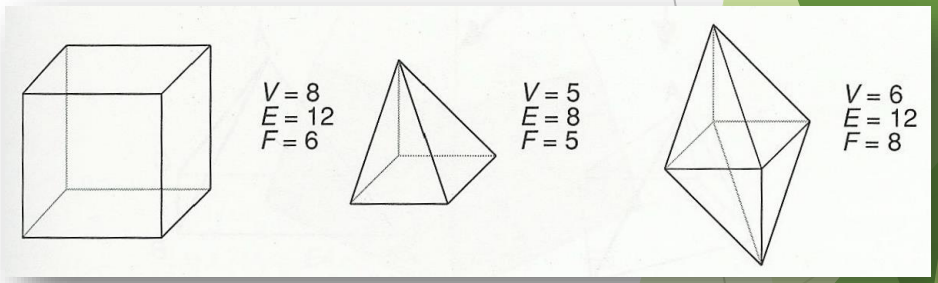


Figure 12.3. Conditions at the edge of a manifold with boundary.

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)

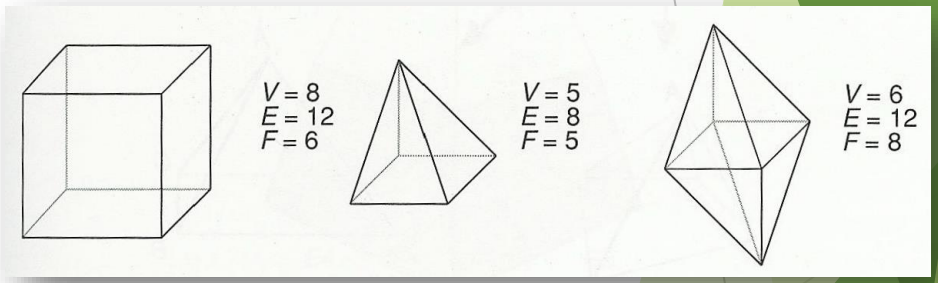
Een eenvoudig polyhedron (geen gaten) kan vervormd worden tot een bol



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 relatie:

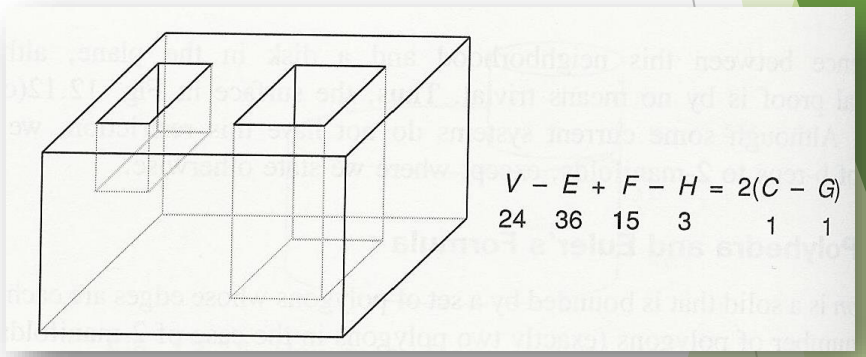
$V - E + F = 2$

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

B-rep van polyhedra

Generalisatie van Euler's formule voor 2-manifolds met zijden met gaten:

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



waar V (Vertices): aantal punten,
E (Edges): aantal randen,
F (Faces): aantal zijden
H (*Holes*): aantal gaten in zijden,
G (*Genus*): aantal gaten door het voorwerp,
C (*Components*): aantal aparte componenten in het voorwerp

Datastructuren

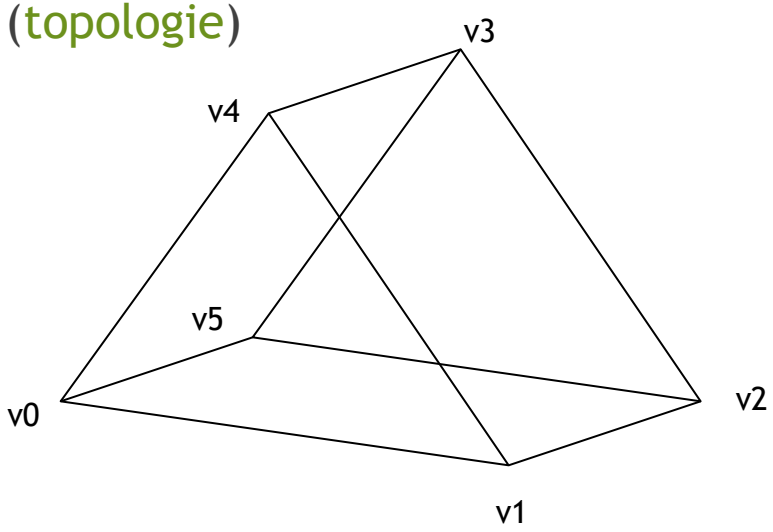
1. Polygoon meshes
2. Winged-edge
3. Spatial partitioning representations

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:

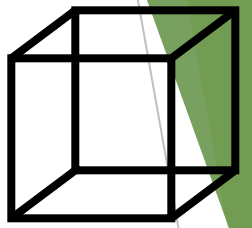
- Lijst van alle punten (geometrie)
- Lijst van alle zijden (topologie)



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.

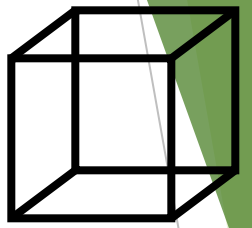
Kubus in OpenGL – versie 1



```
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

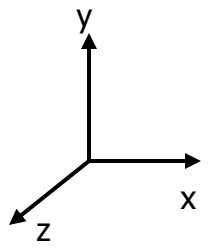
Kubus in OpenGL – versie 2

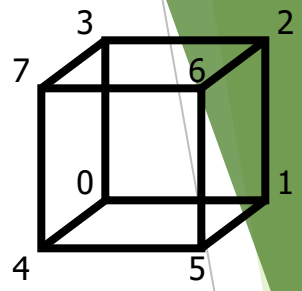


```
face()
{
   glBegin(GL_QUADS);
   glVertex3f(-1.0, -1.0f,  1.0f);
   glVertex3f( 1.0, -1.0f,  1.0f);
   glVertex3f( 1.0,  1.0f,  1.0f);
   glVertex3f(-1.0,  1.0f,  1.0f);
   glEnd();
}

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

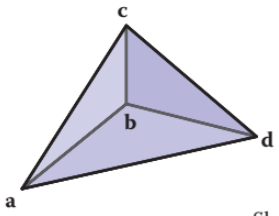
Kubus in OpenGL - versie 3






```
// 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



Separate triangles:

#	Vertex 0	Vertex 1	Vertex 2
0	(a_x, a_y, a_z)	(b_x, b_y, b_z)	(c_x, c_y, c_z)
1	(b_x, b_y, b_z)	(d_x, d_y, d_z)	(c_x, c_y, c_z)
2	(a_x, a_y, a_z)	(d_x, d_y, d_z)	(b_x, b_y, b_z)

Shared vertices:

Triangles	
#	Vertices
0	(0, 1, 2)
1	(1, 3, 2)
2	(0, 3, 1)

Vertices	
#	Position
0	(a_x, a_y, a_z)
1	(b_x, b_y, b_z)
2	(c_x, c_y, c_z)
3	(d_x, d_y, d_z)

Figure 12.6. A three-triangle mesh with four vertices, represented with separate triangles (left) and with shared vertices (right).

2. Winged-edge

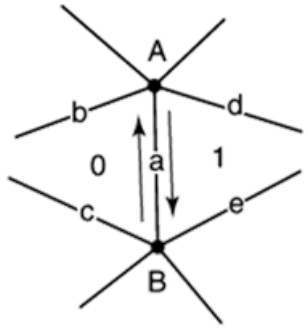
Voor toepassingen waar de polygoon mesh niet verandert is een eenvoudige datastructuur voldoende.

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)



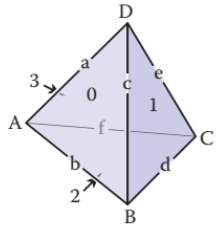
edge	vertex 1	vertex 2	face left	face right	pred left	succ left	pred right	succ right
a	B	A	0	1	c	b	d	e

De winged edge data structuur

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	e	b	d

Vertex	*Edge*
A	a
B	d
C	d
D	e

Face	*Edge*
0	a
1	c
2	d
3	a

Figure 12.15. A tetrahedron and the associated elements for a winged-edge data structure. The two small tables are not unique; each vertex and face stores any one of the edges with which it is associated.

3. Spatial-partitioning representaties

Voorwerp wordt gerepresenteerd door aaneengrenzende,

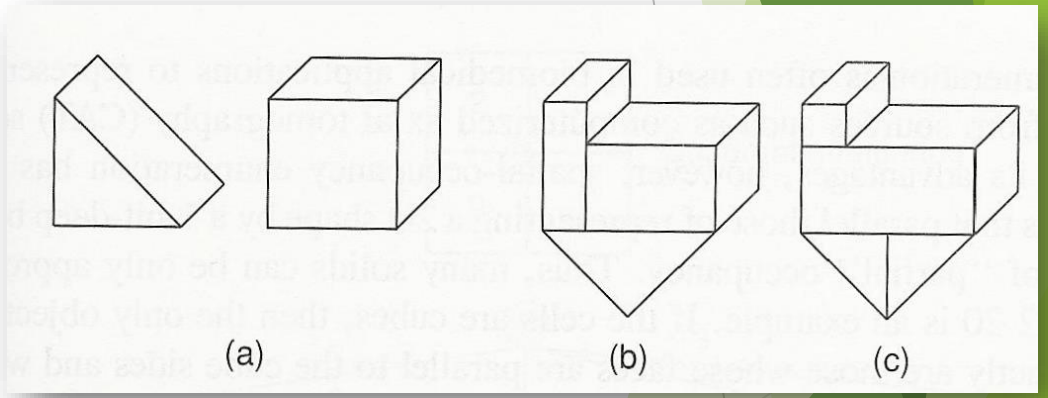
niet-overlappende volumes

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 (zie figuur)



Onder andere gebruikt in eindige elementen methode (Finite Element/Volume Methods, FEM/FVM)

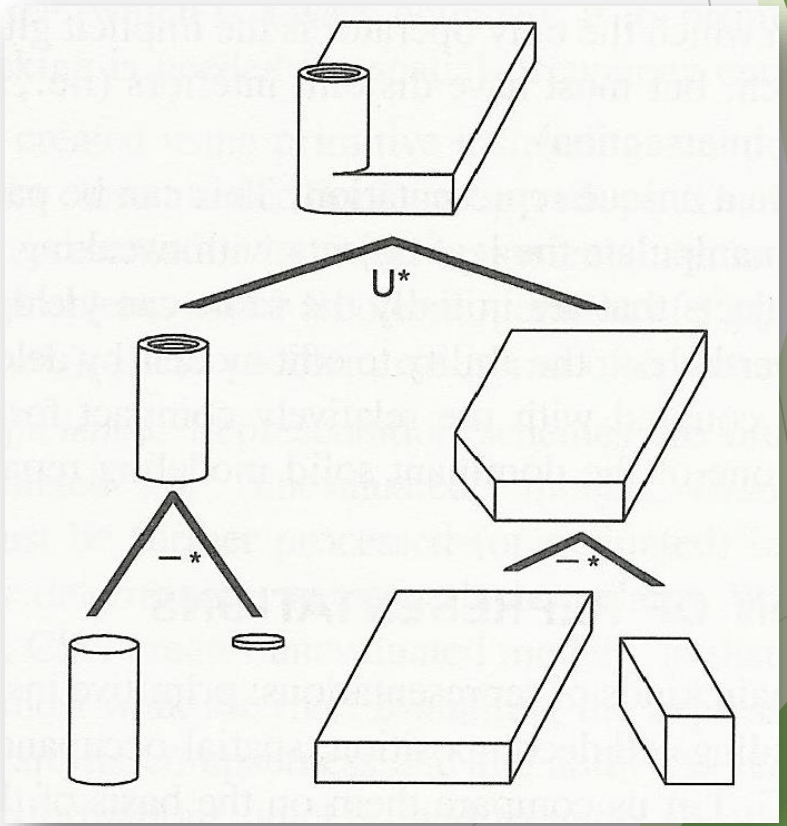
3.1. Cell decomposition

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
- Set operaties: $\cup$, $\cap$, $-$
  - elke bewerking levert een regulier voorwerp op
- Half-spaces; niet begrensde volumes
  - Bv: kubus is 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

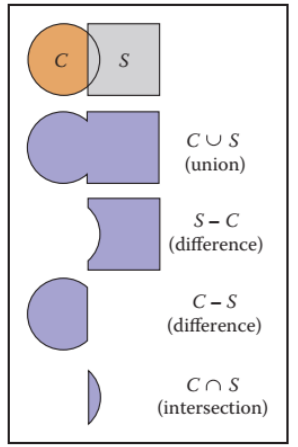
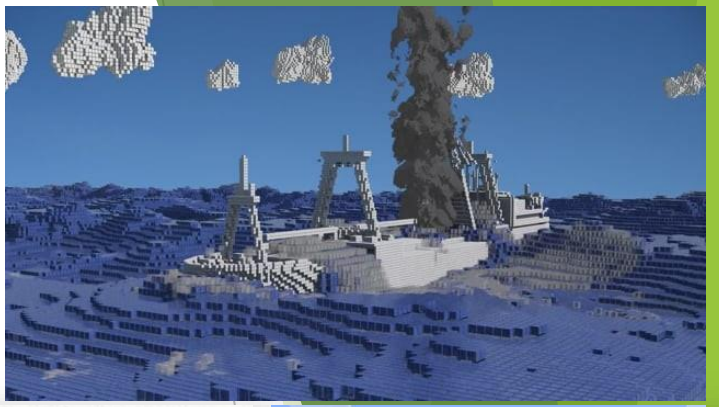


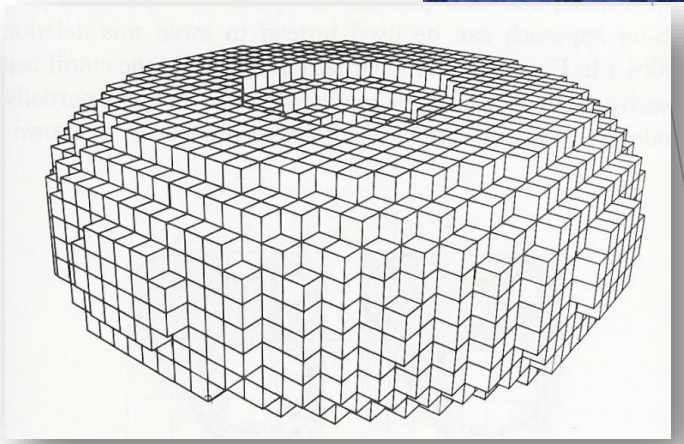
Figure 13.6. The basic CSG operations on a 2D circle and square.

3.2. Spatial-occupancy enumeration



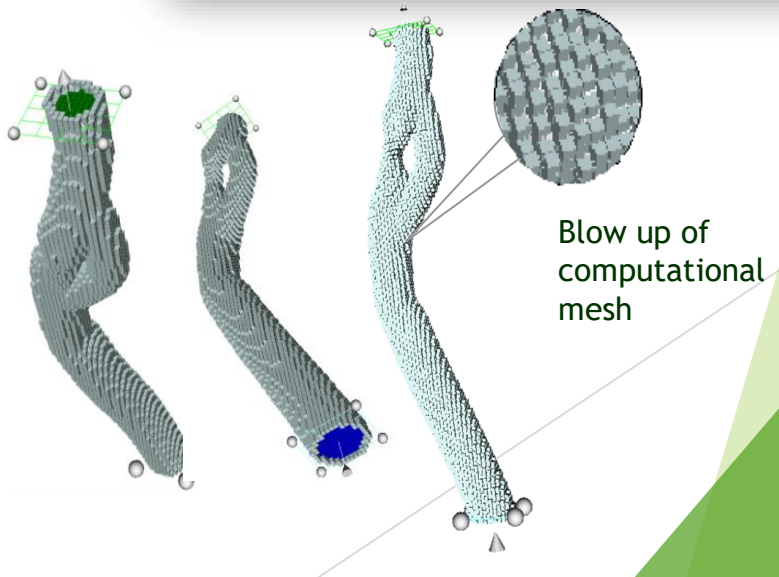
Cuberilles: cellen worden weergegeven door identieke cellen: blokjes (“voxels”) in een regelmatig rooster

- Alleen aangeven of cel wel of niet zichtbaar moet zijn
- Nadeel: nauwkeurigheid afhankelijk van cel grootte
- De representatie is ondubbelzinnig
- De representatie is uniek!



Wordt soms gebruikt in biomedische toepassingen

- CT/MRI, lattice Boltzman



Voxel based simulation used in Houdini (Source) (see also explanation of Volumes in Houdini)

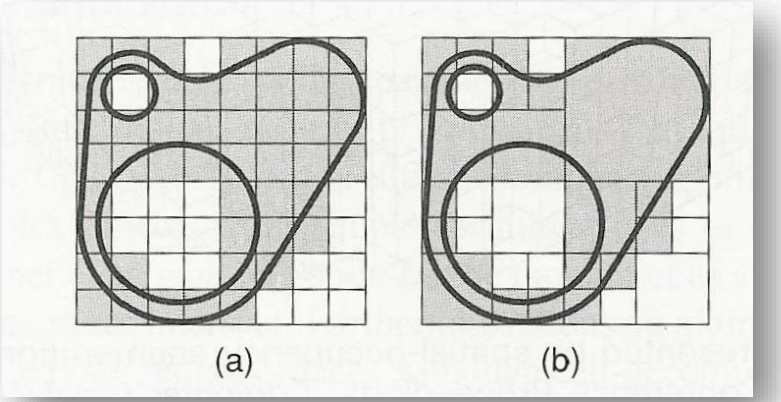
3.3. Quadtrees/Octrees

Hierarchische datastructuur

- 2D: quadtrees
- 3D: octrees

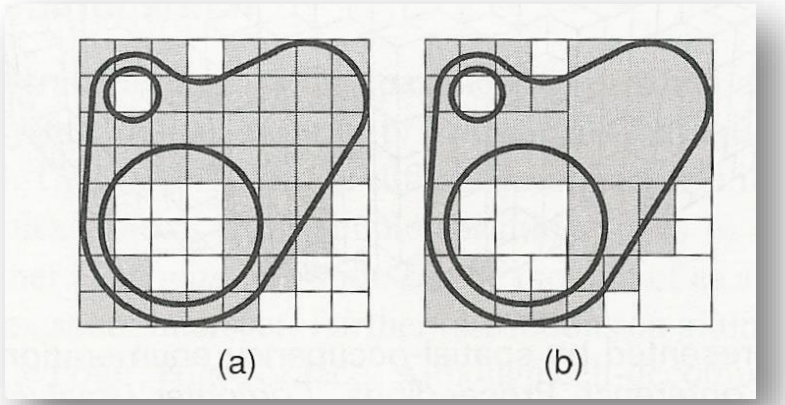
2D quadtree ter illustratie:

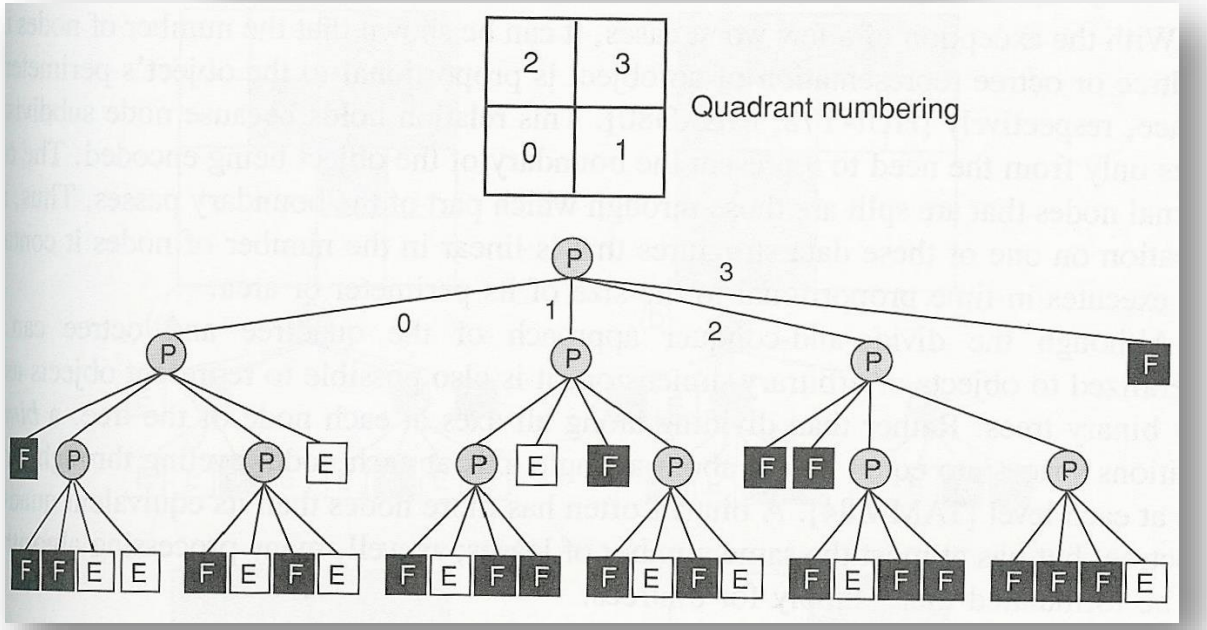
- Spatial occupancy enumeration



- 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





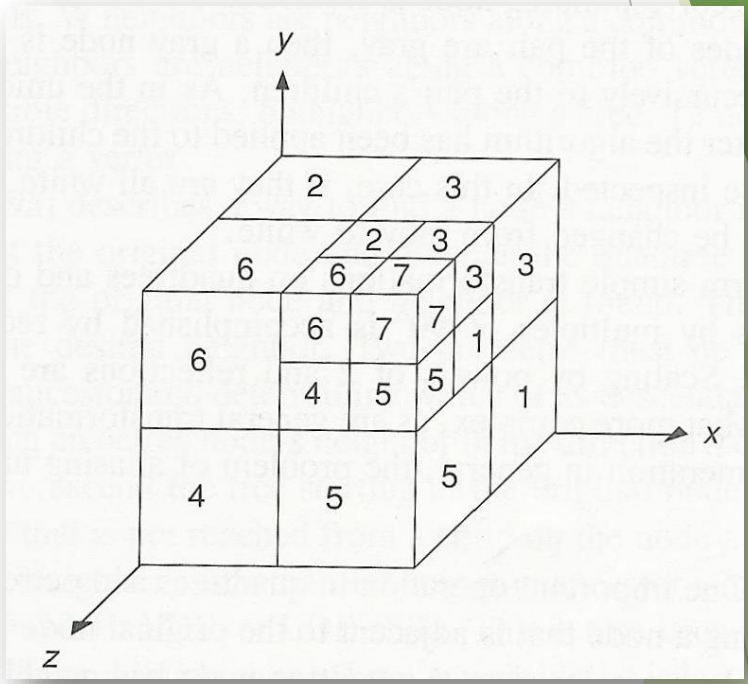
3.3. Quadtrees/Octrees

Octree:

- Blok onderverdeeld in octanten
- Octant 0 is hier niet zichtbaar

Efficiënt op te slaan en te verwerken

Boolse verzameling operatoren zijn eenvoudig toe te passen



3.3. Quadtrees/Octrees

Octree:

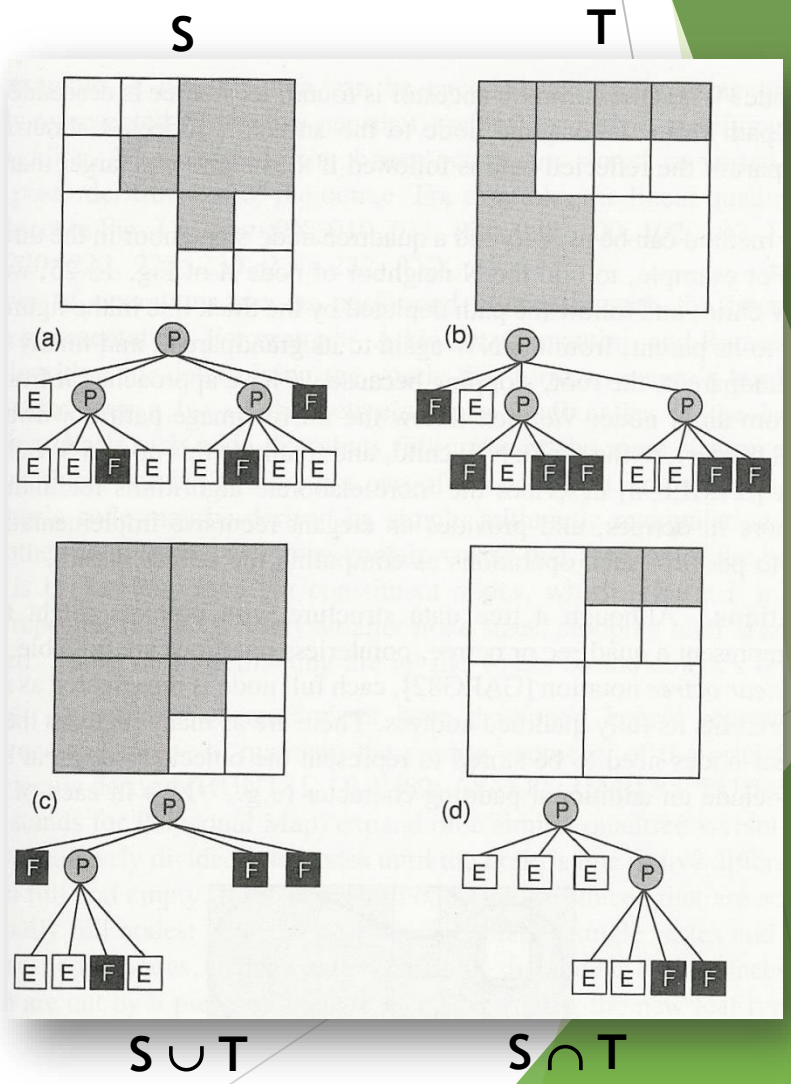
- Blok onderverdeeld in octanten
- Octant 0 is hier niet zichtbaar

Efficiënt 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



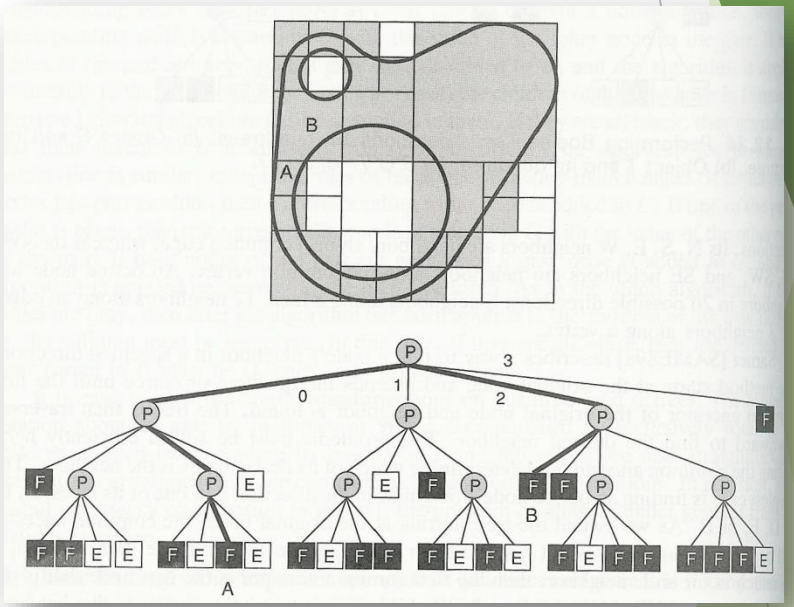
3.3. Quadtrees/Octrees

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 hiërarchisch (recursief) op in twee delen aan weerszijden van een vlak

- 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 aaneengrenzende, niet overlappende, convexe “in” gebieden
  - “Tesselatie”

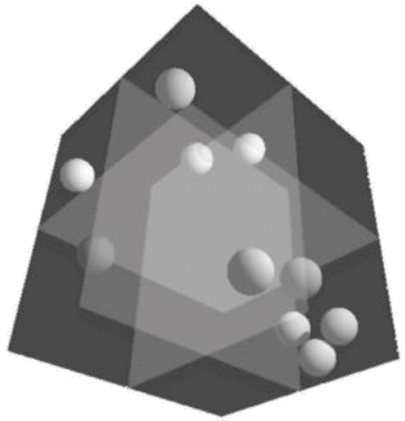
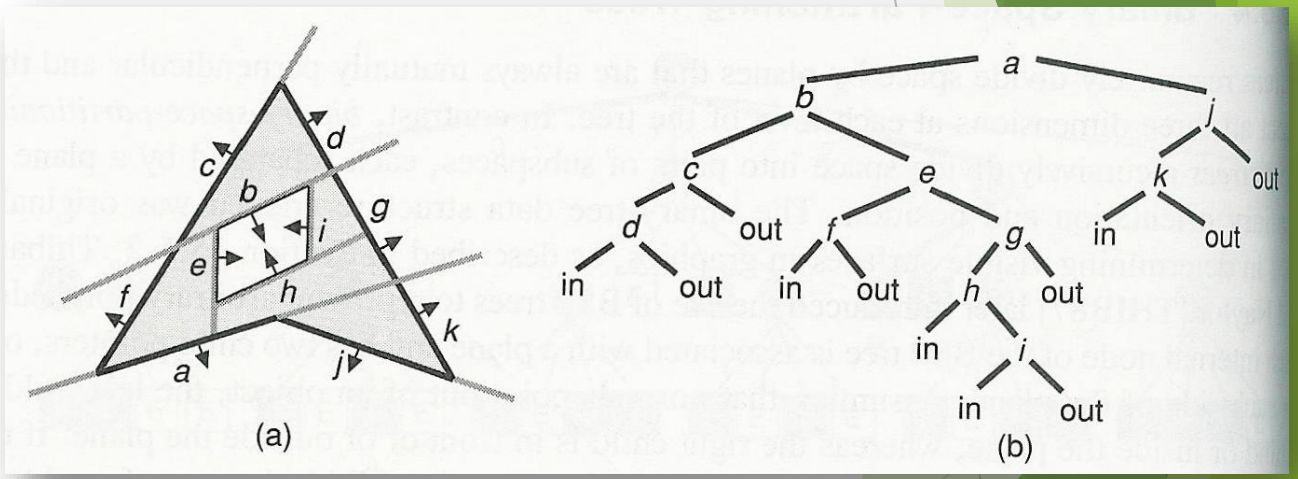


Figure 12.22. Left: a uniform partitioning of space. *Image courtesy David DeMarle.*

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:

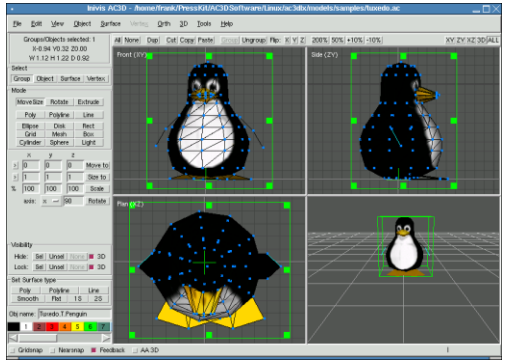
- Maya, Cinema4D, Lightwave, 3D Studio Max, Fusion 360, Houdini, ...

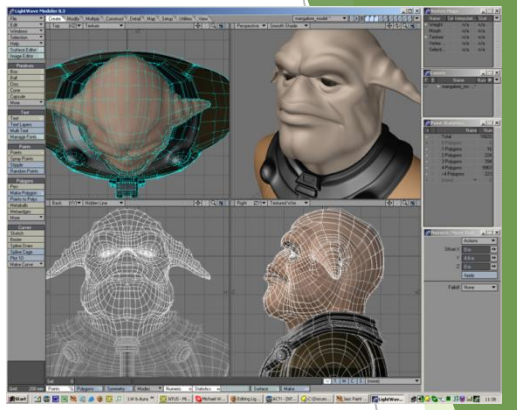
Cheap:

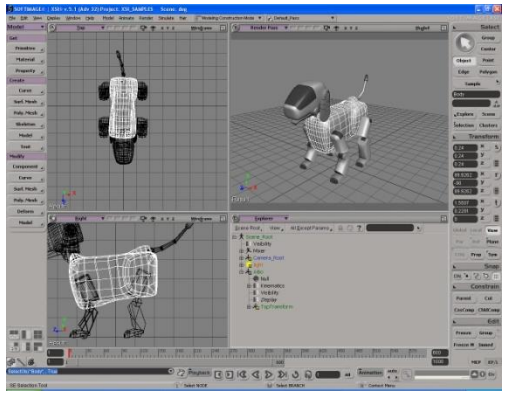
- AC3D, SketchUp, ...

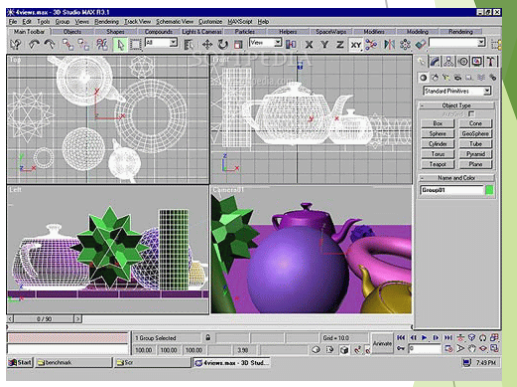
Free:

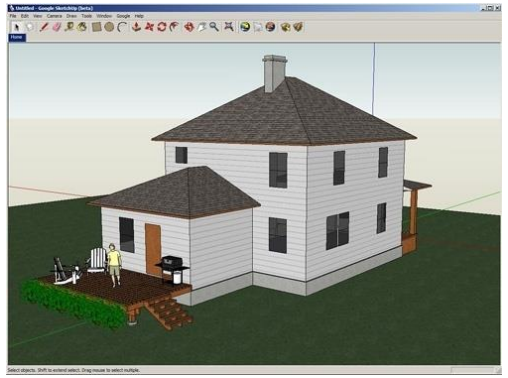
- Blender, Wings3D, Tinkercad, FreeCAD, ...

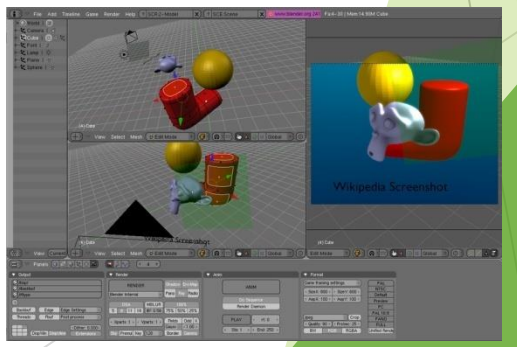










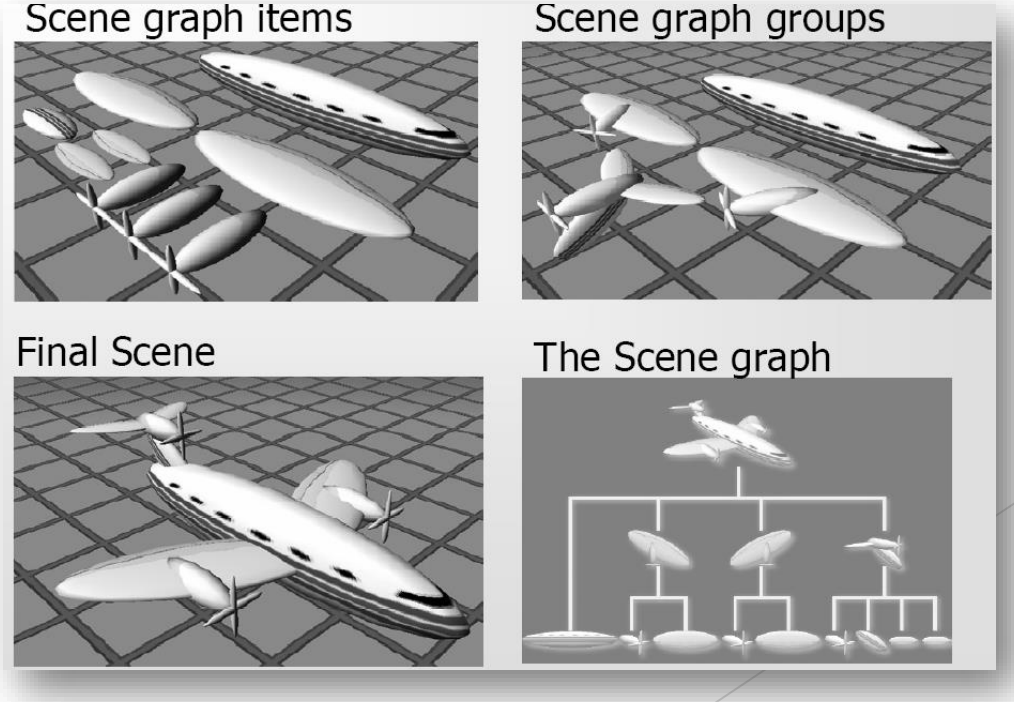


Scenegraphs

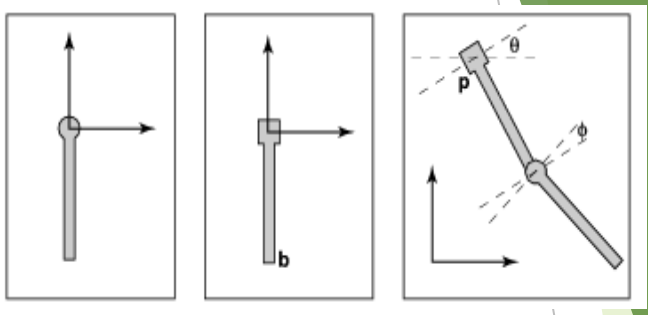
Werelden opgebouwd uit objecten

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

- Voorbeelden:
  - VRML
  - X3D
  - Java3D
  - Unity, Unreal



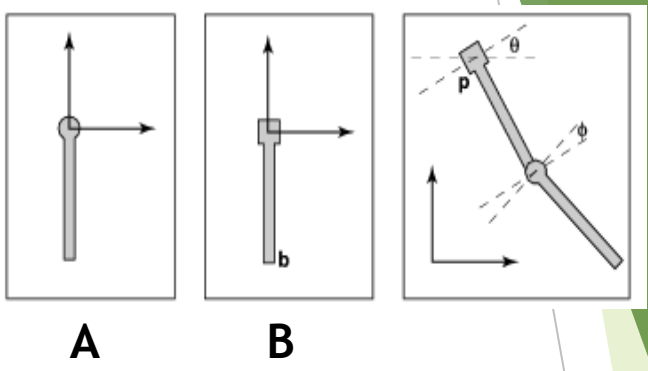
Scenegraphs



Voorbeeld: een slinger met twee scharnieren.

Hoe tekenen we die?

Scenegraphs



A B

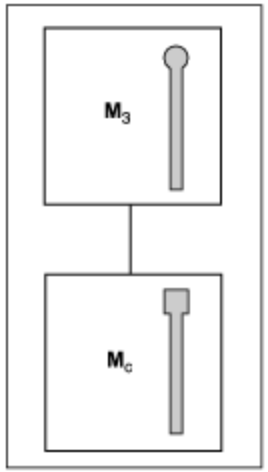
Het bovenste deel (B) kunnen we tekenen door:

$M_1 = \text{rotate}(\theta)$

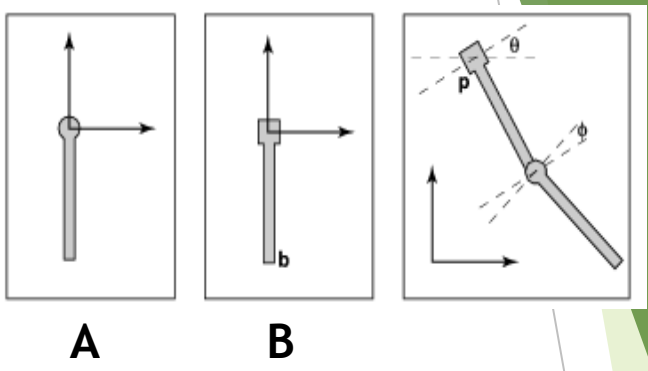
$M_2 = \text{translate}(p)$

$M_3 = M_2 M_1$

Pas M_3 toe op alle punten van het bovenste deel

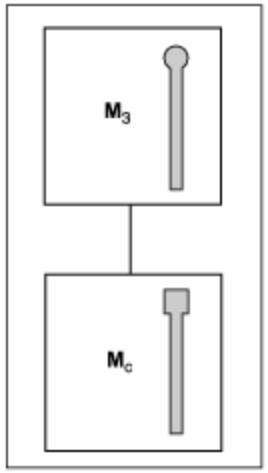


Scenegraphs



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

1. $M_a = \text{rotate}(\varphi)$
2. $M_b = \text{translate}(\boldsymbol{b})$
3. $M_c = MbMa$
4. $M_d = M_3 M_c$



Pas M_d toe op alle punten van het onderste deel

Scenegraphs

Ander voorbeeld:

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

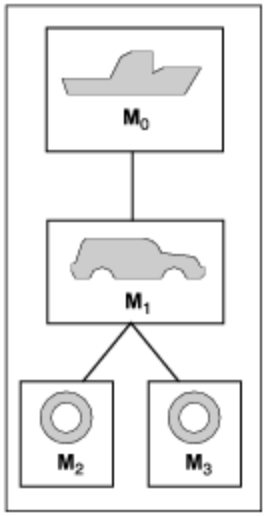
schip M_0

carosserie $M_0\ M_1$

wiel linksvoor $M_0\ M_1\ M_2$

wiel linksachter $M_0\ M_1\ M_3$

…



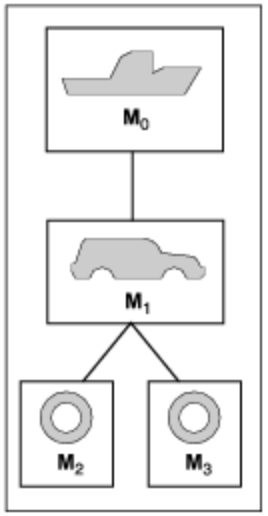
- Scenegraph is een Directed Acyclic Graph (DAG)
- Transformaties zijn cumulatief

  Alle voorwerpen onder een transformatie worden beïnvloed

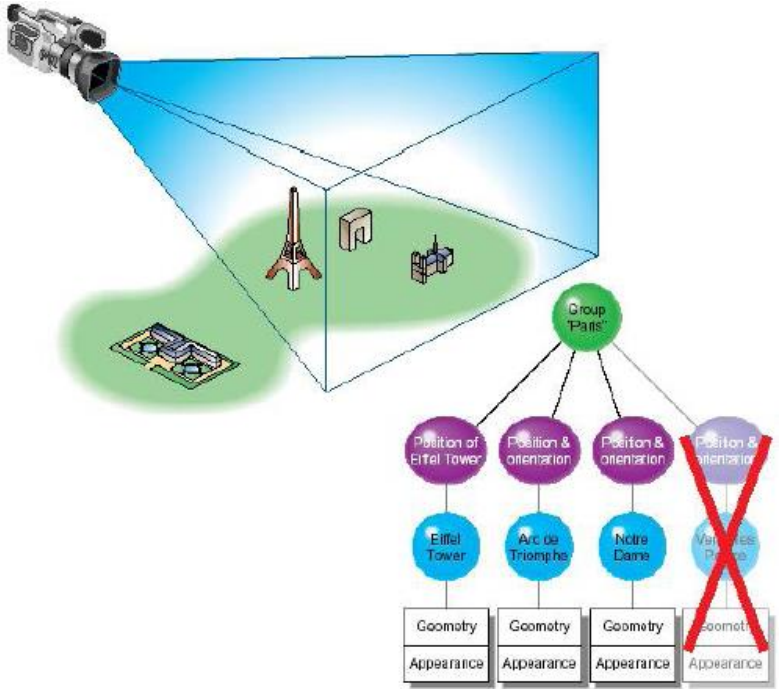
Scenegraphs

De matrix stack: aan de rechterkant van het matrixproduct matrices toevoegen (push) of te verwijderen (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