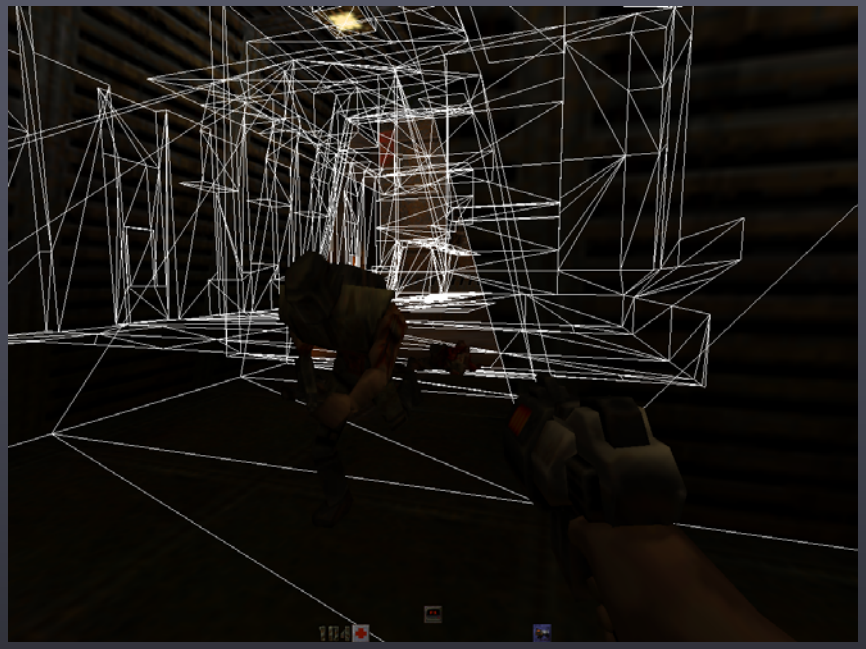
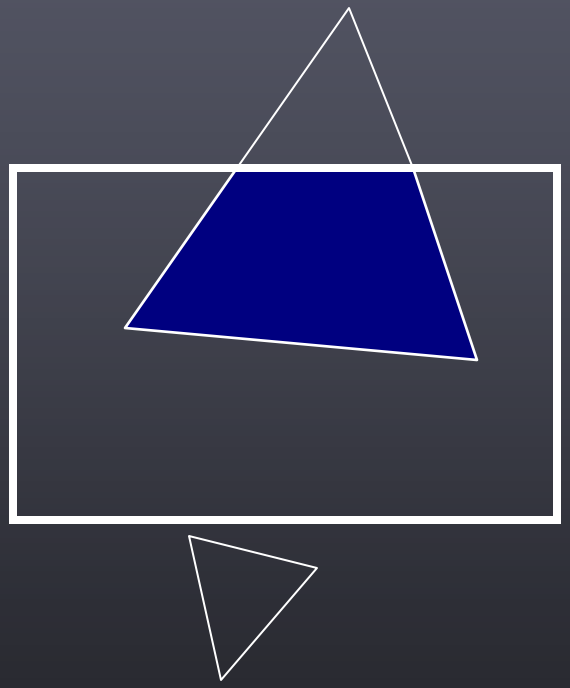


Rastrering och displayalgoritmer

Gustav Taxén
gustavt@csc.kth.se

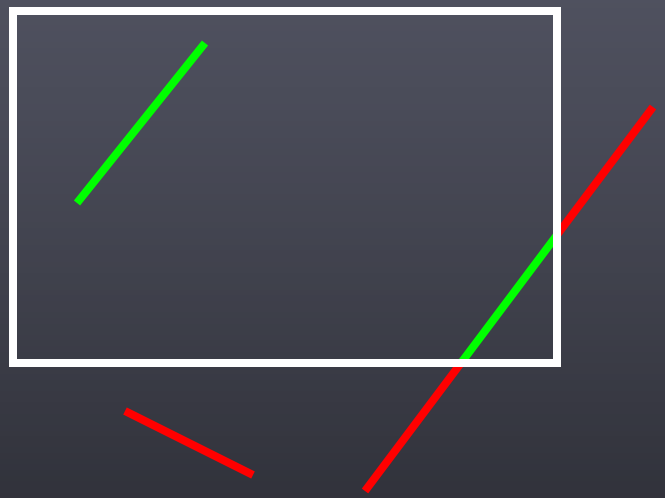
2D1640 Grafik och Interaktionsprogrammering VT 2007

Klippning



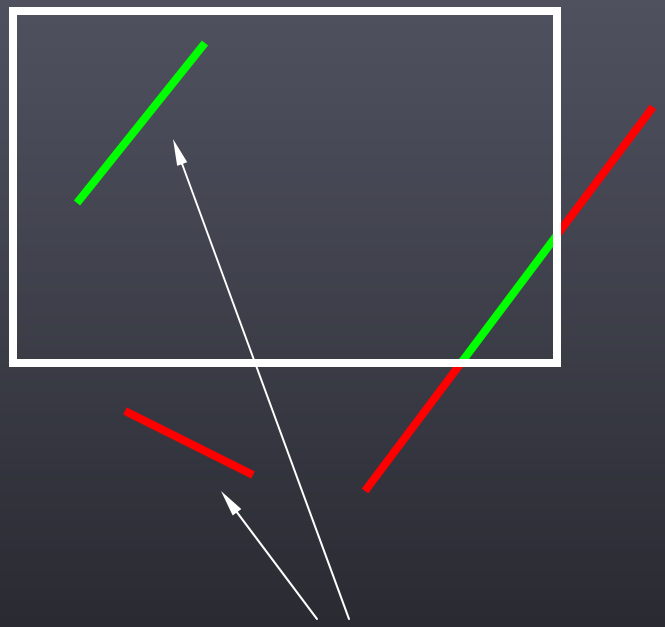
Man vill undvika att rastrera de primitiver som hamnar utanför fönstret. Man vill också undvika att rastrera delar av primitiver som hamnar utanför fönstret.

Klippning i 2D: Linje mot fönster



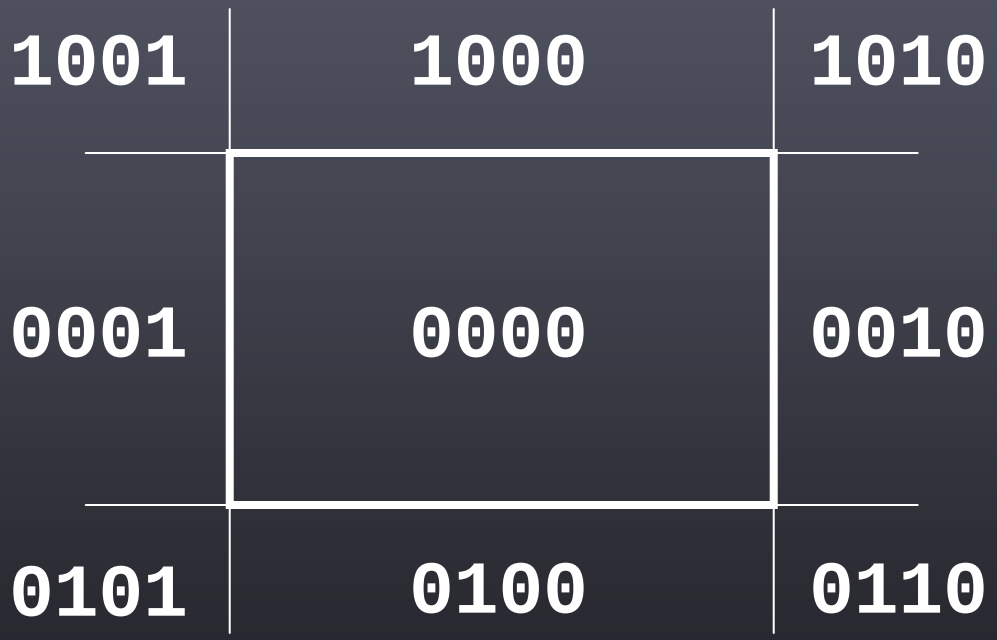
Målet är att så tidigt som möjligt avgöra om en linje ligger helt innanför eller helt utanför fönstret.

Klippning i 2D: Cohen-Sutherland



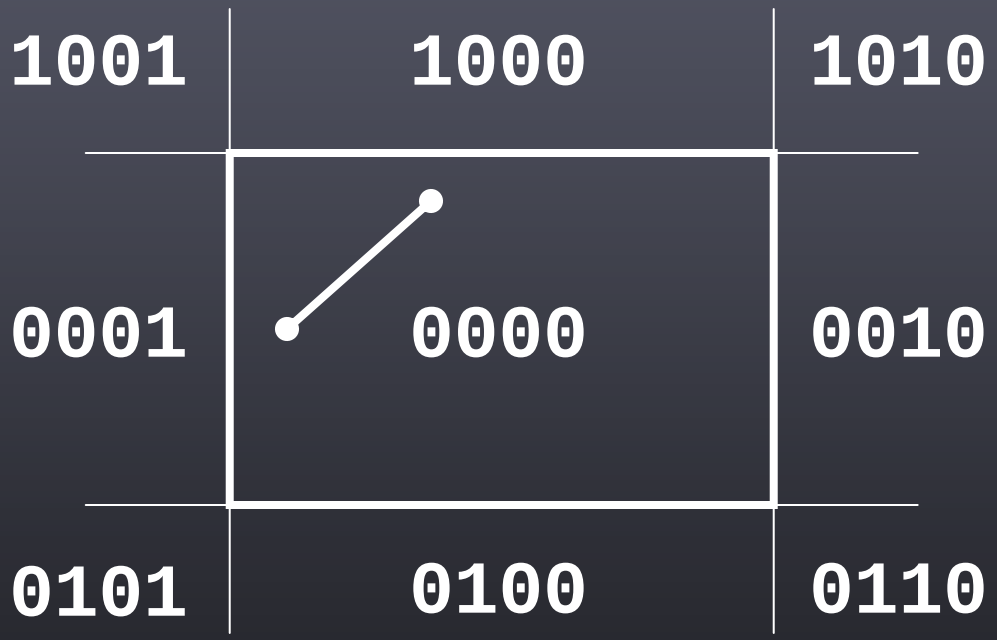
Det vore bra om vi slapp beräkna korsningspunkterna mellan linje och fönsterram där det inte behövs!

Klippning i 2D: Cohen-Sutherland



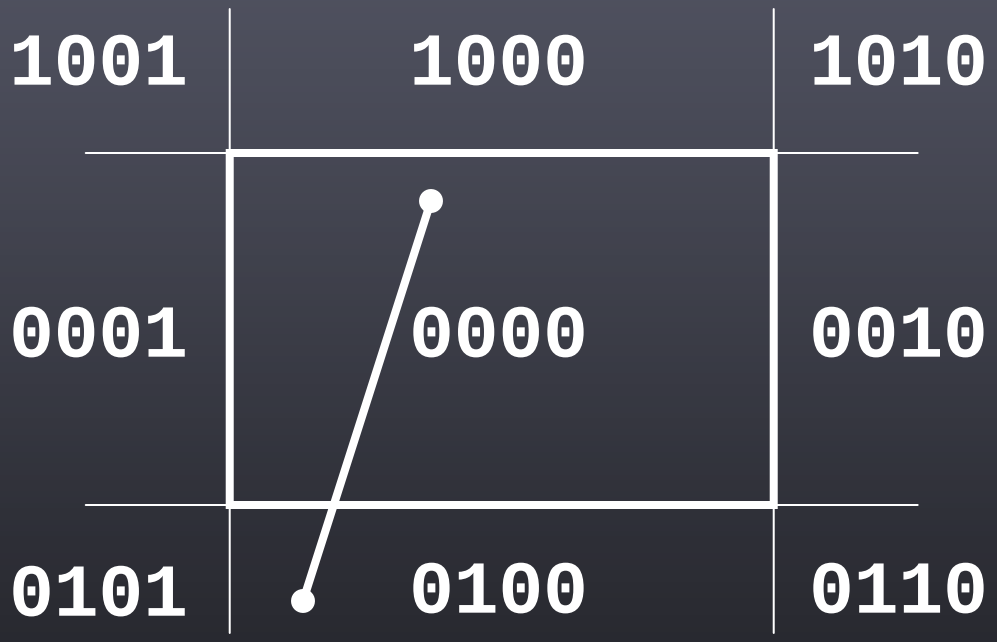
Dela planet i 9 delar. Tilldela en särskild bytekod till varje del. Kontrollera sedan vilka bytekoder linjens slutpunkter har.

Klippning i 2D: Cohen-Sutherland



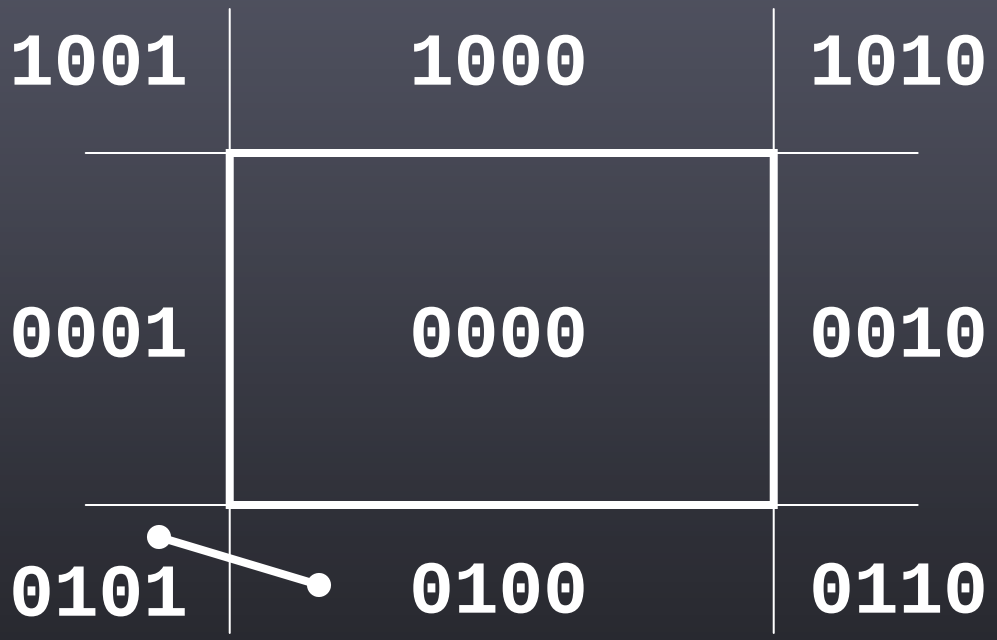
Båda har kod **0000**:
Linjen ligger i fönstret.

Klippning i 2D: Cohen-Sutherland



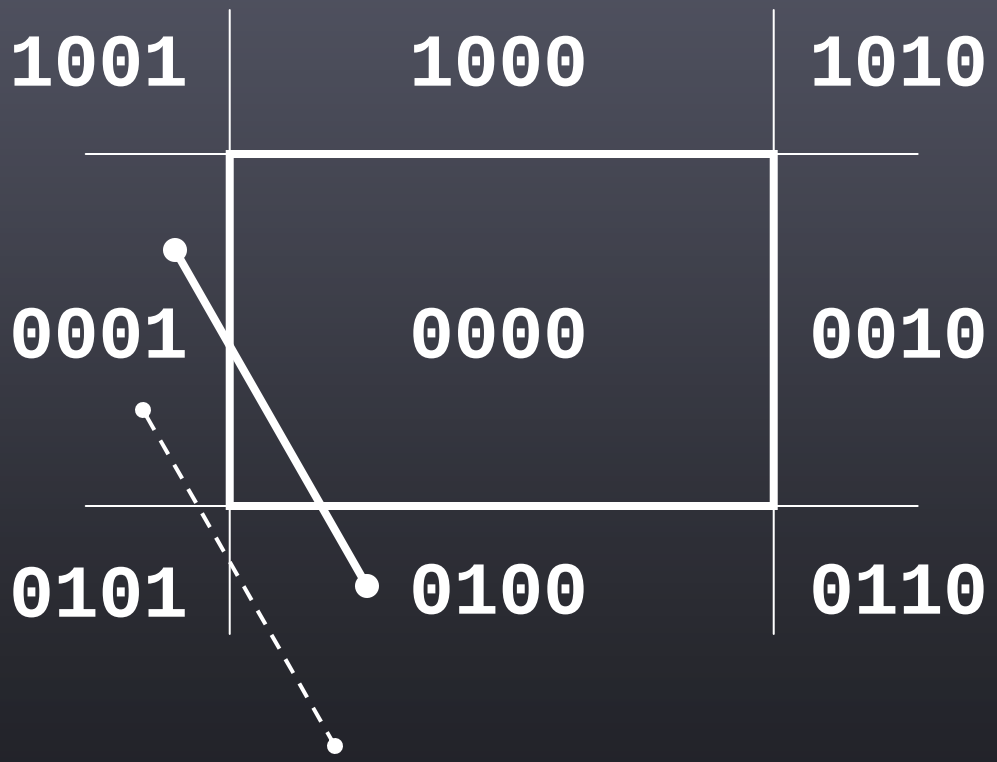
En har kod **0000**,
den andra inte.
Linjen måste klippas
mot fönstret.

Klippning i 2D: Cohen-Sutherland



0101 AND 0100 ≠ 0000
Linjen syns inte i fönstret.

Klippning i 2D: Cohen-Sutherland



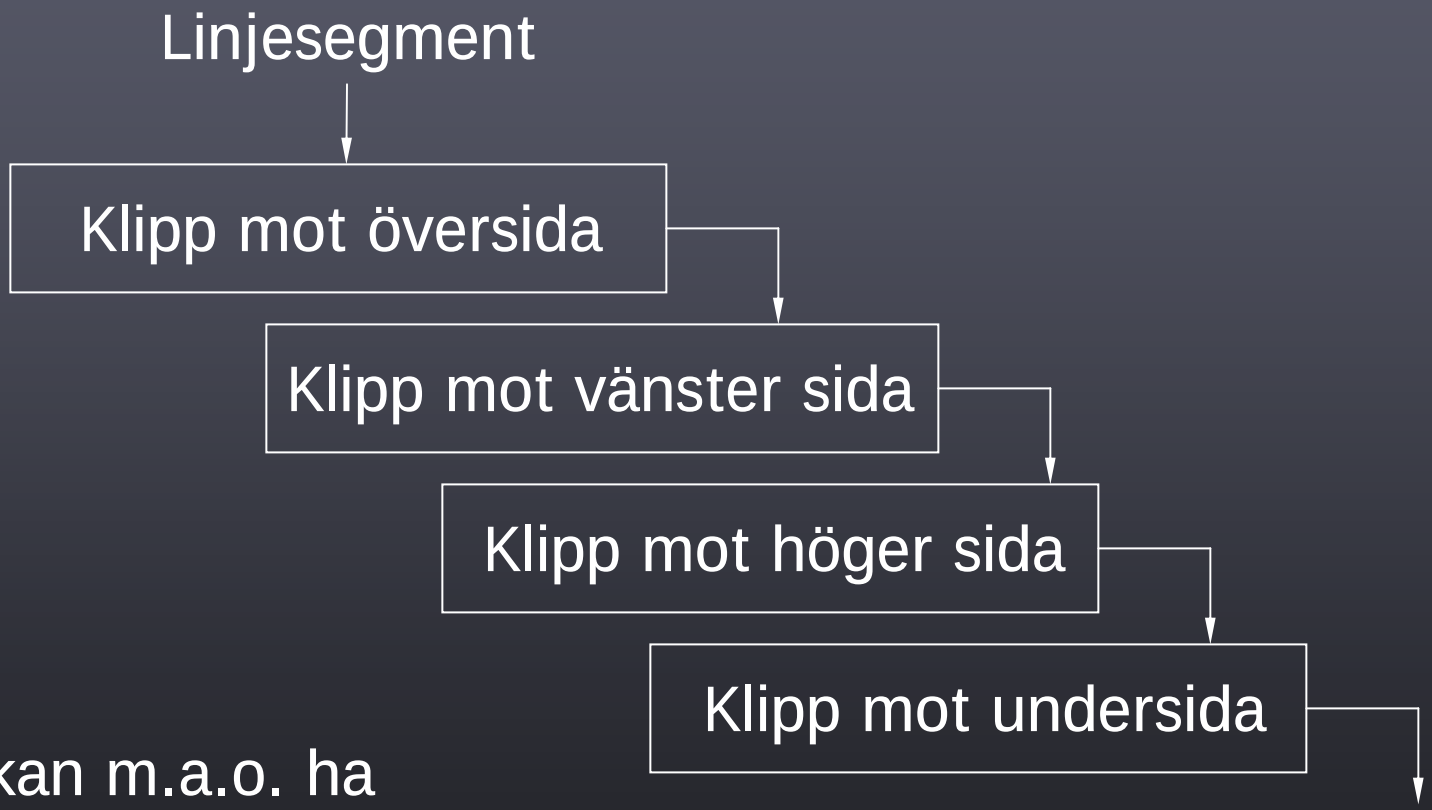
0001 AND 0100 = 0000

Linjen kanske behöver klippas mot fönstret.

Klippning i 2D: Sutherland-Hodgeman

Klipp mot fönstrets kanter var och en för sig i steg.

Klippning i 2D: Sutherland-Hodgeman

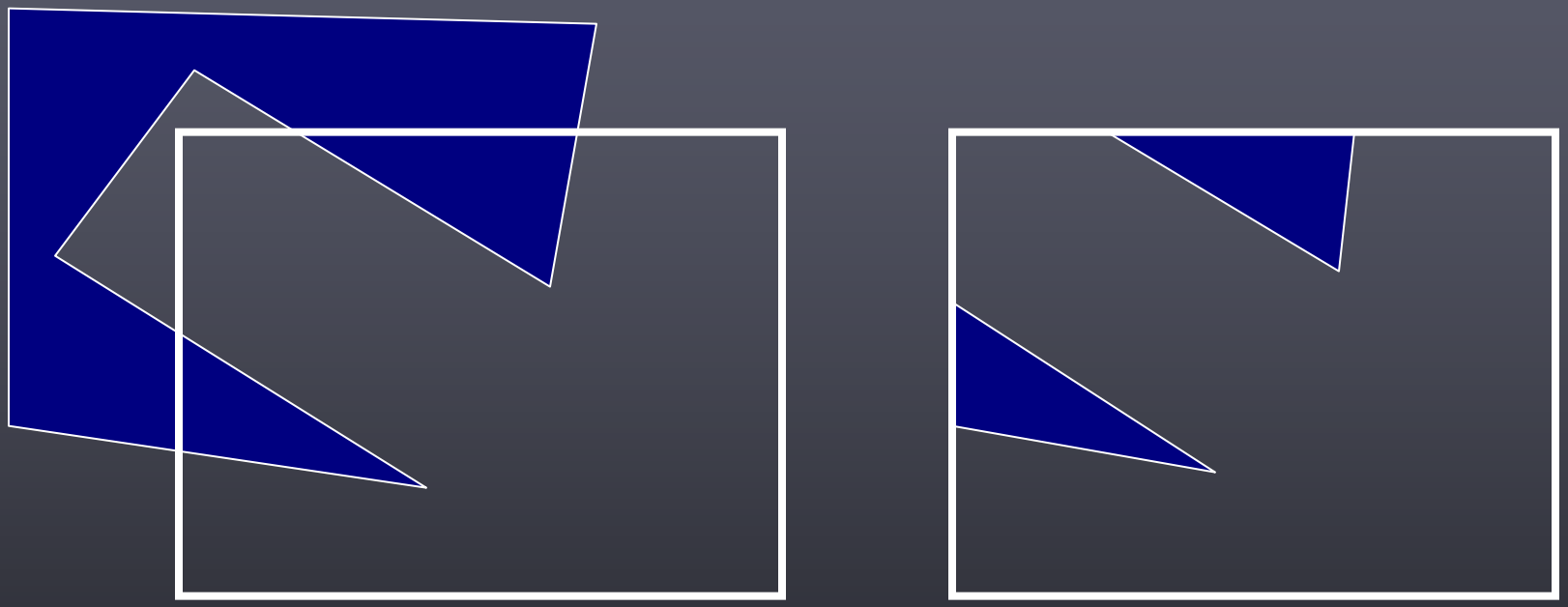


Vi kan m.a.o. ha
4 klippningar
pågående samtidigt.

Klippning i 2D

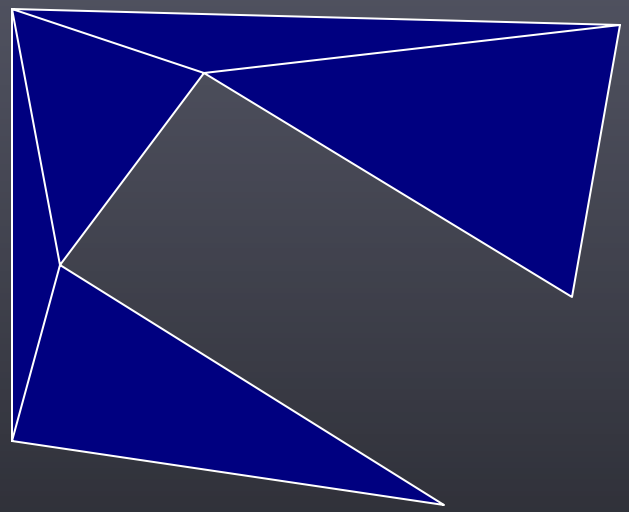
- Fler algoritmer i boken!

Konkava polygoner = problem!



Konvex: för två punkter i polygonen gäller att linjen mellan dem ligger helt inuti polygonen.
Konkav: alla andra polygoner.

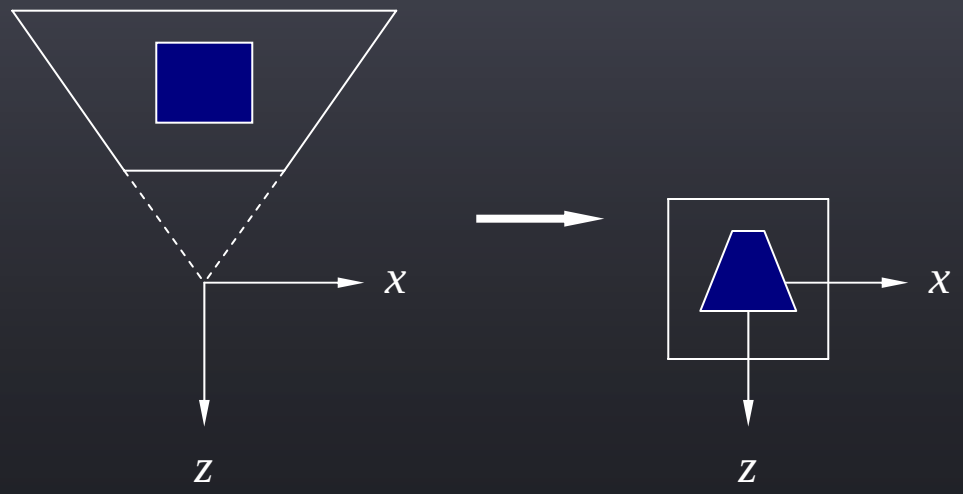
Uppdelening i konvexa polygoner (tesselation)



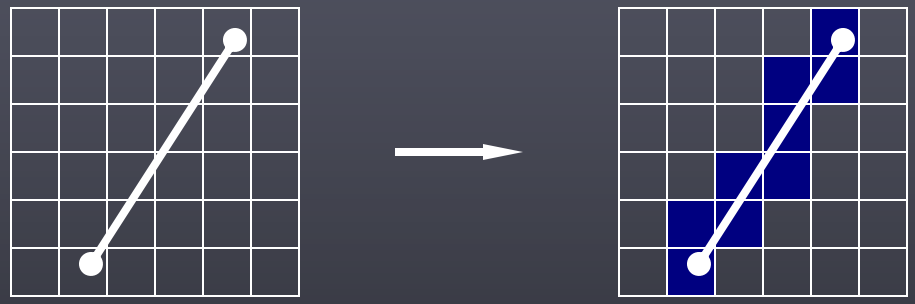
Klippning i 3D

Man vill att klippning ska ske mot en kub,
eftersom man då trivialt kan utöka föregående
klippalgoritmer till 3D.

Det är därför OpenGLs perspektivmatris ser ut som den gör!

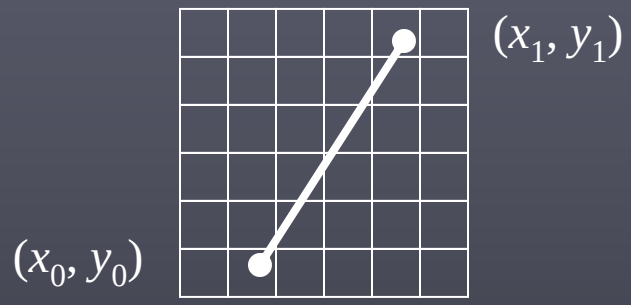


Rastrering



Diskretisering av en matematisk linje eller polygon.

Rastrering av linjer: Digital Differential Analyzer (DDA)

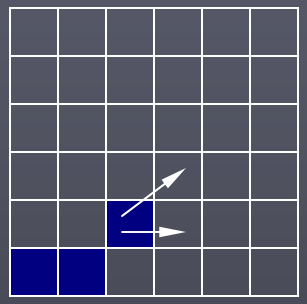


Låt linjens lutning vara $m = (y_1 - y_0) / (x_1 - x_0)$ och antag att $0 < m < 1$.

Sätt $x = x_0$ och $y = y_0$.
För varje $x = x + 1$, låt $y = y + m$.
Avrunda och rita ut pixel.

Övriga m löses analogt.

Rastrering av linjer: Bresenham



Liknar DDA men utan flyttalsoperationer.

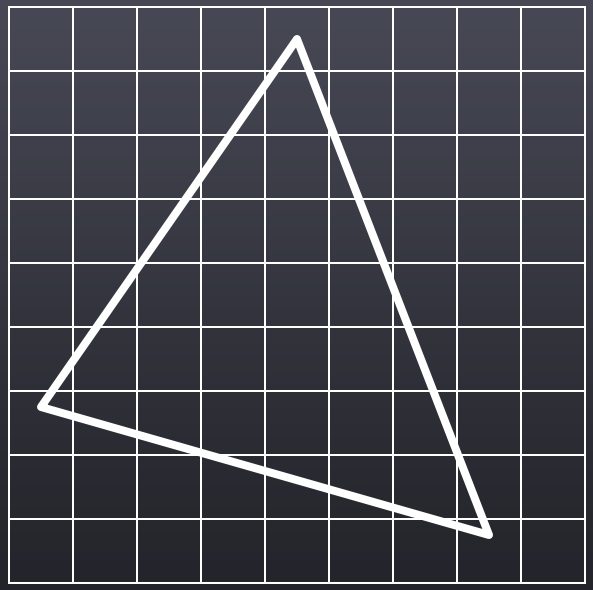
Går ut på att, givet att vi står i en pixel, ta reda på om vi ska flytta ett steg till höger eller ett steg upp + ett till höger.

Analogt för andra lutningar.

http://en.wikipedia.org/wiki/Bresenham's_line_algorithm

Rastrering av polygoner

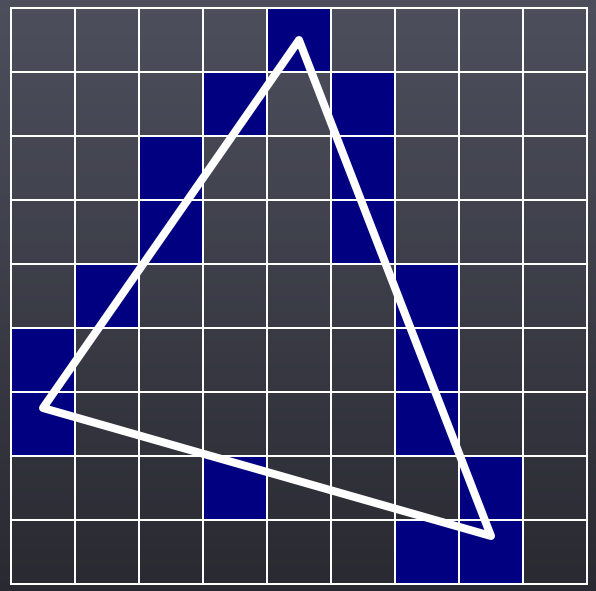
Trianglar är alltid konvexa! - Enklare att rastrera!
Så dela först alla polygoner i trianglar.
Rastrera sedan varje triangel för sig.



Det vore smidigt om vi kunde fylla triangeln rad för rad.

Vi kan då också enkelt interpolera värden över triangeln.

Rastrering av polygoner

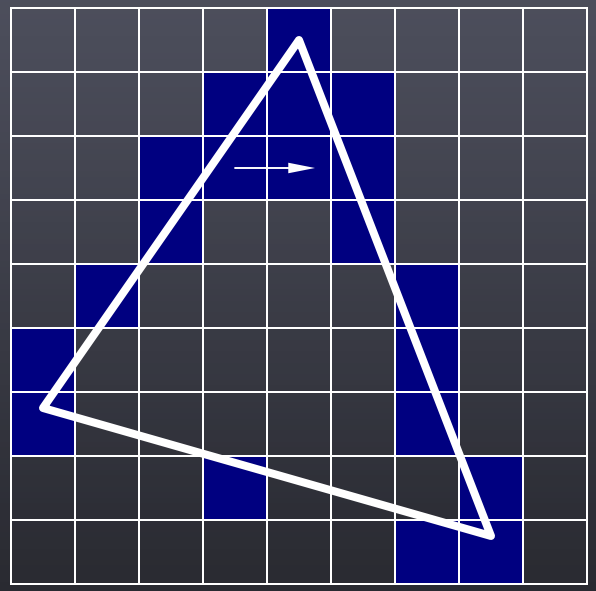


Använd DDA eller Bresenham för att hitta de x-värden som triangeln korsar för varje rad.

Lagra samtidigt interpolerade värden längs med kanterna.

Sortera m.a.p. x-värden.

Rastrering av polygoner



För varje rad som påverkas,
fyll och interpolera samtidigt
värden från vänster till höger.

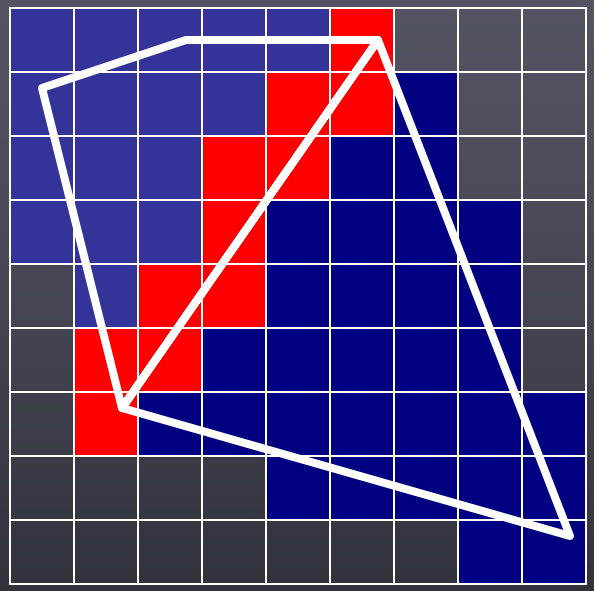
Rastrering av polygoner

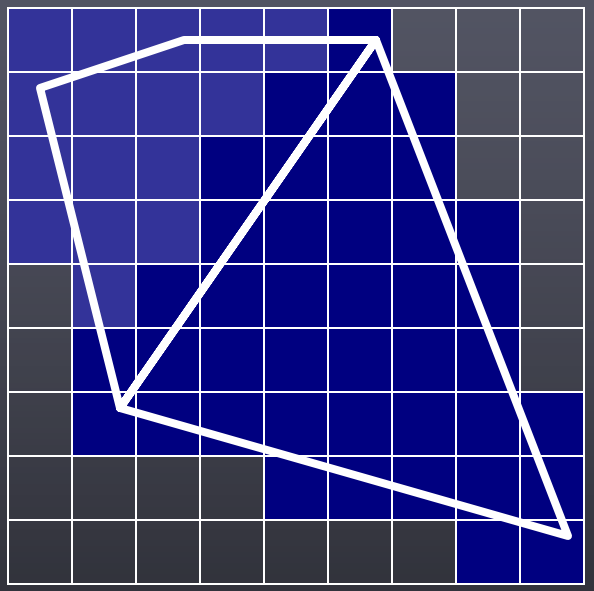
Linjär interpolation

Perspective-correct

Linjär interpolation gör att texturer och ljussättning ser fel ut.
Lösningen är att använda "rational linear interpolation".
Om texturkoordinaterna är 4-dimensionella homogena koordinater kan korrigeringsfaktorn lagras på ett sådant sätt att det sedan räcker med linjär interpolation.

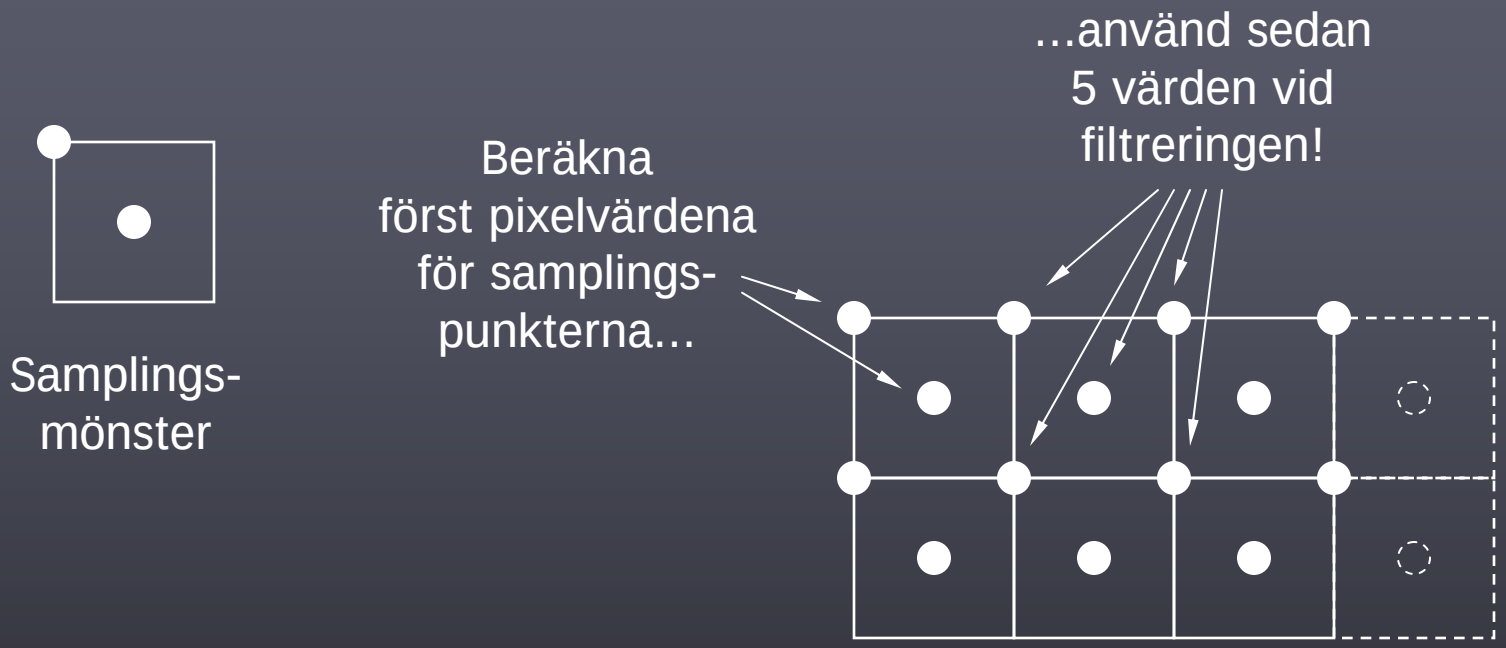
Rastrering av polygoner





Komplikation…
Om två polygoner delar en kant ska
bara den ena fylla i pixelpunkterna!
Annars funkar inte t.ex. färgblandning.

Multi-Sample Anti-Aliasing (MSAA)

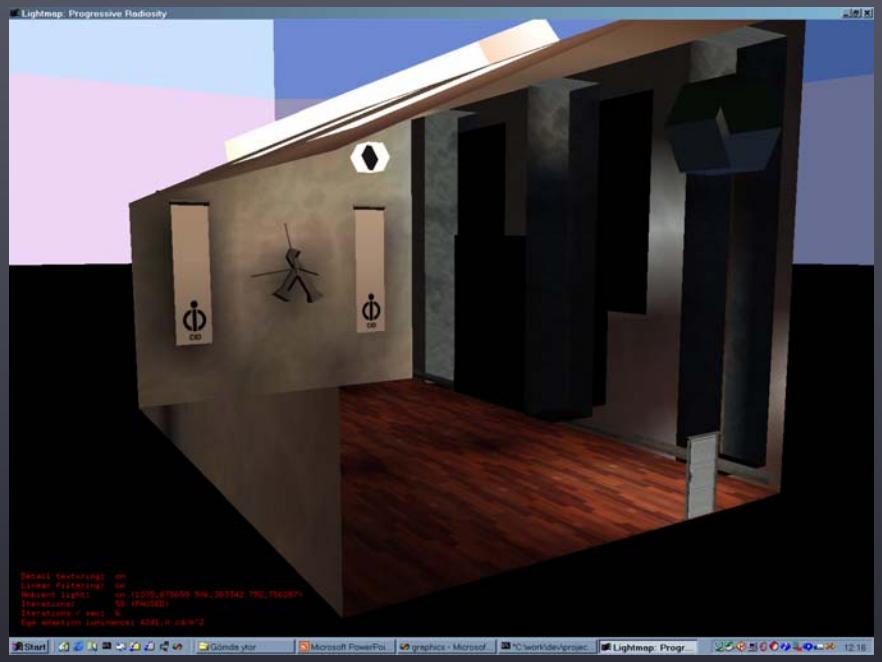


Kräver mer minne i bildbufferten (x2 i det här fallet).

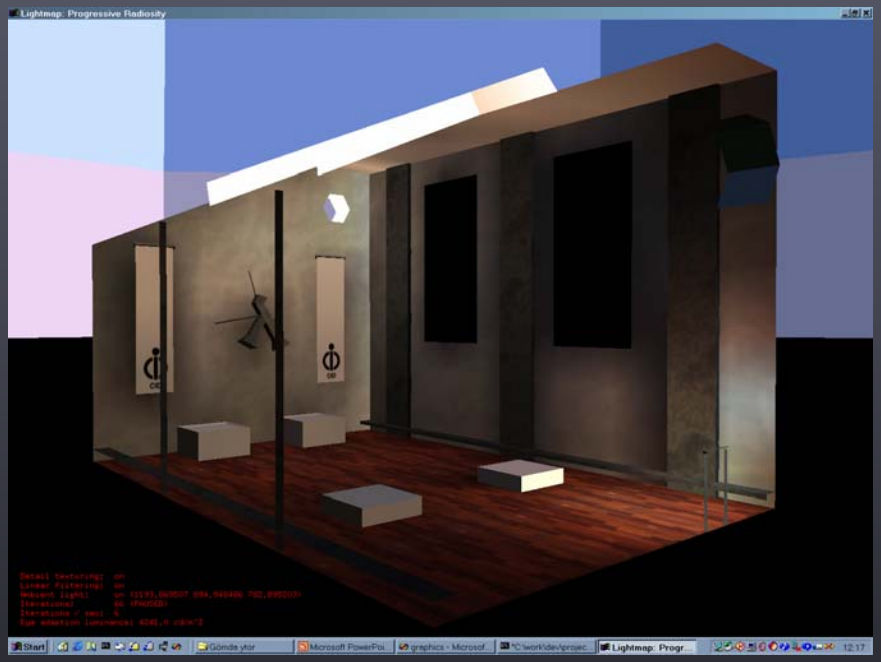
Det finns andra varianter av antialiasing också.

Ofta är samplingsmönstren olika för texturer och geometri.
Normalt tar man fler samplingar av texturer än geometri.

Borttagning av skymda ytor

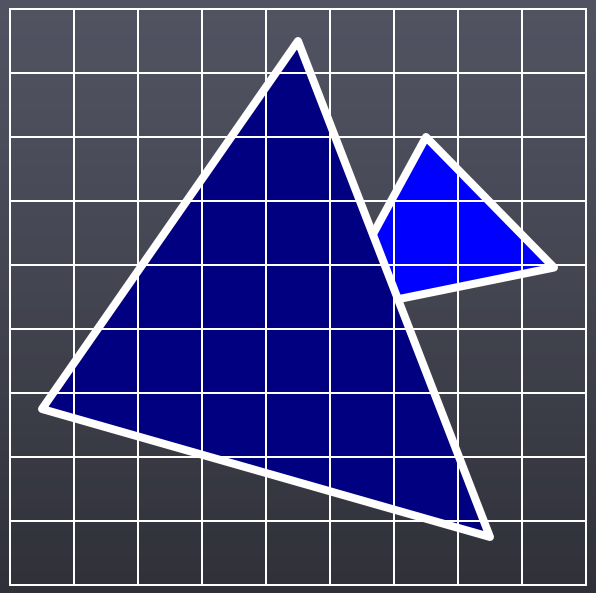


Före...

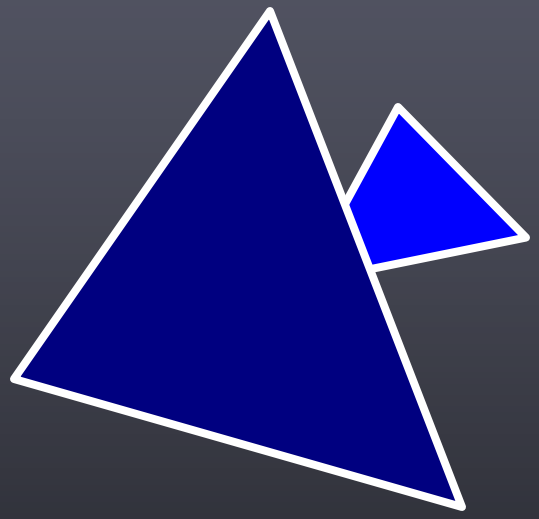


Efter!

Bildrymd vs. objektrymd



Bildrymd:
Ta bort skymda ytor
i rastreringssteget.



Objektrymd:
Klipp polygonerna så att bara
det som kommer att synas
rastreras.

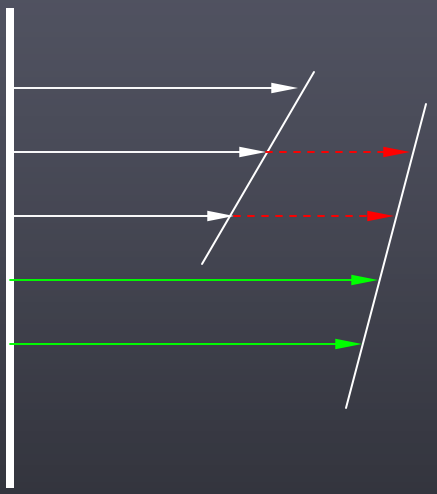
Detail culling

Rita inte saker som är långt bort!
Kombineras ofta med någon form av dimma-effekt.

Ultima IX – Ascension, Origin Systems

Morrowind, Bethesda Softworks

Z-buffring

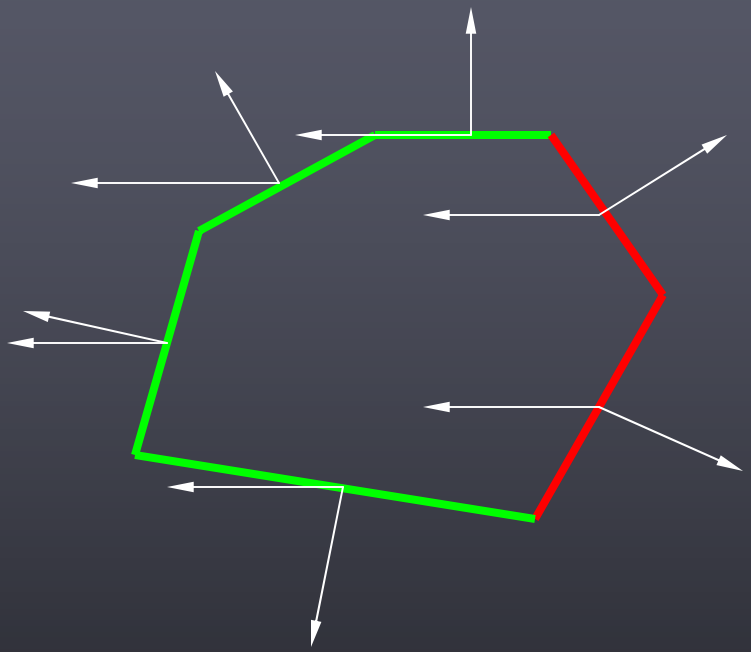


Interpolera och lagra djupvärden för rastrerade trianglar. När en pixel rastreras, kontrollera om dess djupvärde är större än det värde som redan finns i bufferten. I så fall syns inte pixeln.

Z-buffring

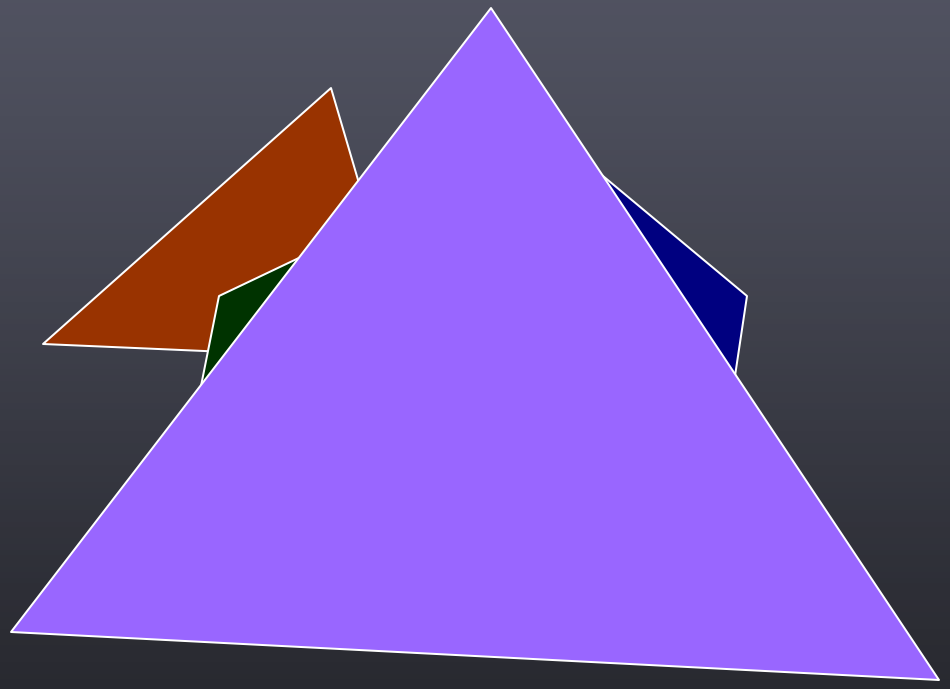
- Fördelar:
  - Effektiv.
  - Funkar oberoende av komplexiteten i det som ritas.
  - Enkel att implementera i hårdvara.
- Nackdelar:
  - Aliasing! 8 bitar $\Leftrightarrow$ 256 distinkta djupvärden.
  - Krävs extra minne på grafikkortet.

Back-face culling



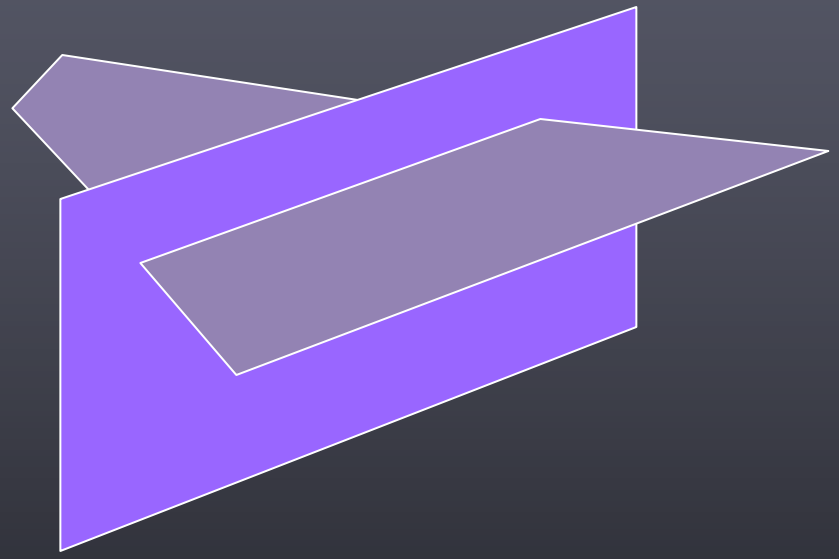
Rita inte om vinkeln mellan kamerariktningen och polygonens normal är större än 90°.
Funkar då modellerna inte är genomskinliga.
Tar inte bort alla gömda ytor (förstås).

Painter's algorithm



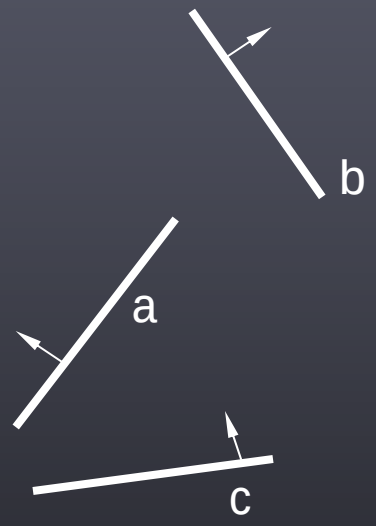
Sortera polygonerna bakifrån och fram.
Rastrera dem en och en.

Problem med painter's algorithm

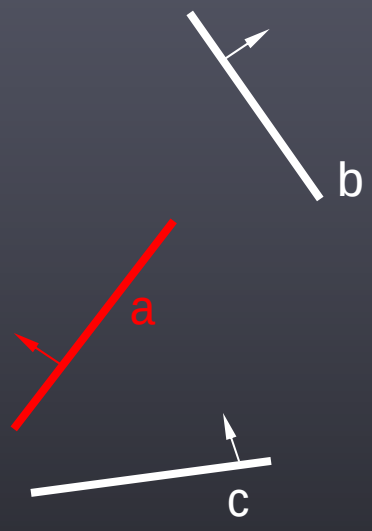


Om två polygoner korsar varandra måste vi dela en av dem!

Binary Space Partitioning Trees (BSP-träd)

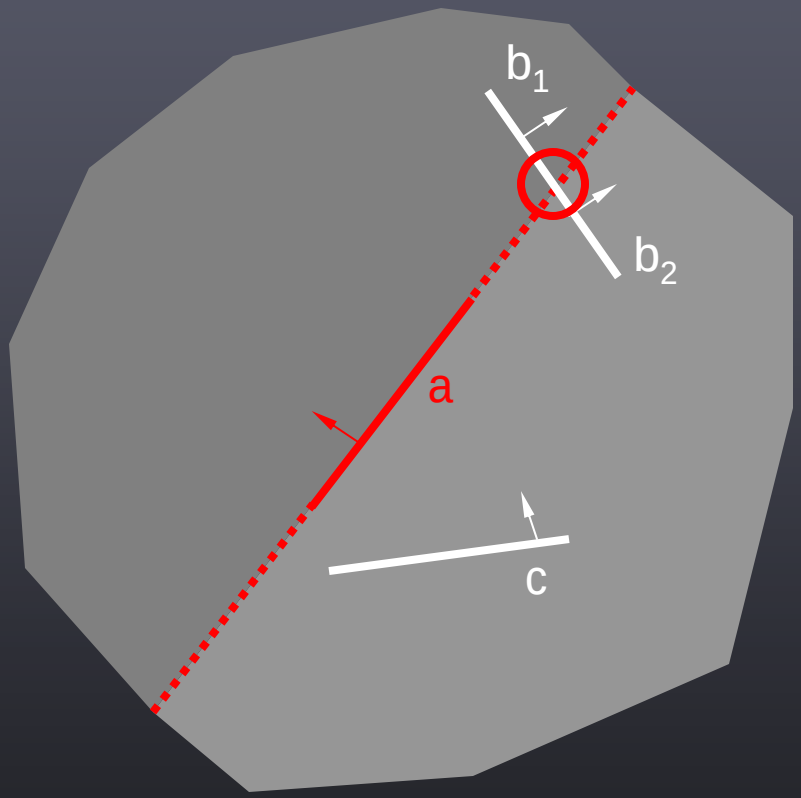


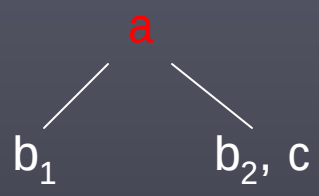
Löser sorteringsproblemet för
statiska, plana, konvexa polygoner
(som dock kan ha hur många hörn som helst).



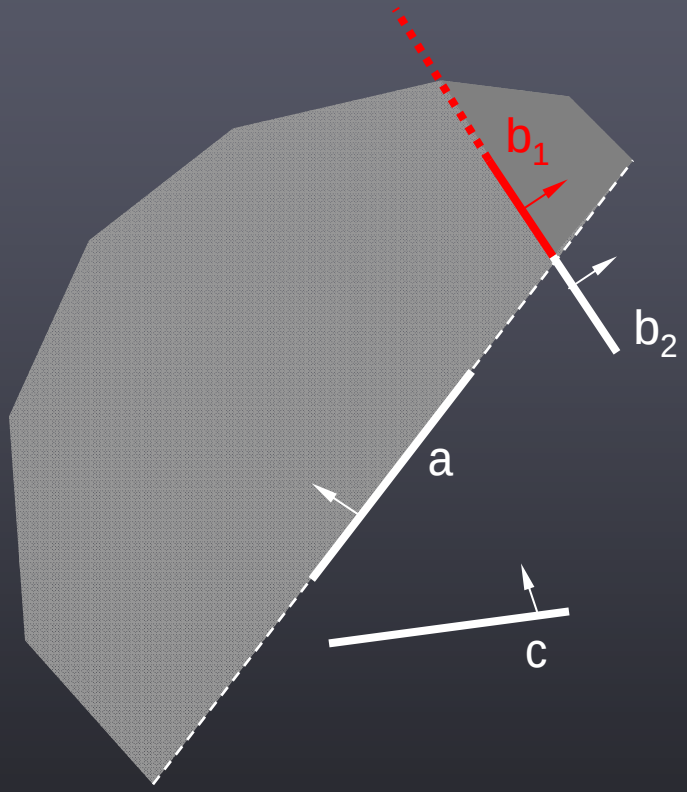
a

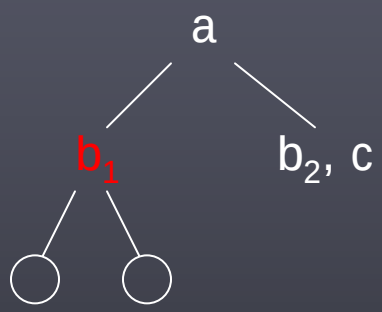
1) Välj en polygon.



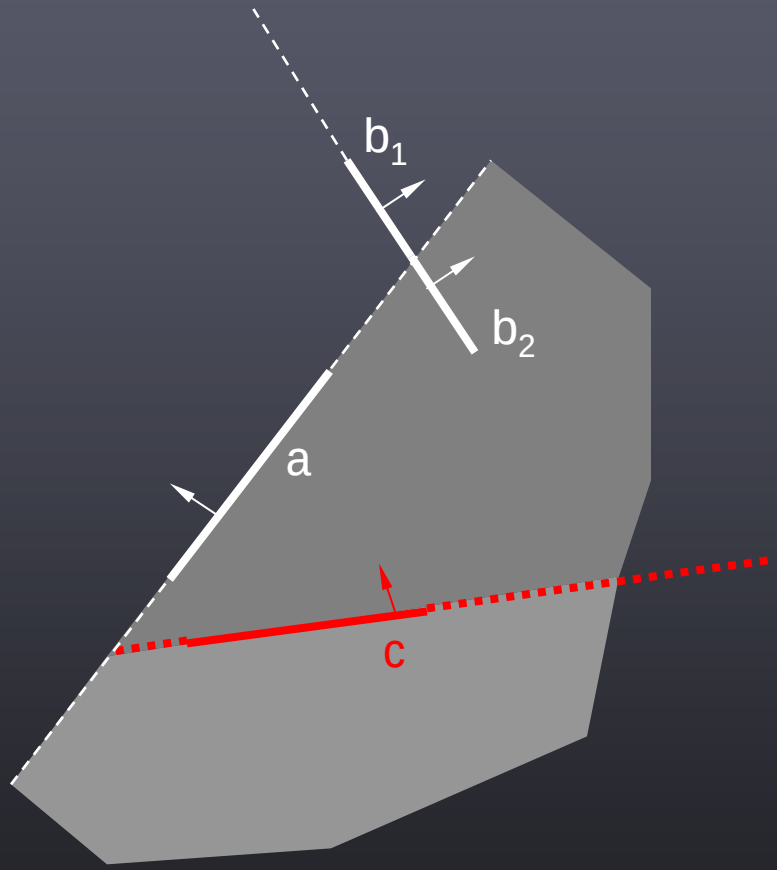


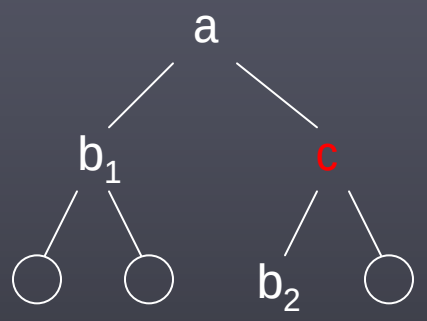
2) Dela rummet i två delar och partitionera.



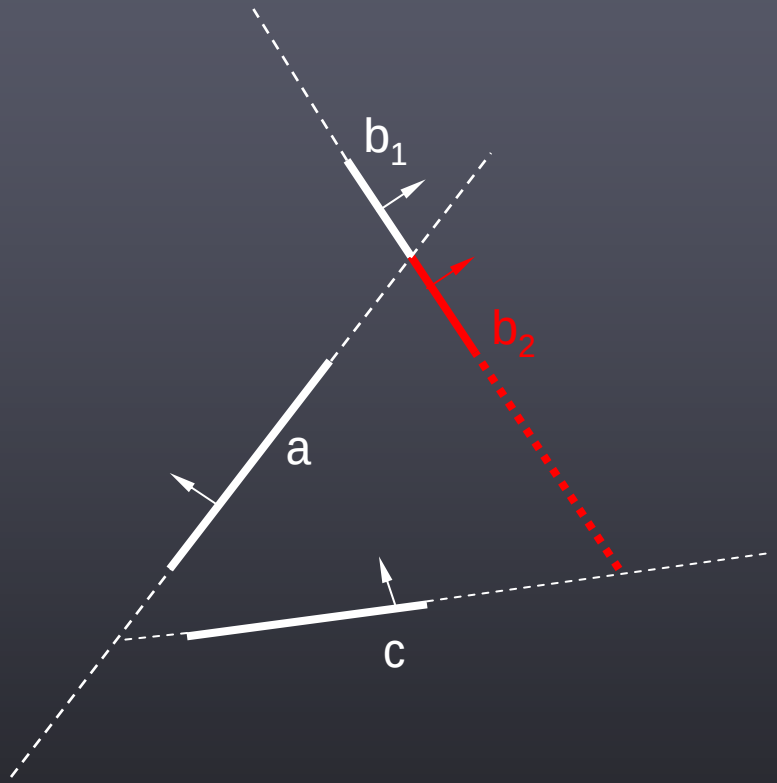


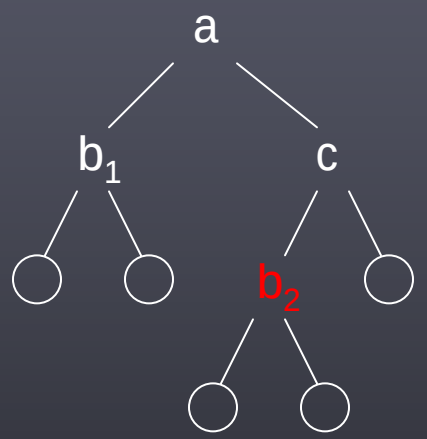
3) För varje delmängd, gör om rekursivt.





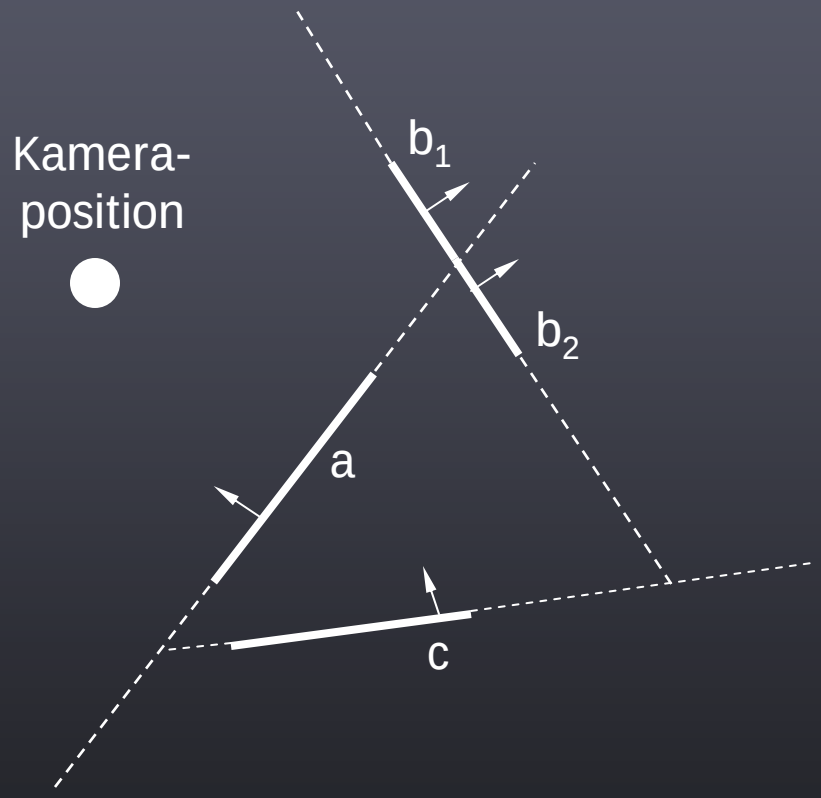
3) För varje delmängd, gör om rekursivt.

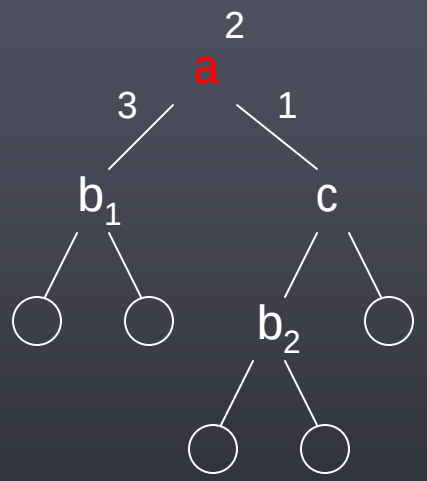




Klart!

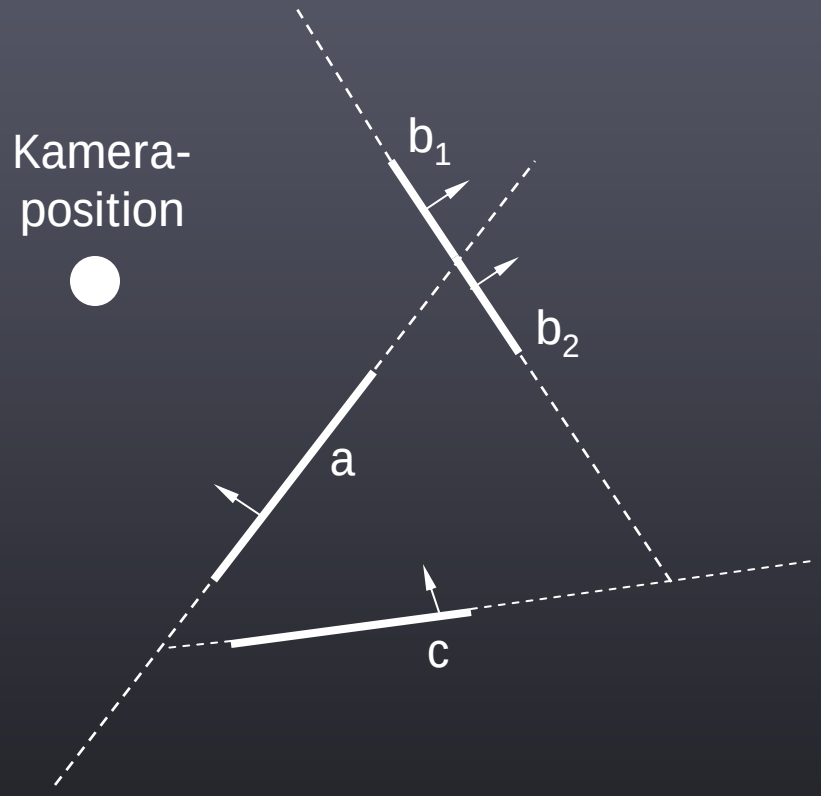
Painter's algorithm och BSP-träd

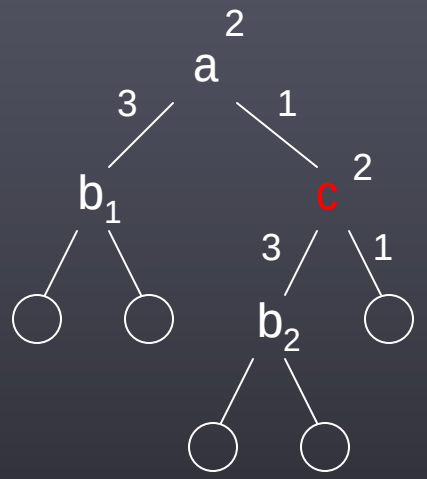




Kameran är framför a, så vi ska först rita ut allt bakom a, sedan a och sist allt framför a.

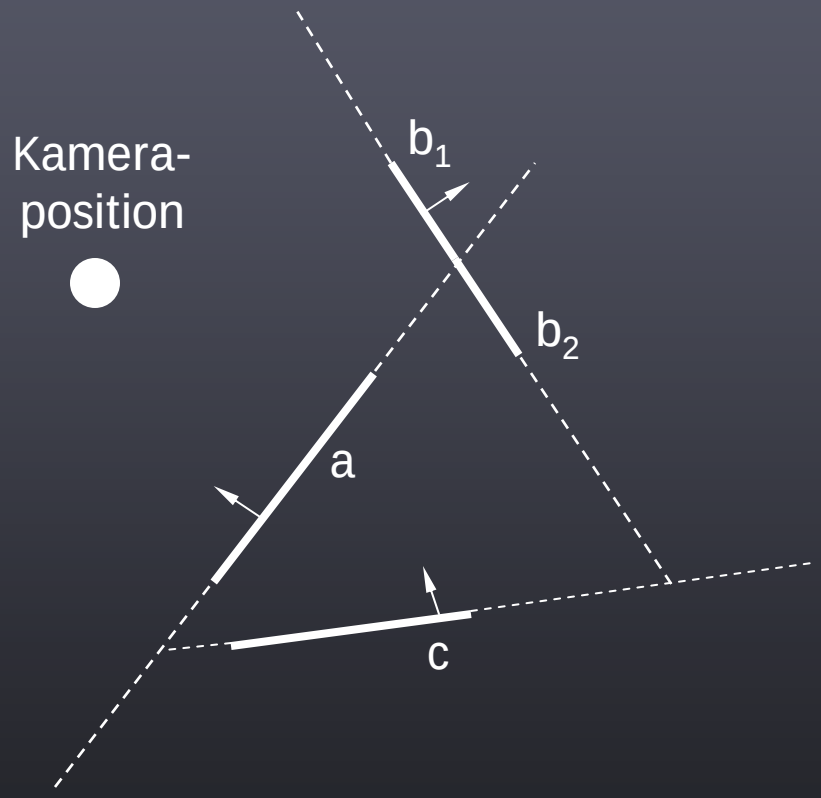
Painter's algorithm och BSP-träd

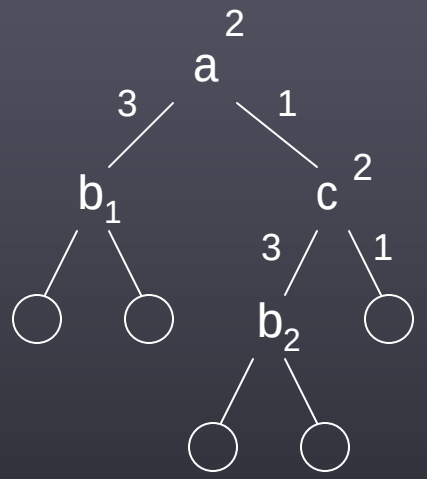




Kameran är framför c, så vi ska först rita ut allt bakom c, sedan c och sist allt framför c.

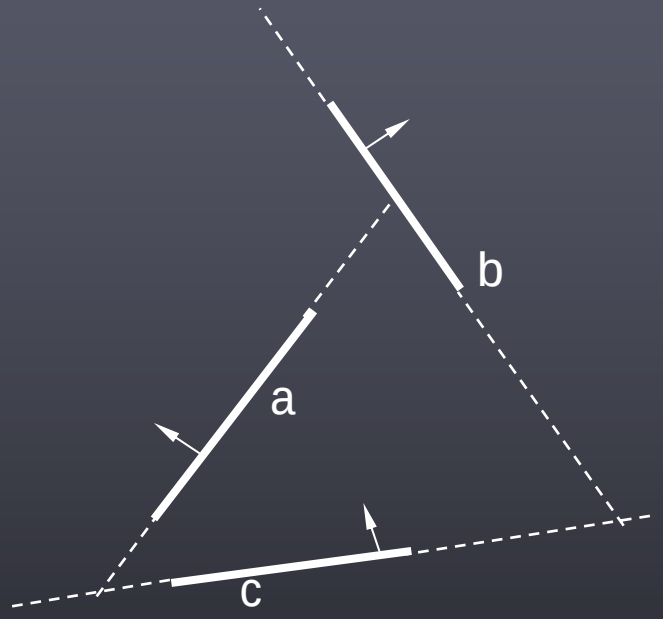
Painter's algorithm och BSP-träd

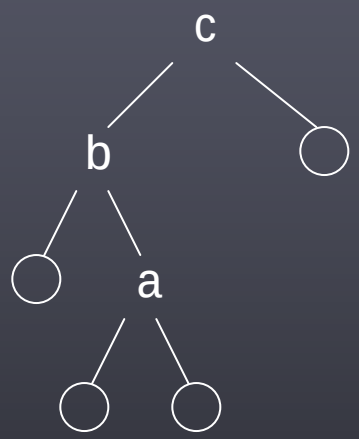


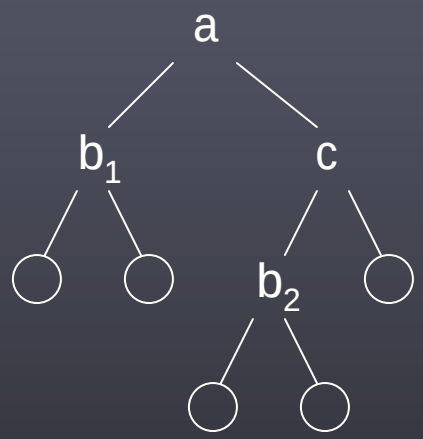


Slutordning: c, b_2, a, b_1.

Val av partitionsplan





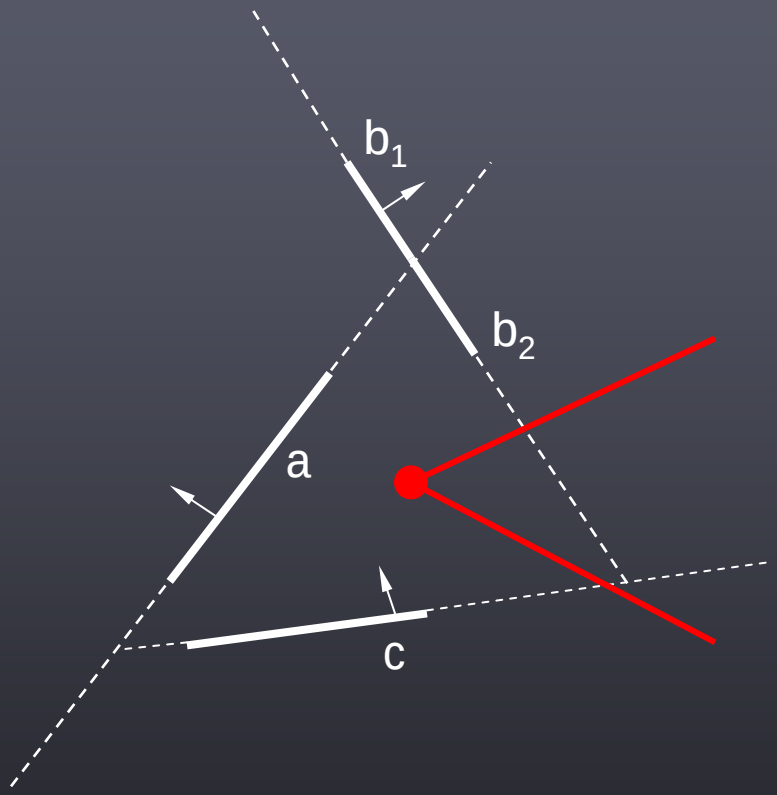


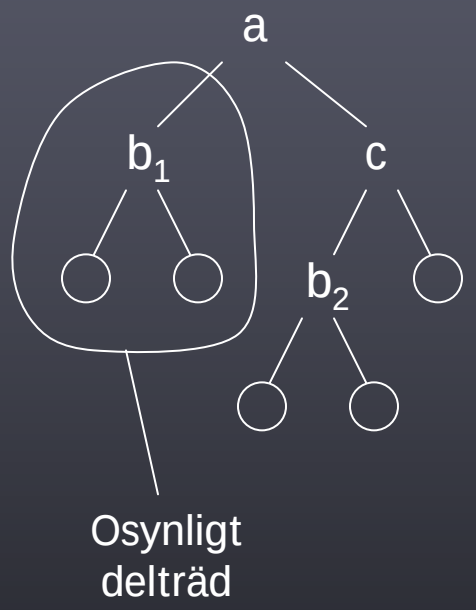
Val av partitionsplan kan ha stor betydelse för slutresultatet.
Tumregel: Välj det partitionsplan som skapar minst antal polygondelningar.

Overdraw

- Med painter's algorithm kommer polygoner långt bak att ritas över av polygoner längre fram.
- Betyder att ljussättning, texturering, m.m. appliceras i onödan för de flesta polygoner!
- Detta problem kallas för ”overdraw”.
- Så varför används BSP-träd om man lika gärna kunde använda z-buffring för att undvika overdraw?

Kombination av BSP-träd och Z-buffring

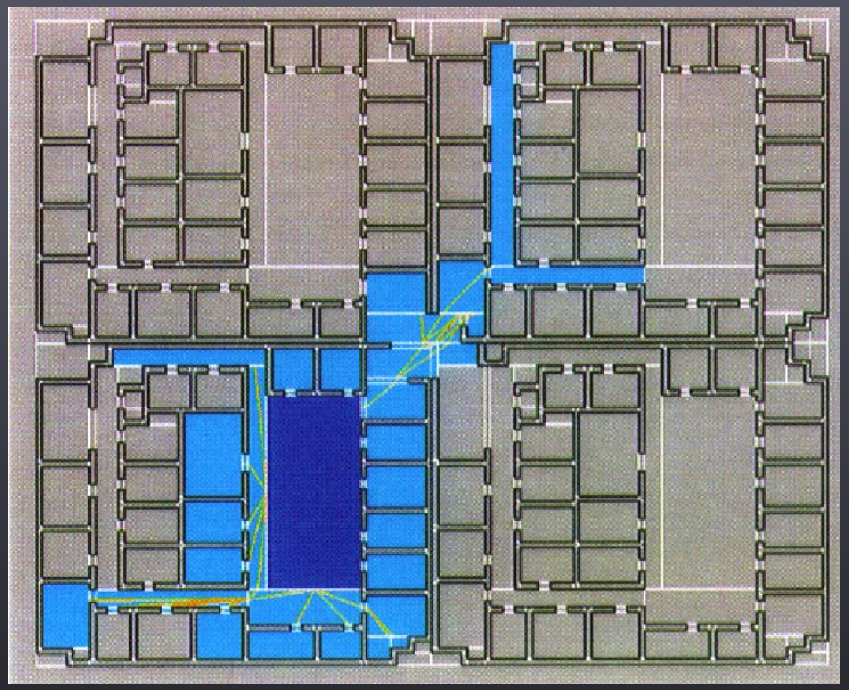




Eftersom kamerans frustum inte korsar a:s plan,
kan vi strunta i allt som ligger framför a!
Använd sedan z-buffring för att undvika overdraw.

Potentially Visible Sets / Portaler

- Närliggande variant:
- Dela världen i celler (kanske m.h.a. BSP-träd).
- För varje cell C, ta reda på vilka andra celler som kan synas från C. Lagra i en lista.
- Vid körning, ta reda på vilken cell kameran är i. Rita den cellen. Rita alla celler i PVS:en.

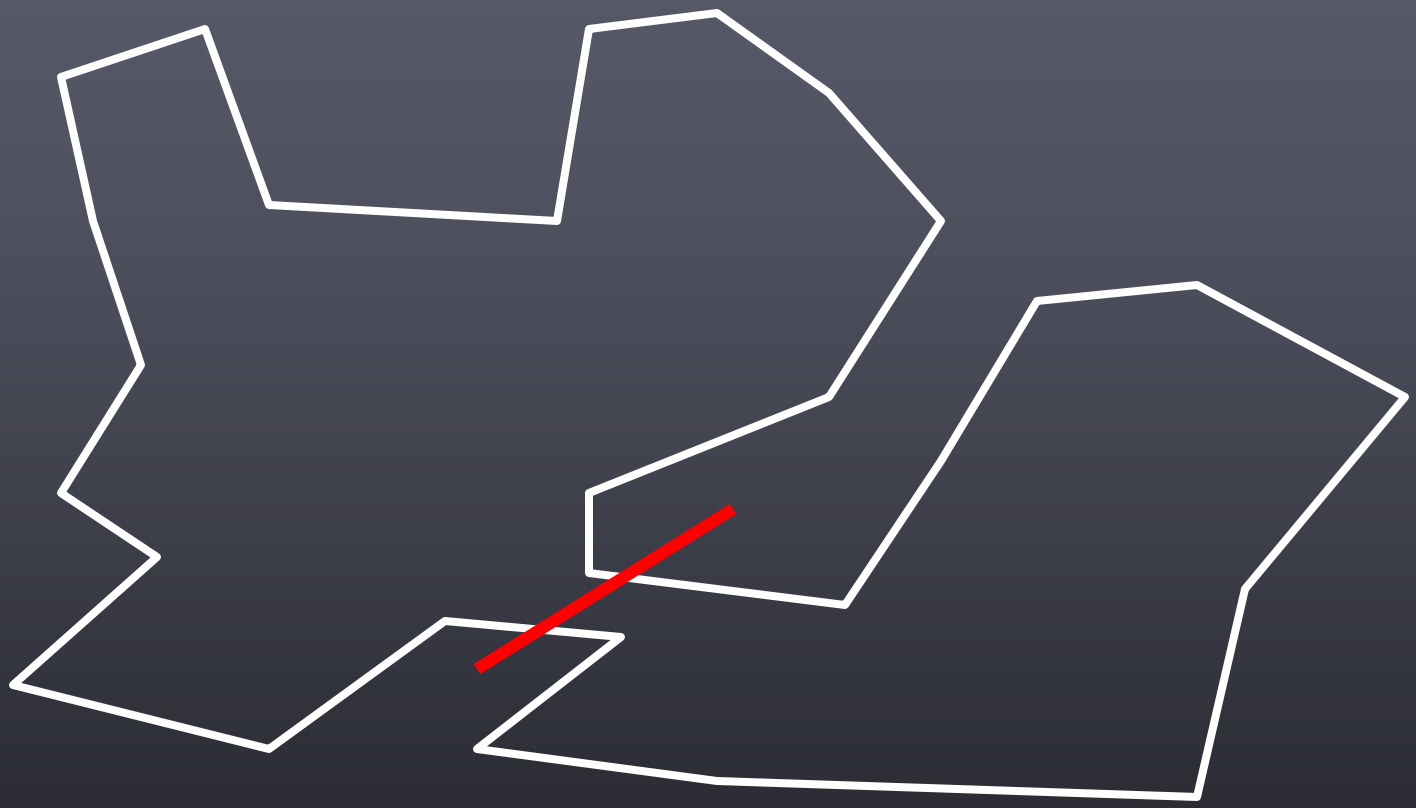


http://graphics.lcs.mit.edu/~seth/pubs/pubs.html

Exempel

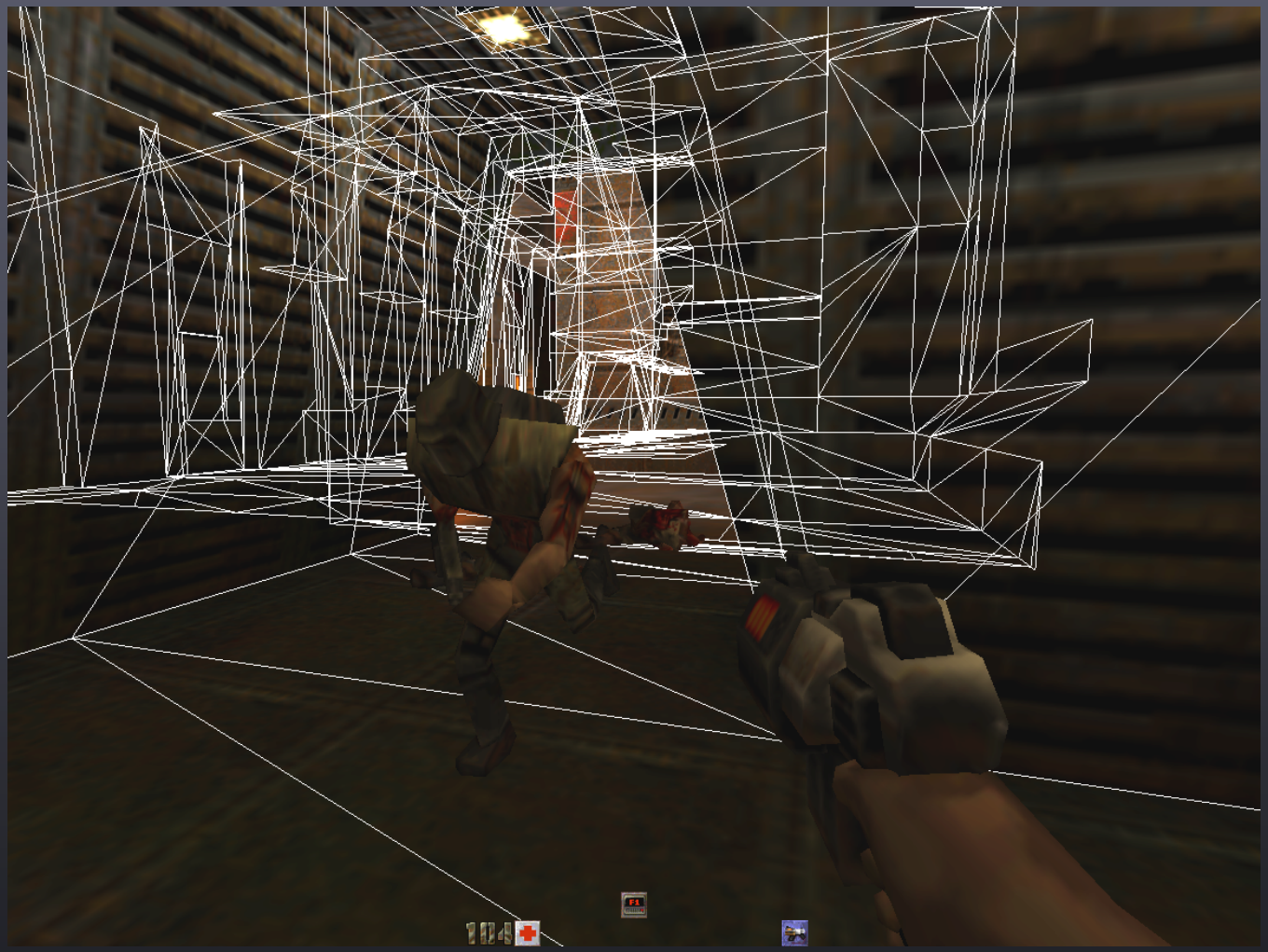
http://www.cs.virginia.edu/%7Eluebke/publications/portals.html

Portaler



Portaler kan specificeras automatiskt med hjälp av algoritmer eller placeras ut för hand i en 3D-editor.

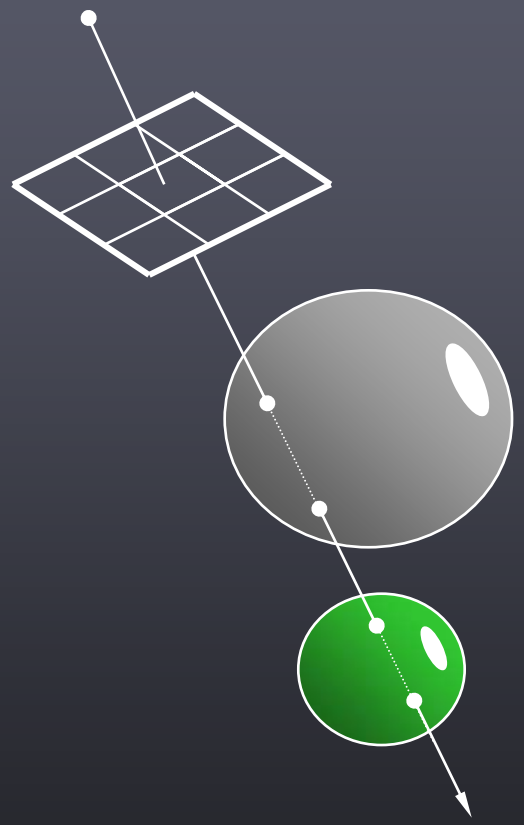
Portaler



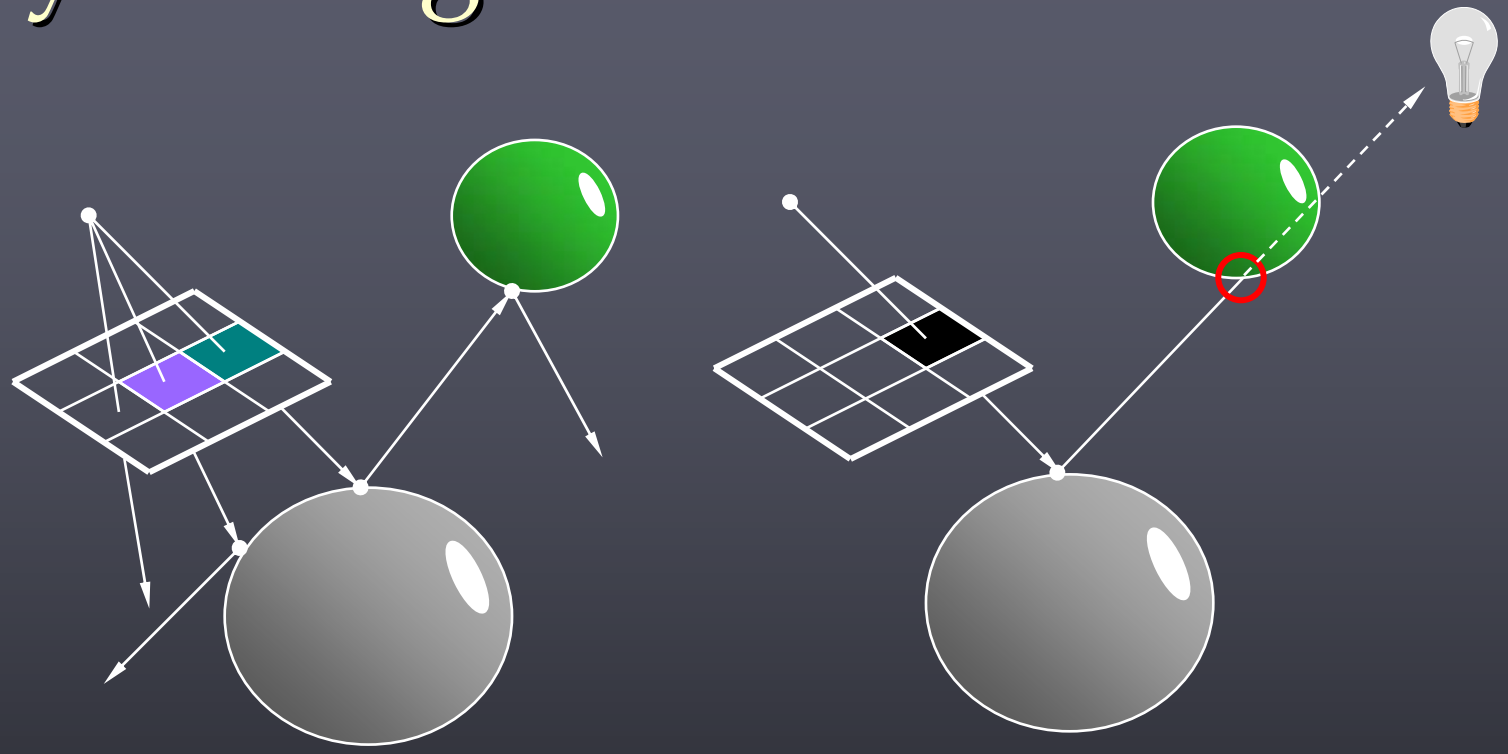
Quake II, Id Software

Ray casting

- Skicka en riktad linje (ray) från fokalpunkten genom varje pixel och ta reda på om/var den korsar alla föremål.
- Välj den korsning som ligger närmast kameran.
- Sätt pixelfärgen till färgen på motsvarande föremål.



Ray tracing

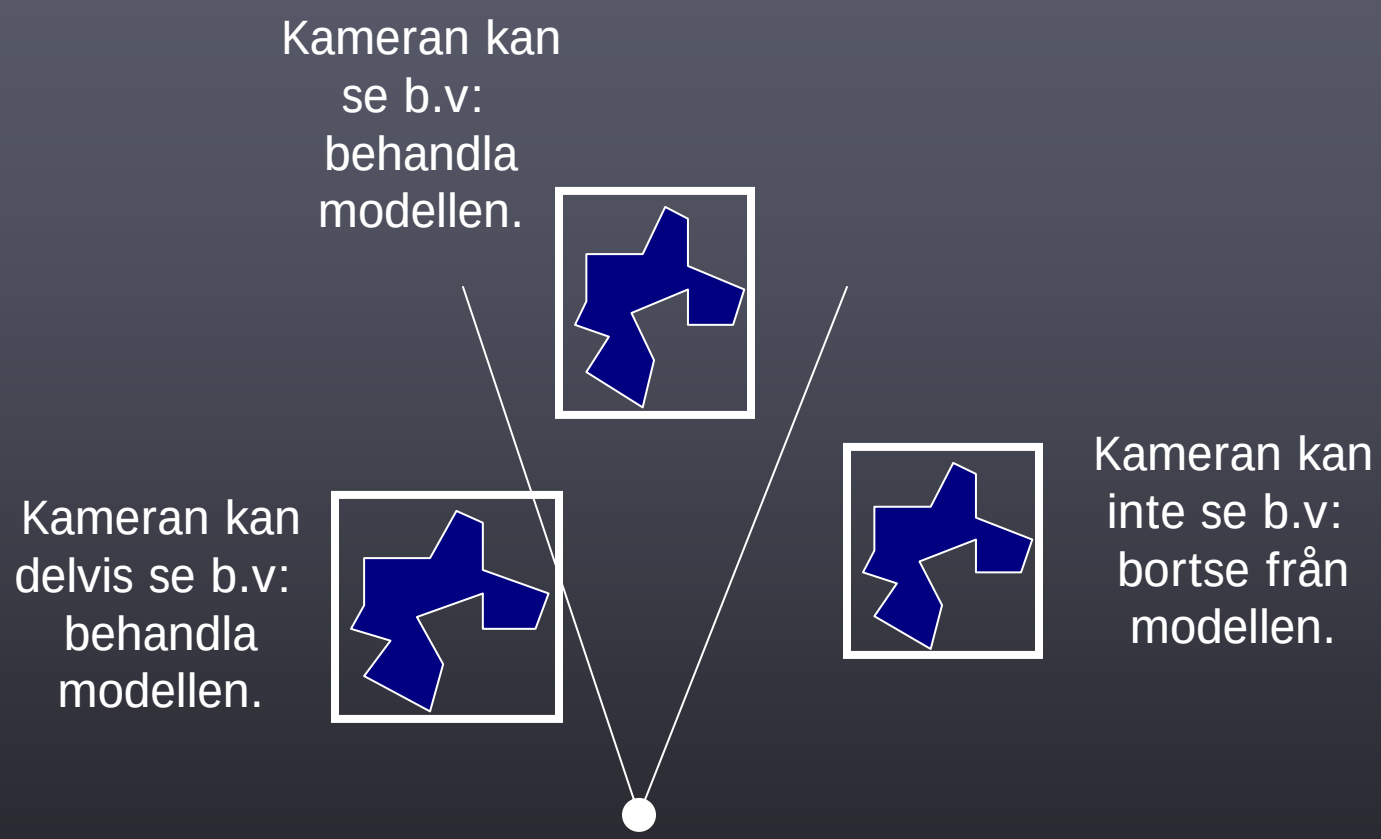


Reflektion

Skuggor (direkt ljus)

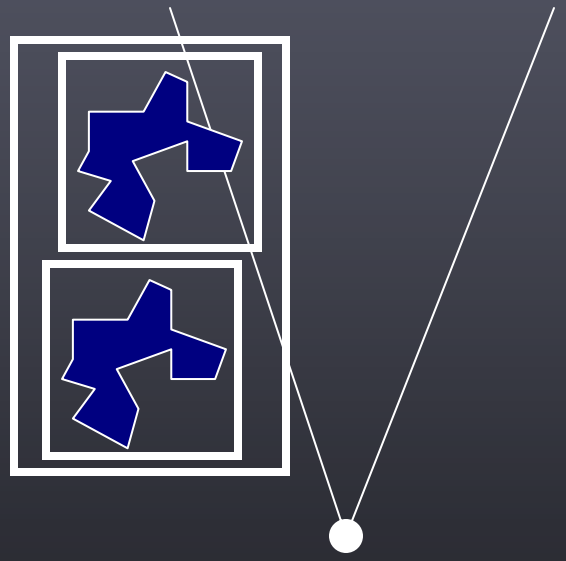
Ray tracing är en s.k. **global belysningsmodell**.
Blanda inte ihop ray casting (visibilitymetod)
och ray tracing (bildsyntesmetod som använder ray casting)!

Bounding volumes



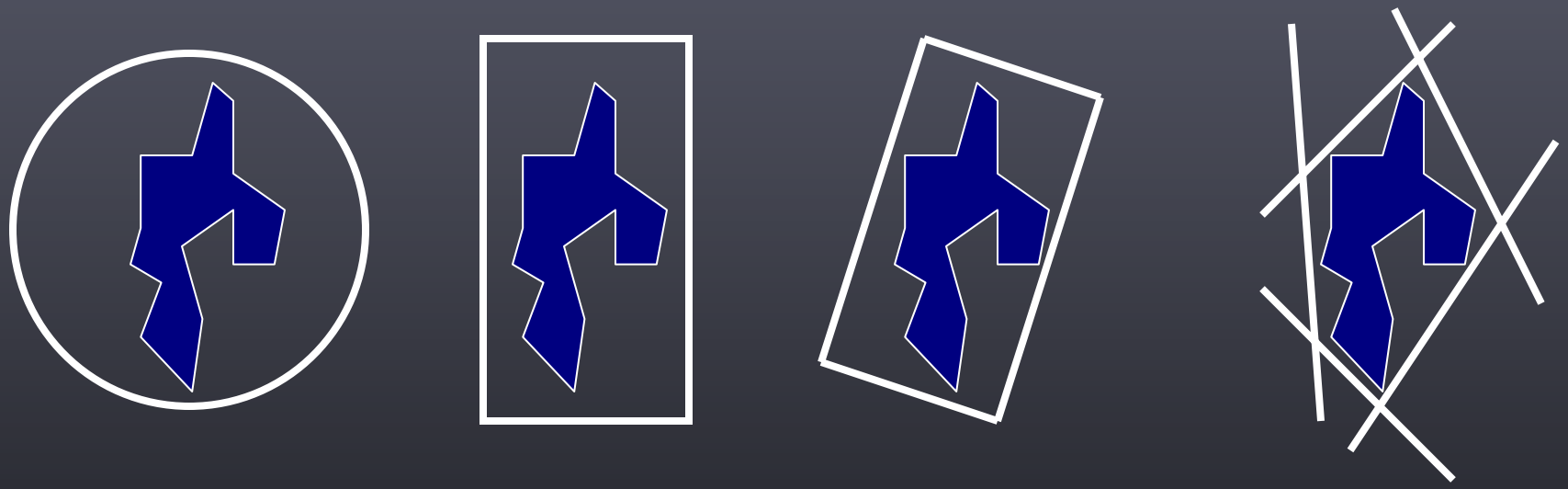
Användbart för modeller med begränsad utsträckning i rummet och för saker som flyttas omkring.

Hierarkiska bounding volumes



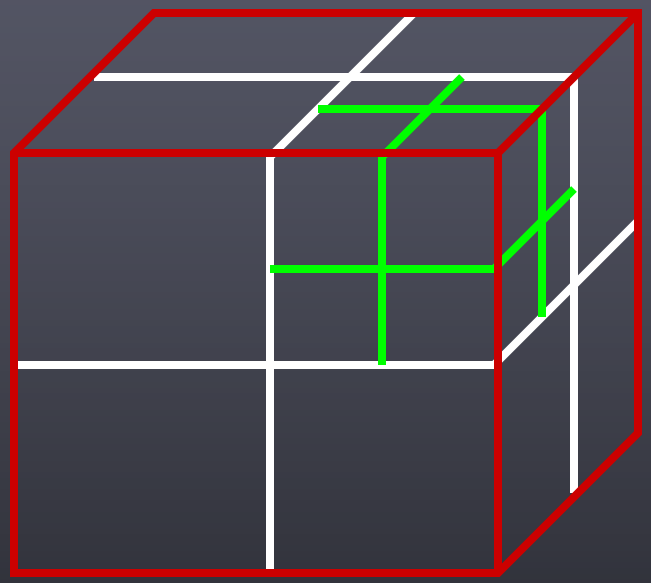
Hierarkier kan också skapas dynamiskt under körning för föremål som flyttas omkring.

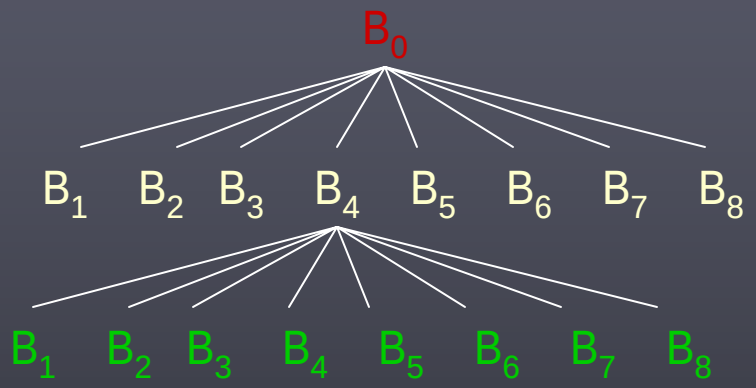
Exempel på olika sorters bounding volumes



Sfär | Axis-aligned box | Oriented box | Oriented planes

Quadtrees / Octrees





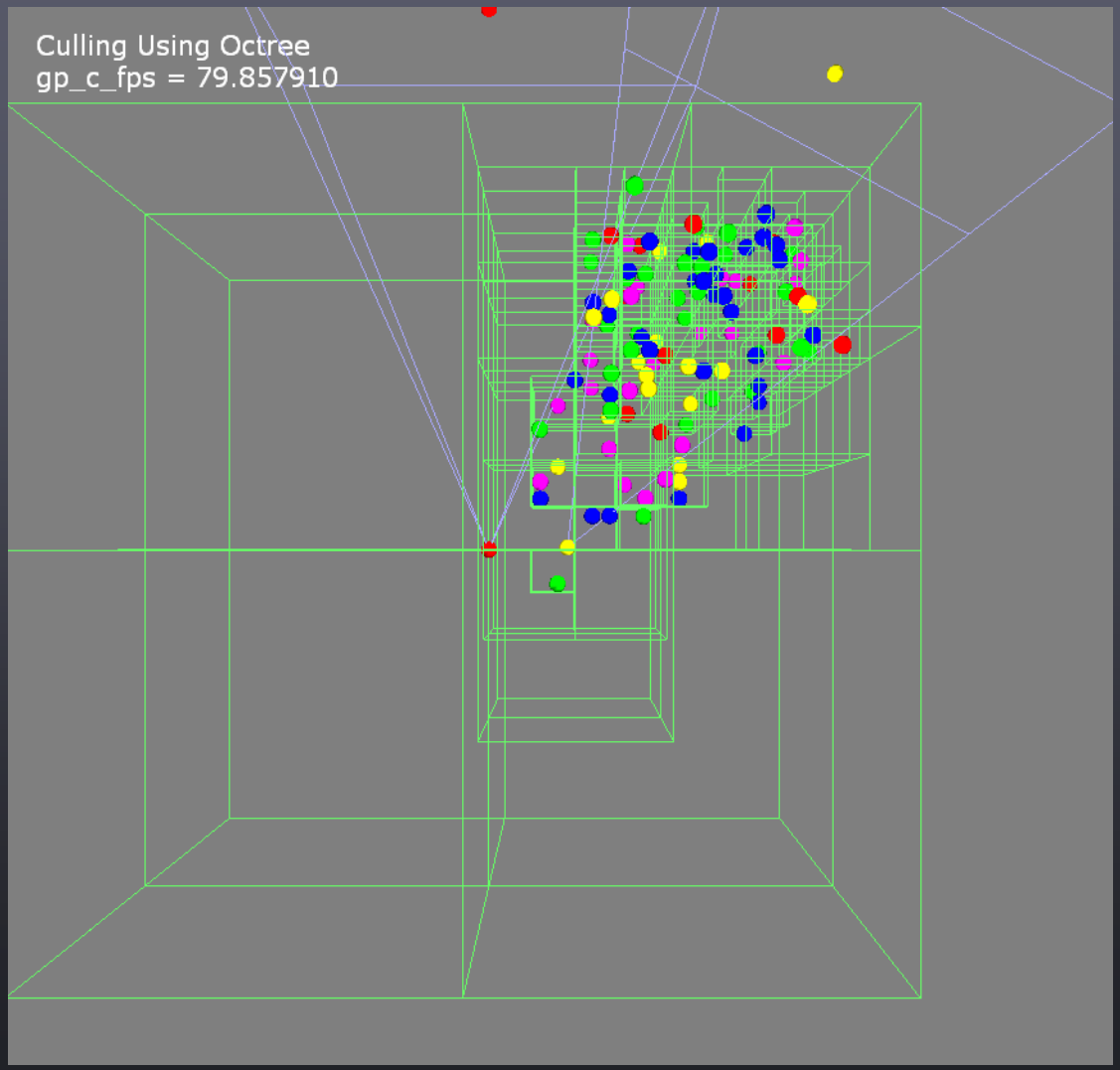
Variant av BSP-träd som kan vara enklare att hantera.
Smidiga för stora, statiska modeller med stor detaljrikedom.

Kombineras med Z-buffring på ungefär samma sätt som i BSP-fallet.

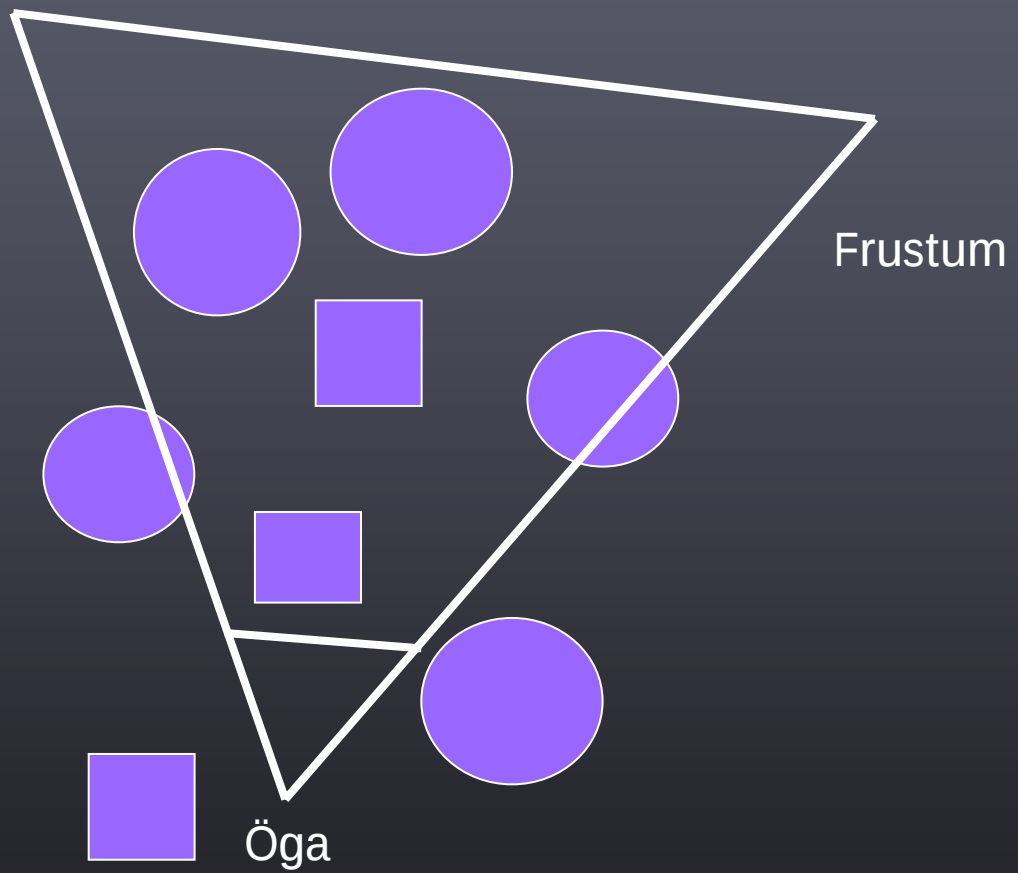
Octrees

http://www.gametutorials.com/Tutorials/OpenGL/Octree.htm

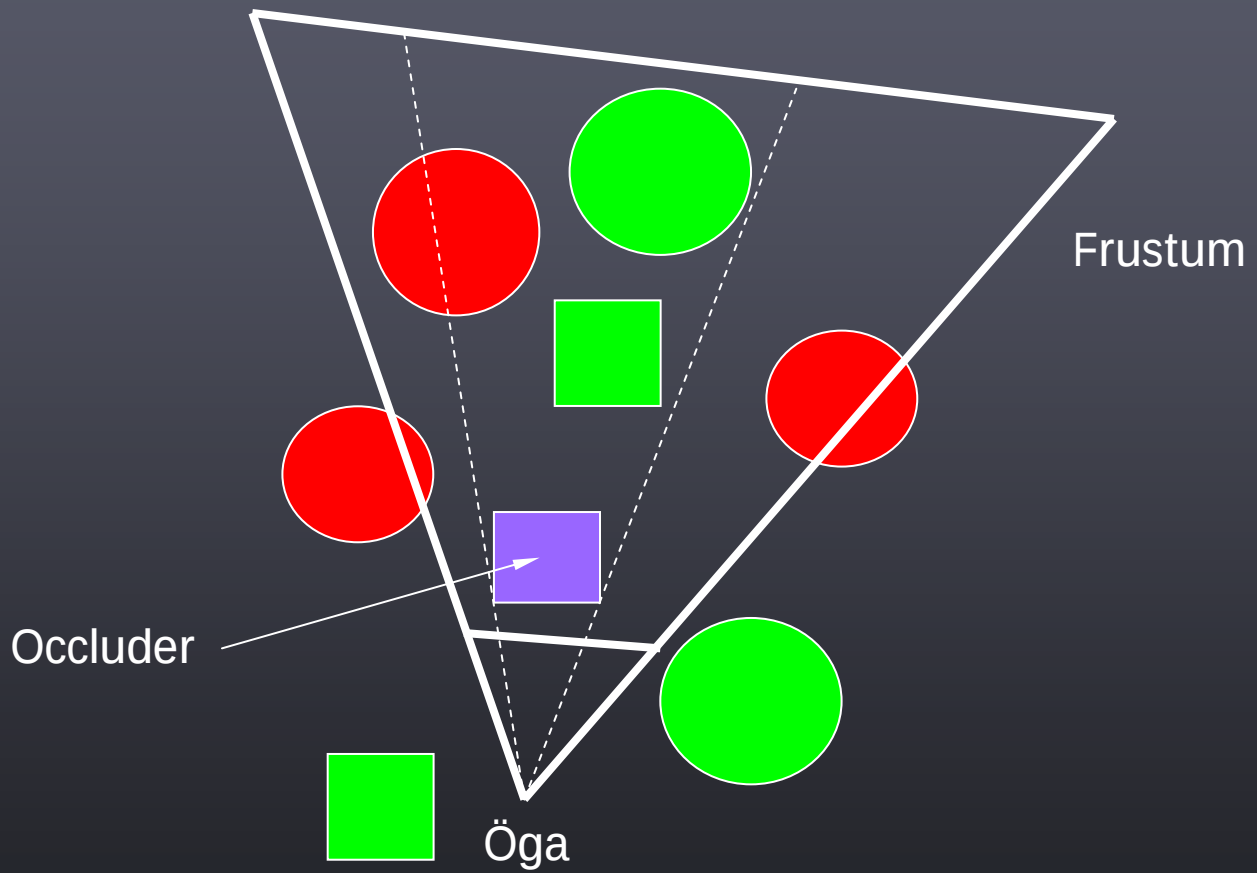
Octrees



Occlusion culling



Occlusion culling



http://www.gamasutra.com/features/19991109/moller_haines_01.htm

Occlusion culling

Just Cause - Avalanche Studios

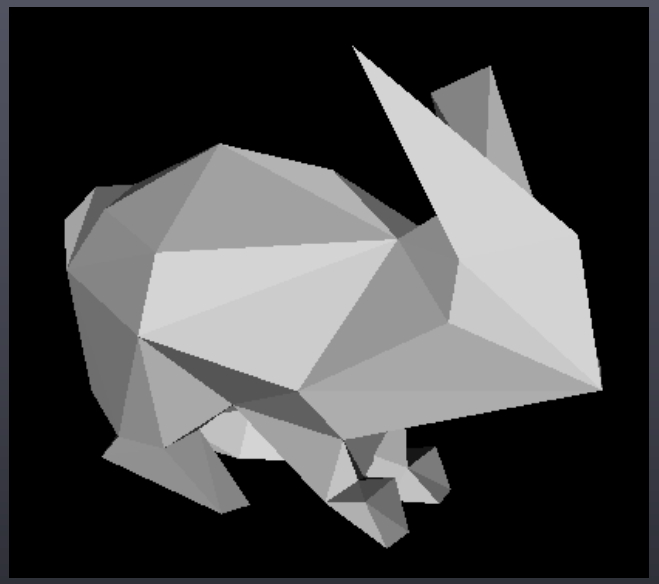
Ibland kan en hög detaljrikedom i scenen faktiskt vara till hjälp!

Level-of-detail

Just Cause - Avalanche Studios

Även om man måste rita många objekt kan man ofta spara resurser genom att använda grövre detaljnivåer långt bort från kameran.

Level-of-detail

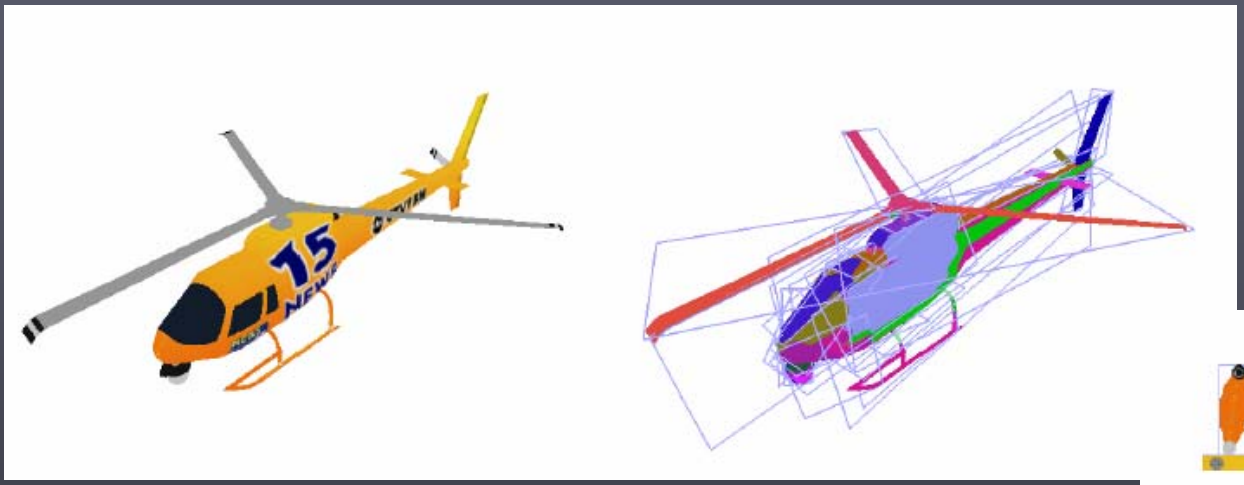


Impostors

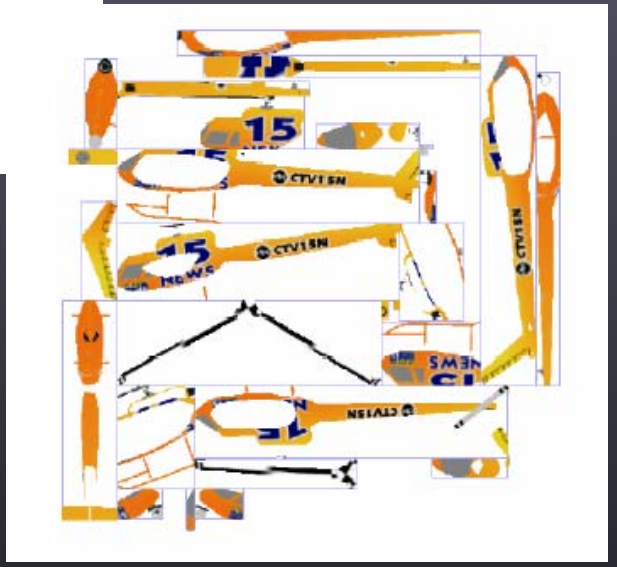
Ett specialfall av level-of-detail. Rendera ett komplext objekt till en textur och använd den istället. Om objektet är långt från kameran är parallax-effekten liten, så en impostor kan återanvändas för fler frames även om kameran flyttas.

Kallas också **billboards**.

Impostors



Billboard Clouds, Xavier D´ecoret m.fl.



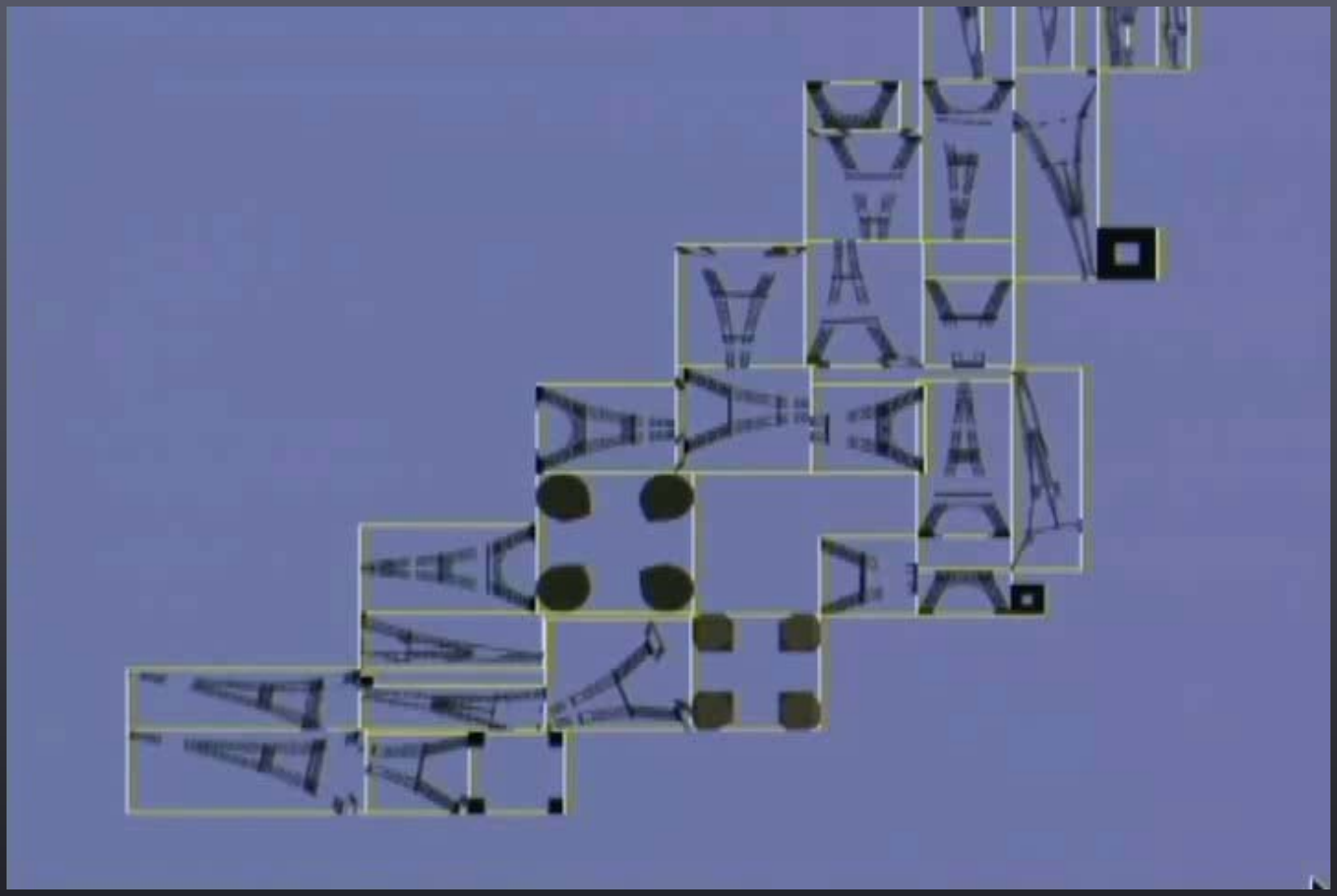
Placerar man texturerna smart kan man approximera föremål på ett sådant sätt att man kan rotera dem också!

Impostors

Original

Billboard cloud

Impostors



Billboard Clouds, Xavier D´ecoret m.fl.

Impostors

Microsoft Flight Simulator 2004

Moln är ett exempel på objekt som ofta kan ersättas av impostors.

Impostors

Microsoft Flight Simulator 2004

I flygsimulatorfallet lönar det sig att arbeta med impostors i två steg: dels som ersättning för 3D-moln, dels för att approximera moln långt från kameran.