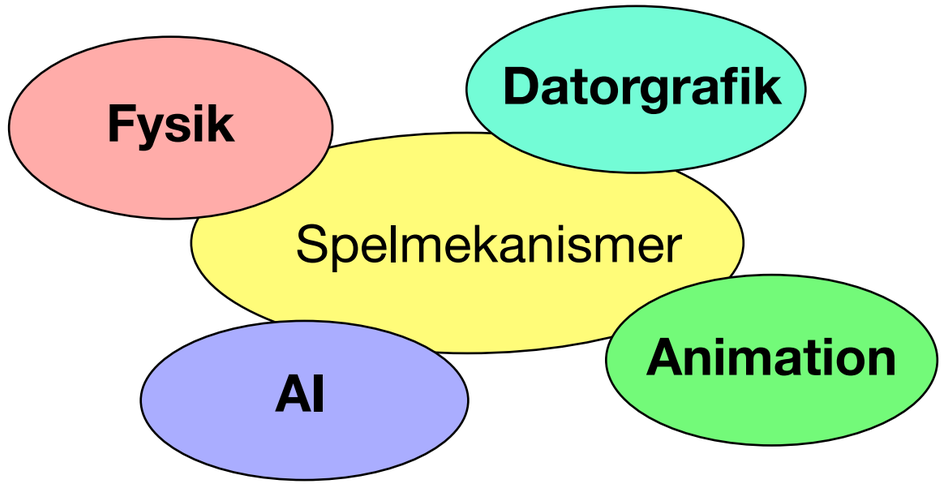
TSBK 03
Teknik för avancerade datorspel

Ingemar Ragnemalm, ISY



Föreläsning 2

Missat förra gången: Repetition.

Buffrar, stencilbuffer
Spegel med stencilbuffer
Stencilbuffer som räknare

Texturering:
• Andra dimensioner
• Multitexturering: Detail textures, scrolling textures
• Projicerade texturer
• Rendering till textur

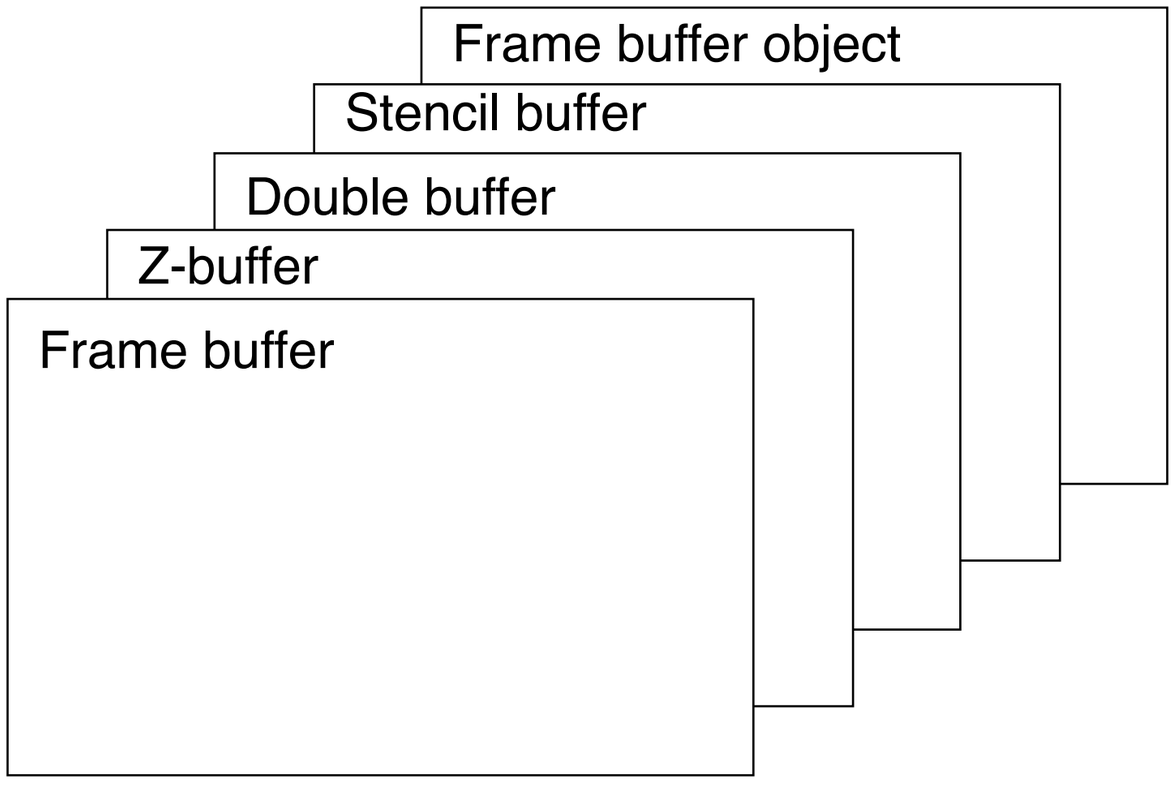
(Meshnavigering)

Ytterligare grafik som följer längre fram:

- Skuggor och ambient occlusion
- Multipass-shaders, faltning
- High dynamic range, blooming
- Bättre bump mapping, parallax mapping, displacement mapping
- Stereo
- Skinning

Bildbuffrar

På GPU'n finns flera bildbuffrar, inte bara den vi ser…



De vanliga

Frame buffer: Den vanliga bildbuffern
Dubbelbuffer (back-buffer): Kopia off-screen
Z-buffer (depth buffer): Lågnivå VSD

samt icke-bildbuffrar:

Vertex buffer objects (VBO): Geometridata i VRAM
Vertex Array Objects (VAO)

Ett par till

Stencil buffer: Maskning

Frame Buffer Objects (FBO): Textur att rita i

Shader Storage Buffer: Buffer med godtyckliga data (fö 5)

Buffer eller textur?

Texturer kan användas för mycket annat än texturdata. Är det en textur eller är det en buffer med data? Det bestämmer du!

Dock, en del specialbuffrar används i speciella steg: Stencil buffer och Z-buffer.

Stencilbuffern

"Stencil", gammal teknik för tryck. "Schablon".
Används för att maskning, för pixelvis bestämma vilka pixlar som får skrivas.

Ritas i med vanliga ritoperationer, t.ex. rita polygoner.

Stencilbuffer som räknare

Buffervärden är heltal t.ex. 8 eller 16 bitar (unsigned). Implementationsberoende.

När du ritar i buffern kan du både sätta värden och räkna upp eller ner.

Tillämpningar för stencilbuffer

Förslag på tillämpningar:

- Maska bort ramar, HUD…
- Dissolve-effekter
- Begränsa ritandet till ett visst objekt, viktigt för t.ex. reflektioner och skuggor
- Portaler
- CSG (Constructive Solid Geometry)

Viktigare än man först tror?

Stencilbuffer i OpenGL:

glStencilFunc(func, ref, mask);
bestämmer stencilbufferns test under ritande

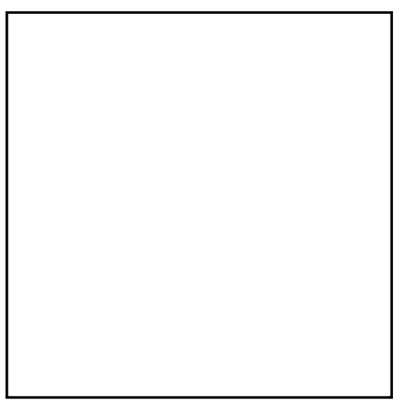
glStencilOp(fail, zfail, zpass);
bestämmer hur stencilbuffern ändras under ritande

Tre utfall: Stencil fail, stencil pass/depth fail, stencil pass/depth pass. Olika operationer kan definieras för alla dessa fall (t.ex. inkrementera, nollställa…)

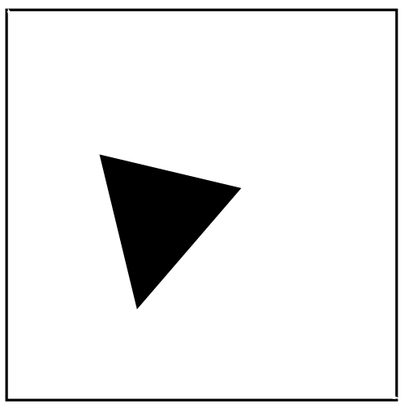
Enkelt exempel

Masking enbart

- Radera stencilbuffern
- Rita masken i stencilbuffern
- Rita andra objekt med stencilbuffer aktiv

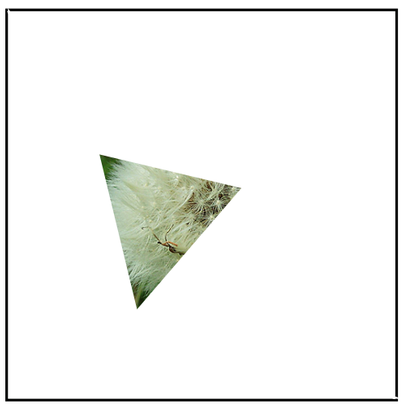


Radera stencil



Rita i stencil

Använt texturerat objekt



och rita maskad med stencil

Förberedelse

Måste initiera OpenGL-context med stencilbuffer:

```
glutInitDisplayMode(GLUT_RGB | GLUT_DOUBLE | GLUT_DEPTH | GLUT_STENCIL);
```

eller motsvarande

Första steget:

Radera stencilbuffern

```
glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT | GL_STENCIL_BUFFER_BIT);
```

Rita masken

Ställ om till att rita i stencilbuffern

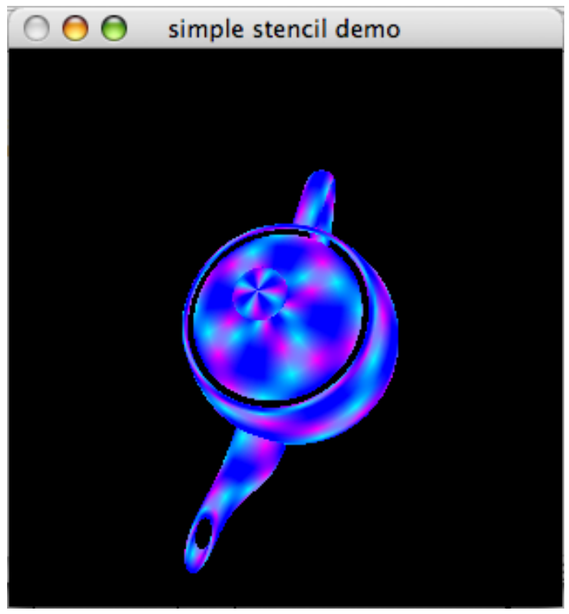
```
// Enable the Stencil Buffer
glEnable(GL_STENCIL_TEST);

// Disable Color Buffer and Depth Buffer
glColorMask(GL_FALSE, GL_FALSE, GL_FALSE, GL_FALSE);
glDepthMask(GL_FALSE);

// Set 1 into the stencil buffer
glStencilFunc(GL_ALWAYS, 1, 0xFFFFFFFF);
glStencilOp(GL_REPLACE, GL_REPLACE, GL_REPLACE);
// Draw the wired sphere in the stencil buffer only
DrawWireframeModel(sphere);
```

Rita scenen

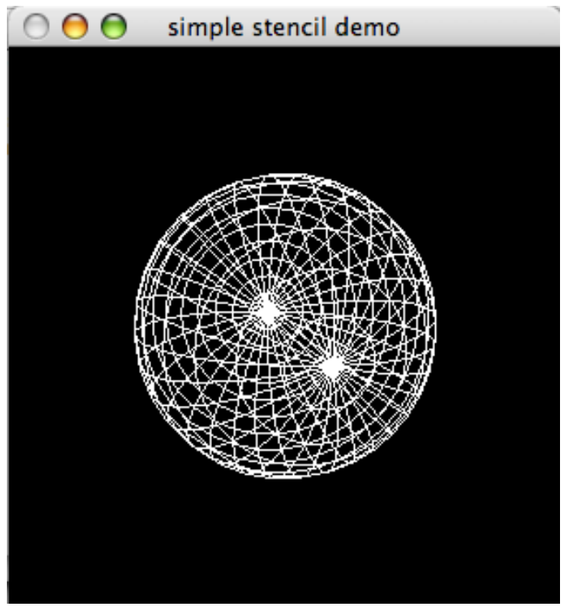
Rita andra objekt med stencilbuffer aktiv

```
// Turn on Color Buffer and Depth Buffer
glColorMask(GL_TRUE, GL_TRUE, GL_TRUE, GL_TRUE);
glDepthMask(GL_TRUE);

// Only draw if Stencil Buffer equals 1
glStencilFunc(GL_EQUAL, 1, 0xFFFFFFFF);
// Keep the content of the Stencil Buffer
glStencilOp(GL_KEEP, GL_KEEP, GL_KEEP);

// Draw the shape
DrawTheShape();

glDisable(GL_STENCIL_TEST); // Turn off
```

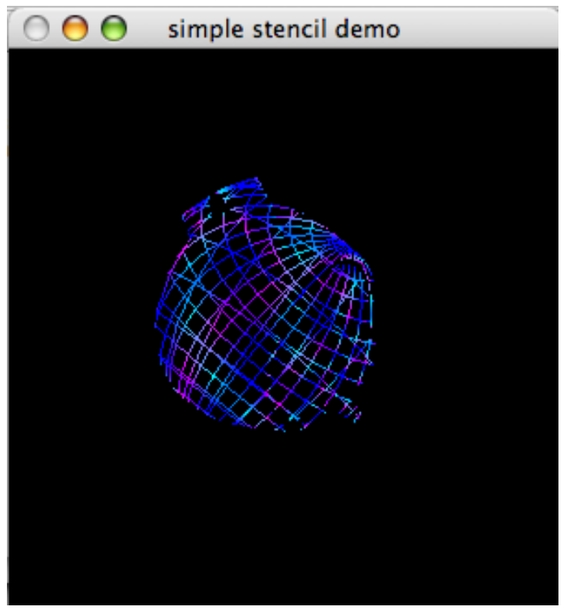
Resultat



Objekt att maska



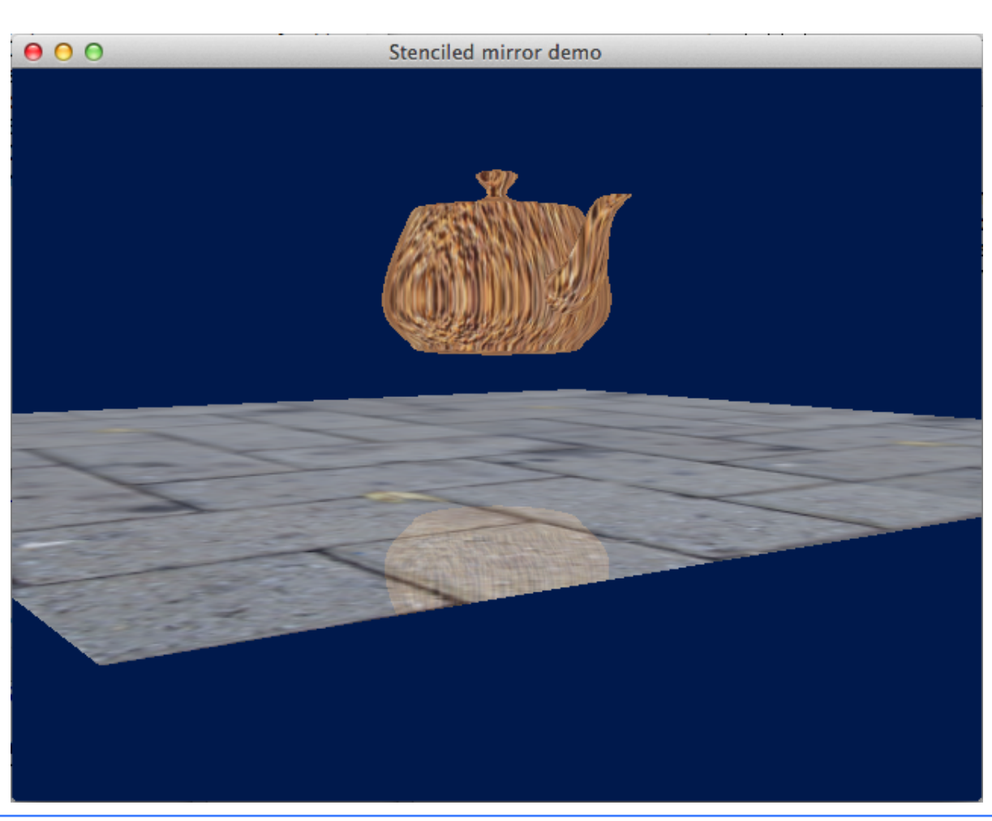
Mask



Resultat

Spegling med stencil

Exempel på vanlig användning av stencilbuffern



Spegling är mer än en speglingsmatris (S(-1, 1, 1))

- Speglade objekt utanför den speglande ytan
- Ickeperfekt spegling

Möjliga ytterligare problem:

- Objekt bakom spegeln
- Objekt som skär spegeln
- Icke plana ytor (cube mapping!)

Algoritm

1) Radera stencilbuffern
2) Rita spegeln i stencilbuffern
3) Rita speglade objekt ”bakom spegeln” maskat med stencilbuffern
4) Rita spegeln (med lagom transparens)
5) Rita objekten framför spegeln

Varianter

Rita spegeln först, rita speglade objekt med Z-buffer avslagen.

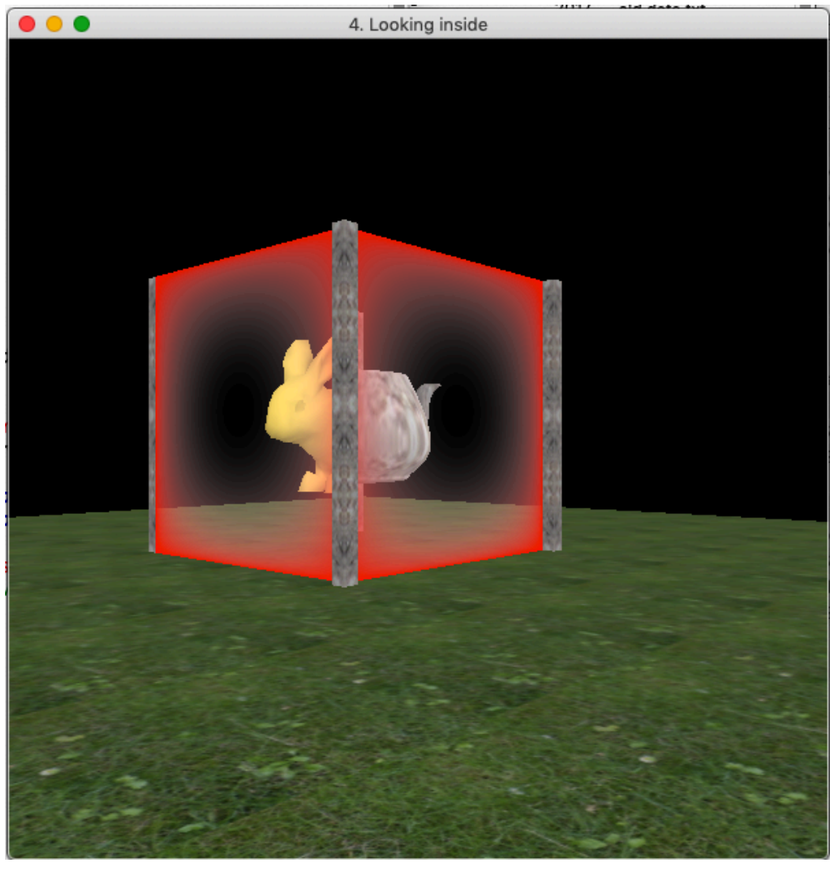
Clipping av objekt som skär spegeln. (Demoner som går genom speglande portaler...?)

Besläktad metod: Maskning av plana skuggor

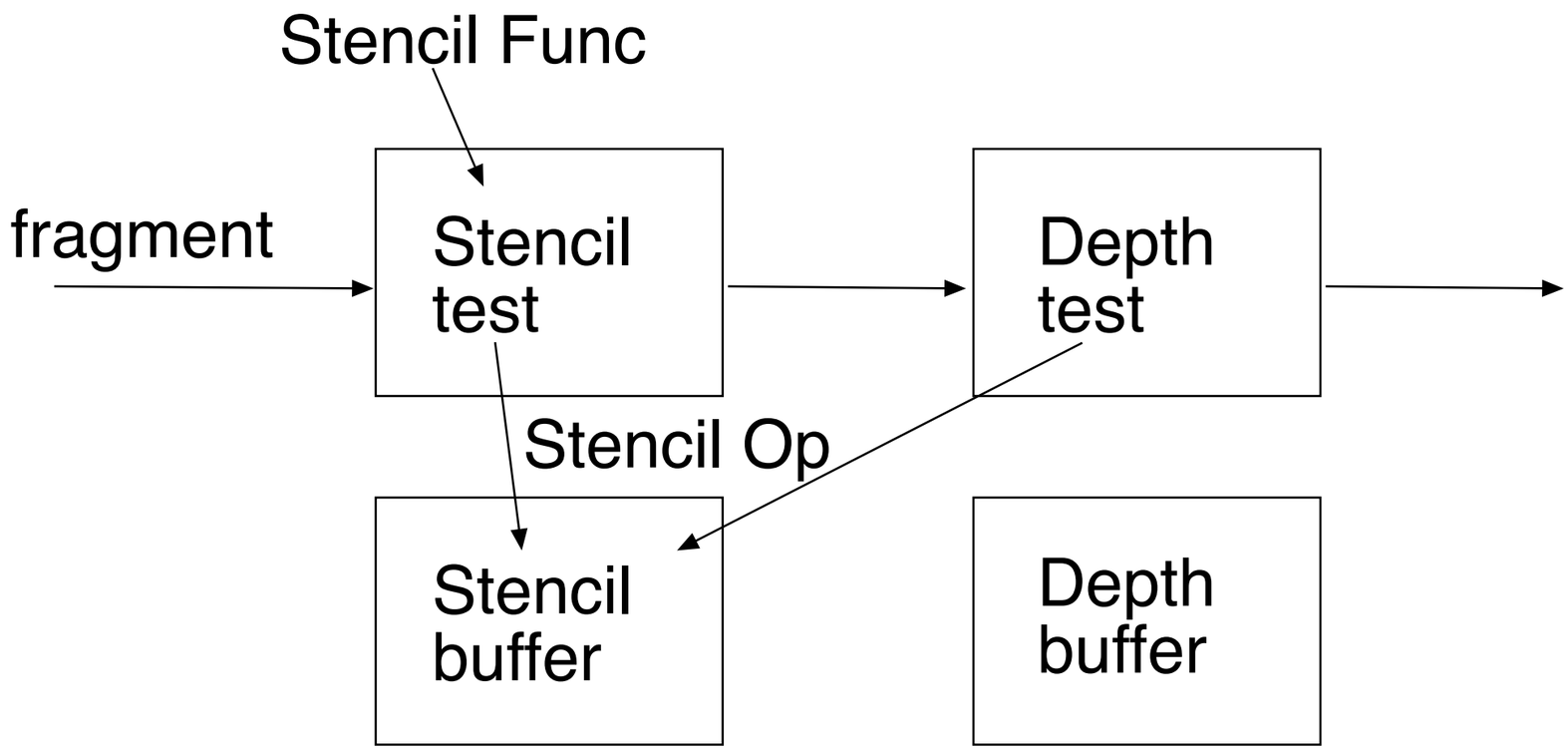
Kommer senare!

Stencilportaler

Stencilbuffern kan användas för att begränsa ritandet till en portal. (Andra metoder finns.)



Stencilbufferns delsteg



Sista operationerna före skrivning till frame buffer

func-parametern:

GL_ALWAYS: Default, avstängd, allt passerar.
GL_LESS, GL_LEQUAL, GL_GREATER, GL_GEQUAL, GL_EQUAL, GL_NOTEQUAL: Villkor, vad maskar eller ej?

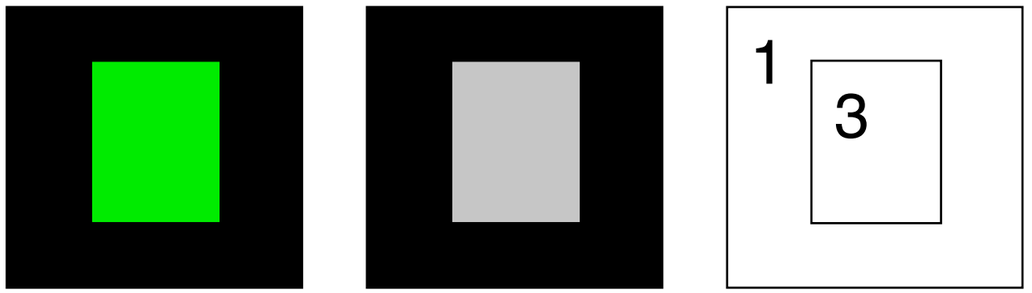
ref: Värden som jämförs med.

mask: Mask för logisk AND med ref och stencilvärde

Även: glStencilFuncSeparate.

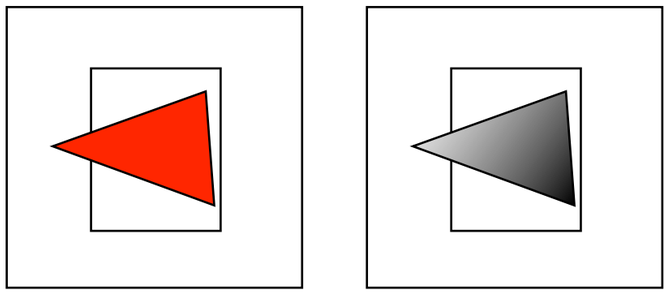
Stencilbuffer, exempel

Bild före:



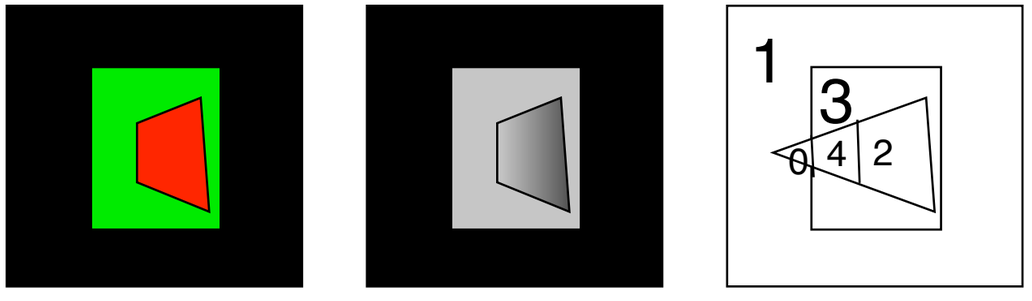
Frame buffer | Depth buffer | Stencil buffer

Rita triangel:



ref = 2
func = LESS
ops: fail: ZERO
zfail: INCR
zpass: REPLACE

Bild efter:



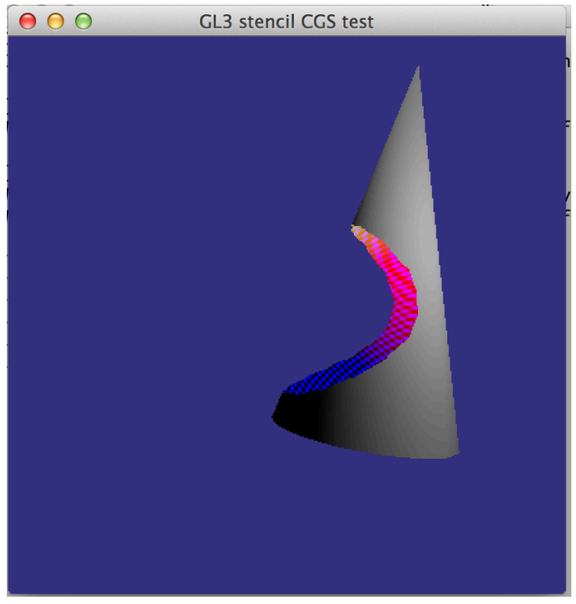
Frame buffer | Depth buffer | Stencil buffer

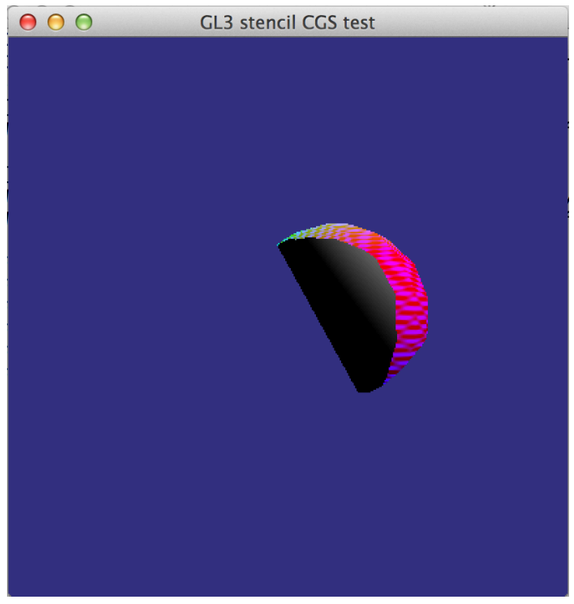
Exempel på tillämpning: Constructive Solid Geometry

Tillåter "and" och "or" samt subtrahering av ett objekt från ett annat.

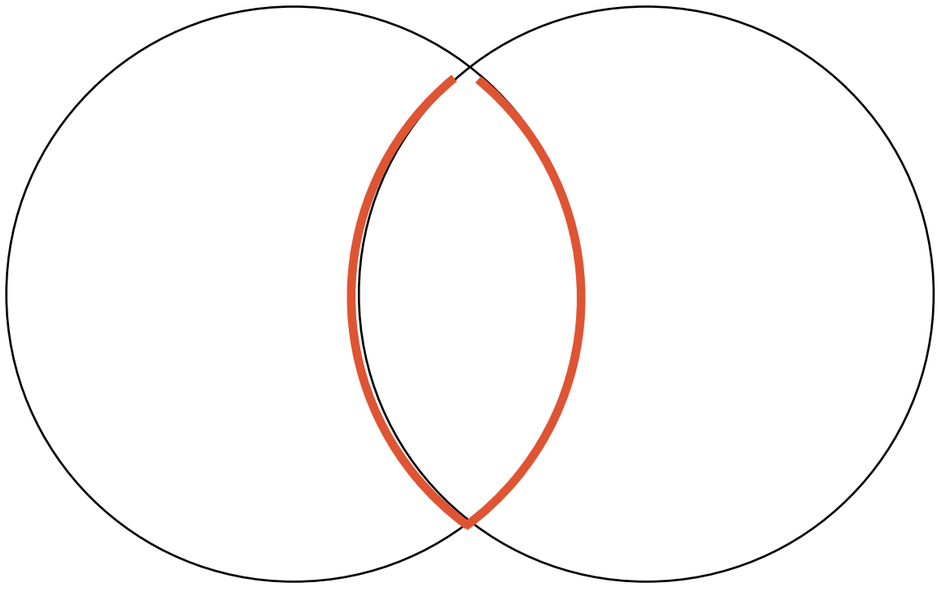
CSG med stencilbuffern

Räkna upp och ner för varje yta för att avgöra om vi är på in- eller utsidan.





Utnyttja Z-buffern och culling för att rendera rätt delar



Stencilbuffern, slutsatser

Inga mjuka val, på eller av
Kan fungera som binär mask
Kan fungera som räknare

En av grafikhårdvarans viktigaste komponenter
för ”smarta trick”.