

# Mer datorgrafik

Detaljer som vi hoppade över i grundkursen:

- **Buffrar**
- **Stencil buffer**
- **Accumulation buffer**
- **Mer texturmappning**
- **Rendera till textur, FBO**
- **Optimering, VBO**

Ingemar  
Ragnemalm  
ingis@isy.liu.se

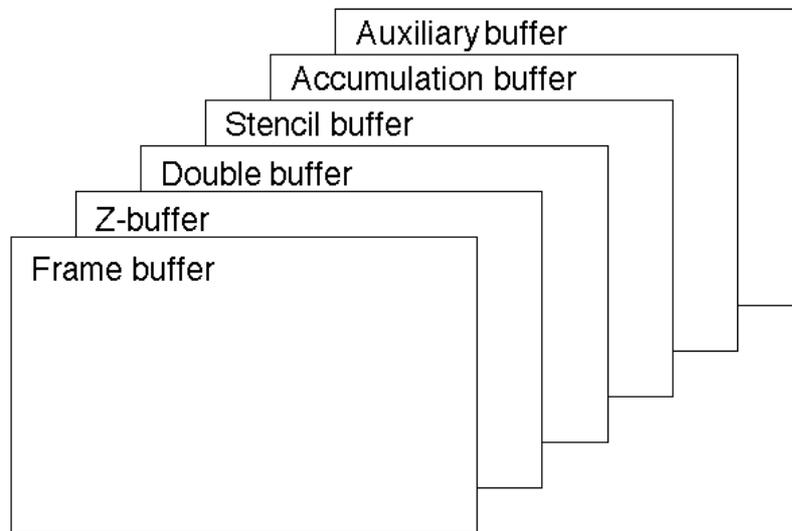
## Ytterligare grafik som följer längre fram:

- **Skuggor**
- **Multipass-shaders, faltning**
- **High dynamic range, blooming**
- **Bättre bump mapping, parallax mapping, displacement mapping**
- **Text**
- **Moln, hår, växter**
- **Ljusmodeller**

Ingemar  
Ragnemalm  
ingis@isy.liu.se

# Bildbuffrar

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



Ingemar  
Ragnemalm  
ingis@isy.liu.se

## De vanliga

**Frame buffer: Den vanliga bildbuffern**

**Dubbelbuffer (back-buffer): Kopia off-screen**

**Z-buffer (depth buffer): Lågnivå VSD**

Ingemar  
Ragnemalm  
ingis@isy.liu.se

# Ett par till

**Stencil buffer: Maskning**

**Accumulator buffer: Sammanvägning av bildrutor**

**Auxiliary buffers: Tillämpningsdefinierade**

**Frame Buffer Objects (FBO): Buffer som är textur**

Ingemar  
Ragnemalm  
ingis@isy.liu.se

## Buffer eller textur?

**Bufferoperationer kan ofta (alltid?) göras med shaders. Rendering till dem görs med FBO. Det verkar sannolikt att många bufferfunktioner kommer att lösas i shaders i framtiden.**

# Stencil buffer

“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.

Används ofta som om den var binär, men är heltal, t.ex. 8 eller 16 bitar (unsigned). Precisionen är implementationsberoende.

Många tillämpningar - vilka?

Ingemar  
Ragnemalm  
ingis@isy.liu.se

# Stencil buffer

Förslag på tillämpningar:

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

Viktigare än man först tror?

Ingemar  
Ragnemalm  
ingis@isy.liu.se

# Stencil buffer

I OpenGL:

**glStencilFunc(func, ref, mask);**  
bestämmer stencilbufferns funktion 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...)

Ingemar  
Ragnemalm  
ingis@isy.liu.se

## Enkelt exempel

**Masking enbart**

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

Ingemar  
Ragnemalm  
ingis@isy.liu.se

# Enkelt exempel

## Måste initiera OpenGL-context med stencilbuffer:

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

## Radera stencilbuffern

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

# Enkelt exempel

## Rita masken 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  
glutWireSphere(1.2, 32, 16);
```

# Enkelt exempel

## 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);

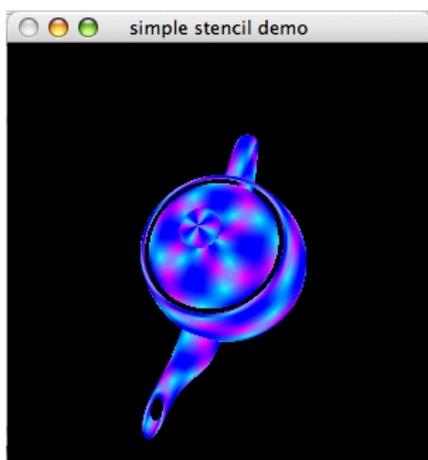
// Only write to the Stencil Buffer where 1 is set
glStencilFunc(GL_EQUAL, 1, 0xFFFFFFFF);
// Keep the content of the Stencil Buffer
glStencilOp(GL_KEEP, GL_KEEP, GL_KEEP);

// Draw the shape
DrawTheShape();

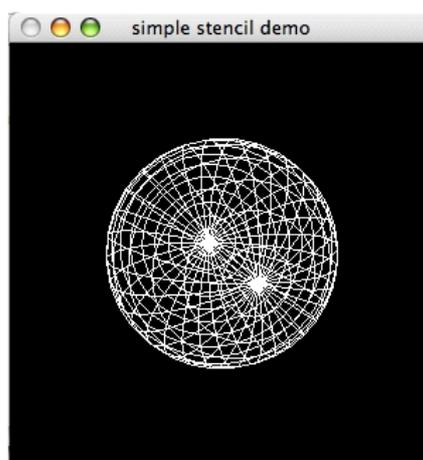
glDisable(GL_STENCIL_TEST); // Turn off
```

Ingemar  
Ragnemalm  
ingis@isy.liu.se

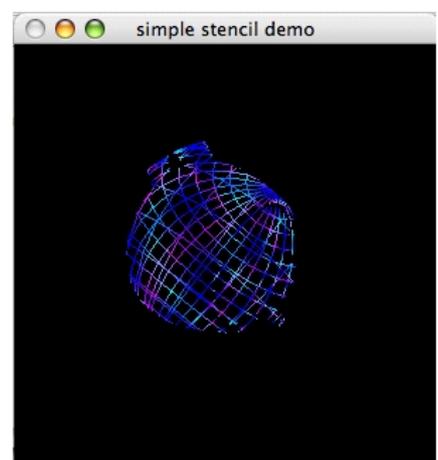
# Enkelt exempel



Objekt att maska



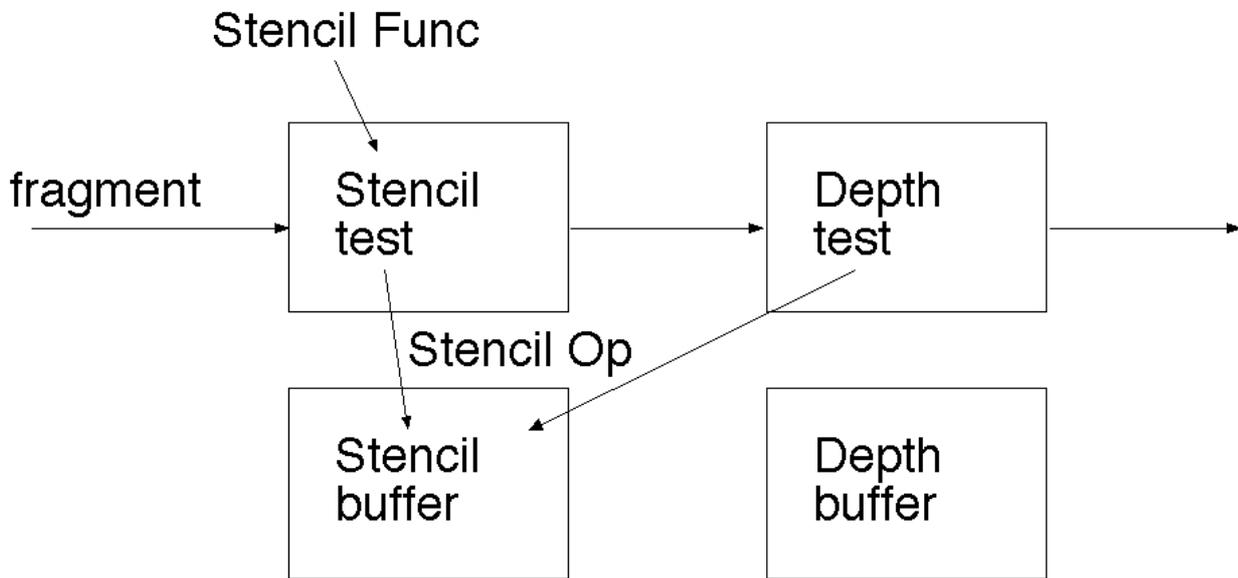
Mask



Resultat

Ingemar  
Ragnemalm  
ingis@isy.liu.se

# Stencil buffer

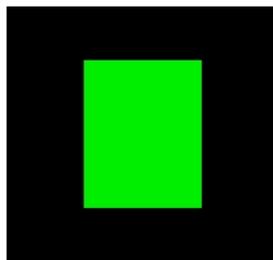


Sista operationerna före skrivning till frame buffer

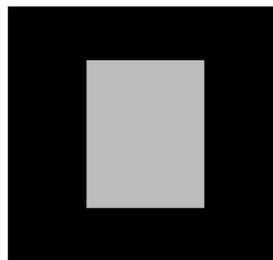
Ingemar  
Ragnemalm  
ingis@isy.liu.se

## Stencil buffer, exempel

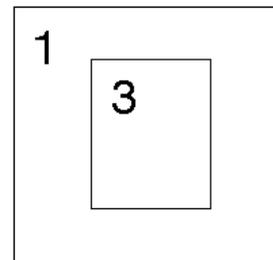
Bild  
före:



Frame buffer

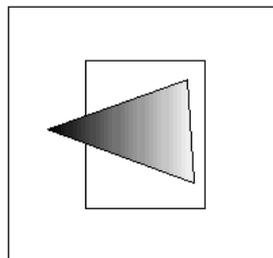
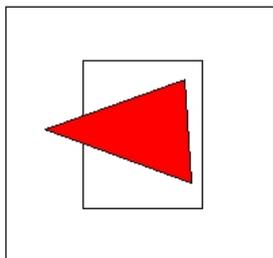


Depth buffer

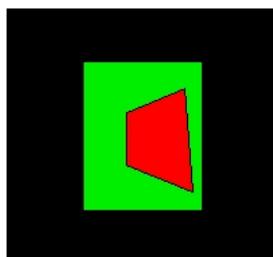


Stencil buffer

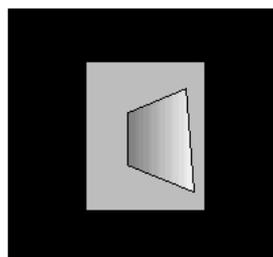
Rita  
triangel:



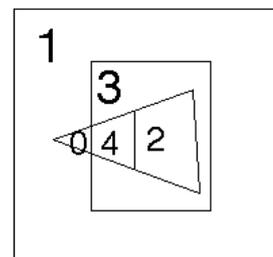
ref = 2  
func = LESS  
ops: fail: ZERO  
zfail: INCR  
zpass: REPLACE



Frame buffer



Depth buffer



Stencil buffer

Ingemar  
Ragnemalm  
ingis@isy.liu.se

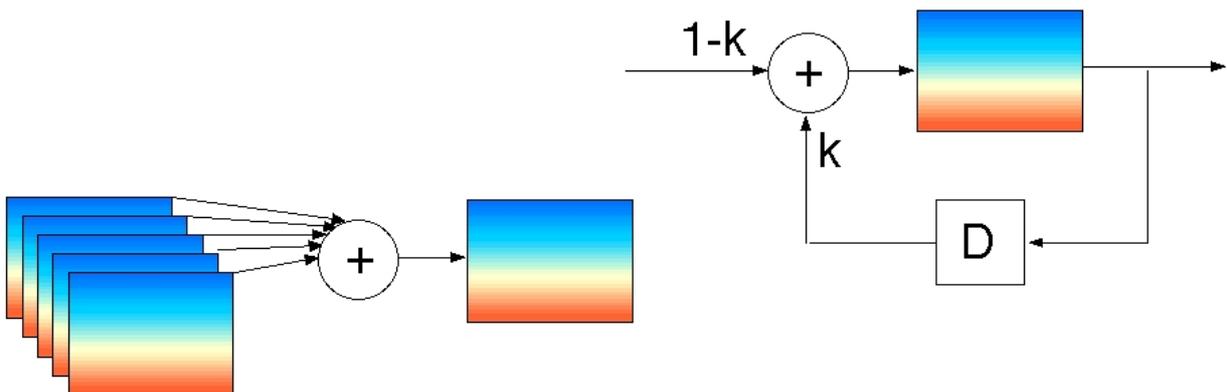
# Bufferoperationer

Radera: `glClear`  
Kopiera mellan buffrar  
Kopiera till och från CPU  
Välj buffer att rita i  
Välj buffer att läsa från

Ingemar  
Ragnemalm  
ingis@isy.liu.se

## Accumulation buffer

En buffer för att väga samman, ackumulera,  
flera frames!



Ingemar  
Ragnemalm  
ingis@isy.liu.se

# Accumulation buffer

glAccum(op, value);

GL\_ACCUM:  $A = A + FB * value$

GL\_LOAD:  $A = FB * value$

GL\_RETURN:  $FB = A$

GL\_ADD:  $A = A + value$

GL\_MULT:  $A = A * value$

Behagligt enkelt gränssnitt!

A = accumulation buffer

FB = frame buffer, vald ritbuffer

Ingemar  
Ragnemalm  
ingis@isy.liu.se

# Accumulation buffer

1) Rita framebuffer flera gånger med någon vald skillnad, ackumulera till en slutlig sammanvägd bild. Flera “exponeringar” vid samma tid.

2) Behåll gammalt resultat, ackumulera med nya frames. Flera “exponeringar” vid olika tid.

Många möjligheter!

Ingemar  
Ragnemalm  
ingis@isy.liu.se

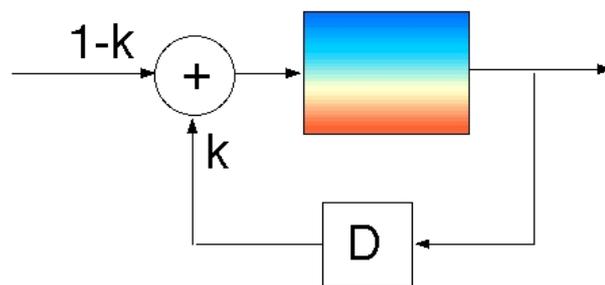
# Accumulation buffer

- Rörelseoskärpa, temporalfiltrering
- Anti-aliasing
- Skärpedjupseffekter
- Faltning

Allt som man gör med jittering i strålföljning kan göras med ackumulatorbuffern - men det kostar!

Ingemar  
Ragnemalm  
ingis@isy.liu.se

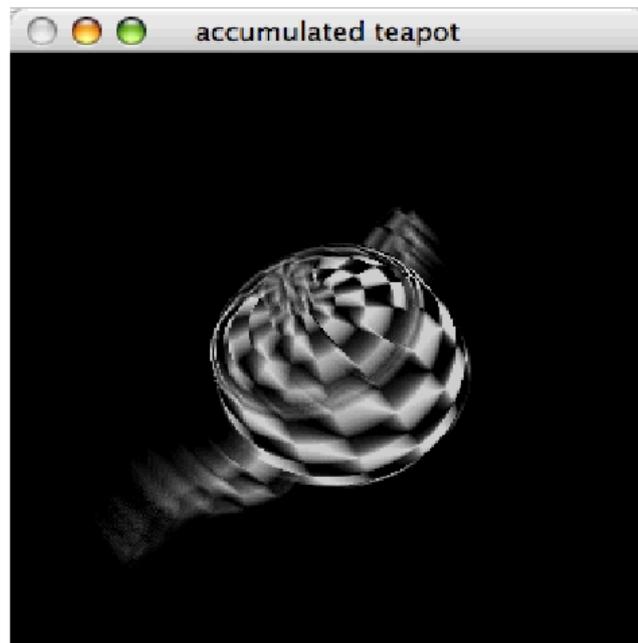
## Rörelseoskärpa, temporalfiltrering



```
// Temporalfilter med glAccum!  
glAccum(GL_MULT, 0.6);  
glAccum(GL_ACCUM, 0.4);  
glAccum(GL_RETURN, 1.0);
```

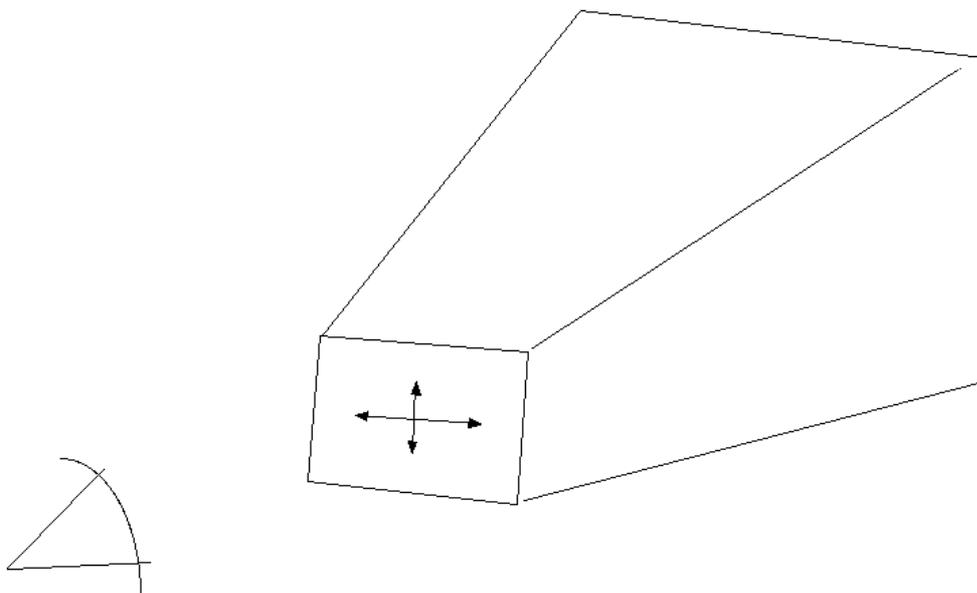
Ingemar  
Ragnemalm  
ingis@isy.liu.se

# Rörelseoskärpa



## Anti-aliasing med ackumulatorbuffer

Flytta frustum inom en pixel



**Slumpmässigt bäst; aliasing dränks i brus, men ackumulatorbuffer kan inte ge bra brus.**

# Nackdel med accumulation buffer

Ofta dåligt stöd i konsument-GPUer! Utförs ofta i  
ljukvara – långsamt!

Egen implementation med hjälp av shaders kan  
behövas – men är ganska lätt!

Ingemar  
Ragnemalm  
ingis@isy.liu.se

## Bättre anti-aliasing

OpenGL har inbyggt stöd för anti-aliasing.

Anti-aliasing per primitiv. Dåligt, stöds inte av alla  
GPUer.

Bättre: Full-screen anti-aliasing (FSAA)

Initiera glut med GLUT\_MULTISAMPLE  
`glEnable(GL_MULTISAMPLE);`

Ännu en lösning: FSAA med FBO.

Ingemar  
Ragnemalm  
ingis@isy.liu.se

# Texturmapping

I TSBK05/07: 2D-texturer för materialytor, koordinater gavs manuellt eller med glTexGen.

Låt oss gå lite längre:

- Fler dimensioner
- Optimering
- Multipass
- Texturplacering
- Multitexturering