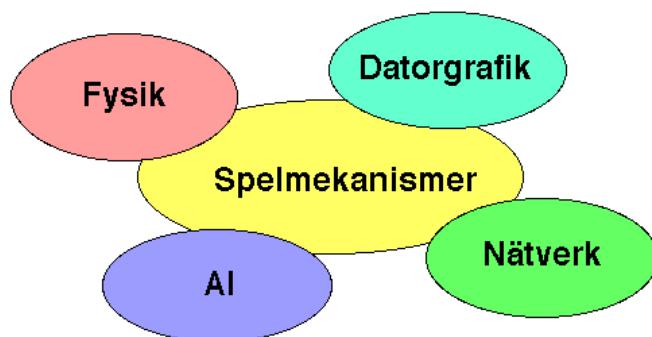


TSBK 10

Teknik för avancerade datorspel

Ingemar Ragnemalm, ISY



Ingemar
Ragnemalm
ingis@isy.liu.se

Föreläsning 2

- Lite om enkäten
- Buffrar
- Evaluators
- Texturmappning

Ingemar
Ragnemalm
ingis@isy.liu.se

Enkäten

Betryggande förkunskaper, och inte så överdrivet höga att vi inte kan lära er något.

Trevlig fördelning av kompetensområden och spetskunskaper

Önskemål om bl.a. shaderprogrammering. 

Ingemar
Ragnemalm
ingis@isy.liu.se

Förkunskaper

Spelprogrammering: Övervägande “medel”.

Datorgrafik och AI: Övervägande ganska hög

Fysik: Måttligt till ganska hög

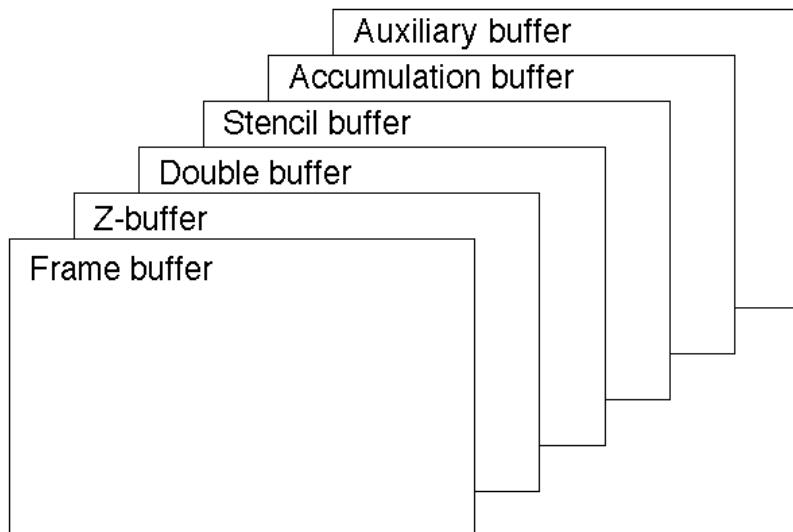
Nätverk: Övervägande måttligt

Projektfunderingarna välspridda över delämnen!

Ingemar
Ragnemalm
ingis@isy.liu.se

Buffrar

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

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

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

Viktigare än man först tror?

Ingemar
Ragnemalm
ingis@isy.liu.se

Stencil buffer

I OpenGL:

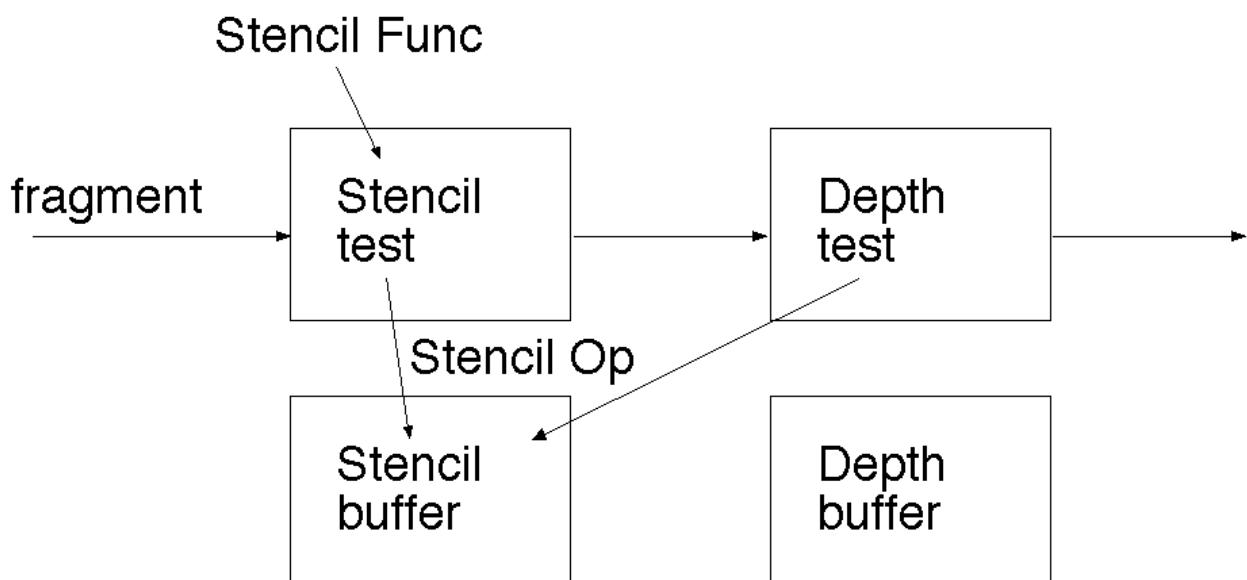
`glStencilFunc(func, ref, mask);`
bestämmer stencilbufferns funktion under ritande

`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

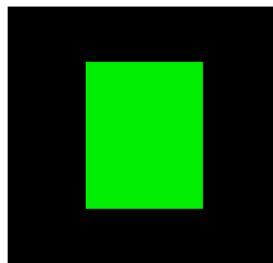
Stencil buffer



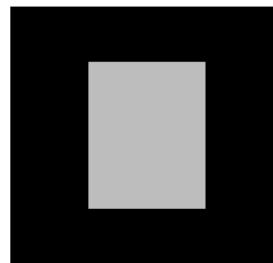
Ingemar
Ragnemalm
ingis@isy.liu.se

Stencil buffer, exempel

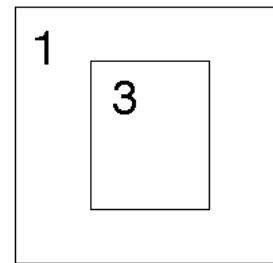
Bild före:



Frame buffer

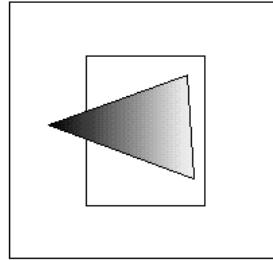
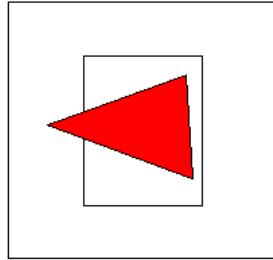


Depth buffer

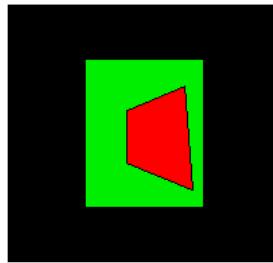


Stencil buffer

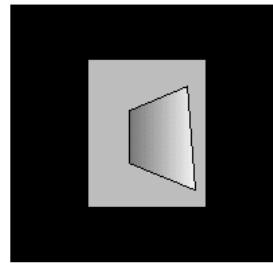
Rita triangel:



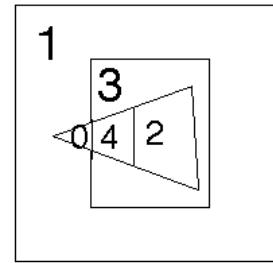
ref = 2
func = LESS
ops: fail: ZERO
zfail: INCR
zpass: REPLACE



Frame buffer



Depth buffer

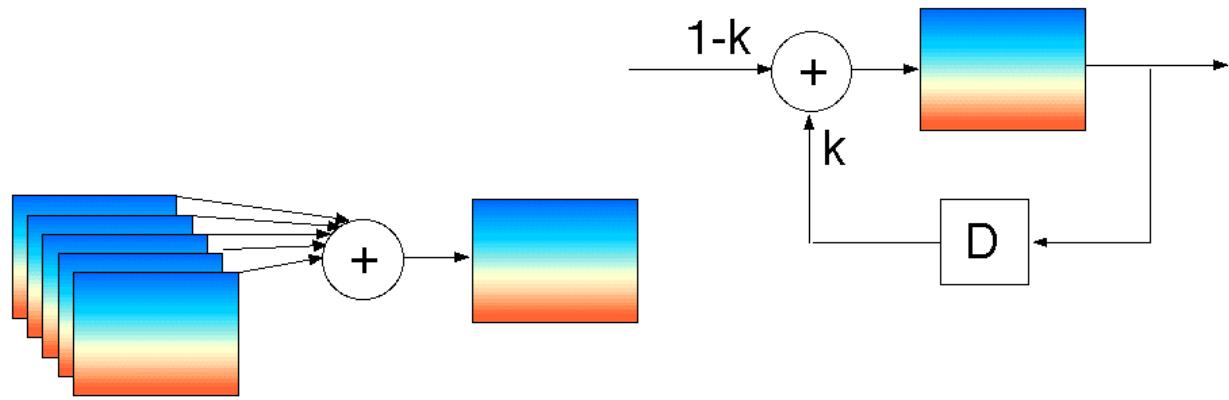


Stencil buffer

Ingemar
Ragnemalm
ingis@isy.liu.se

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

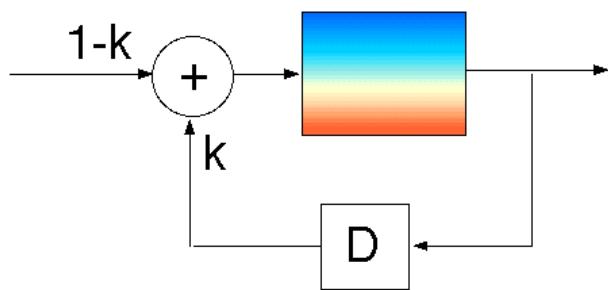
Accumulation buffer

- 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

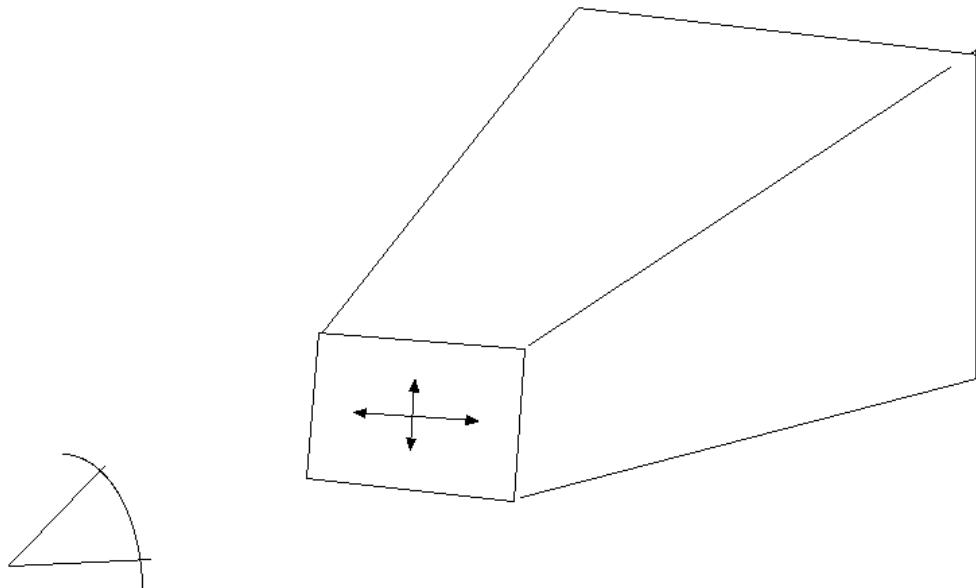


```
// Temporalfilter med glAccum!
glAccum(GL_MULT, 0.6);
glAccum(GL_ACCUM, 0.4);
glAccum(GL_RETURN, 1.0);
```

Ingemar
Ragnemalm
ingis@isy.liu.se

Anti-aliasing med ackumulatorbuffer

Flytta frustum inom en pixel

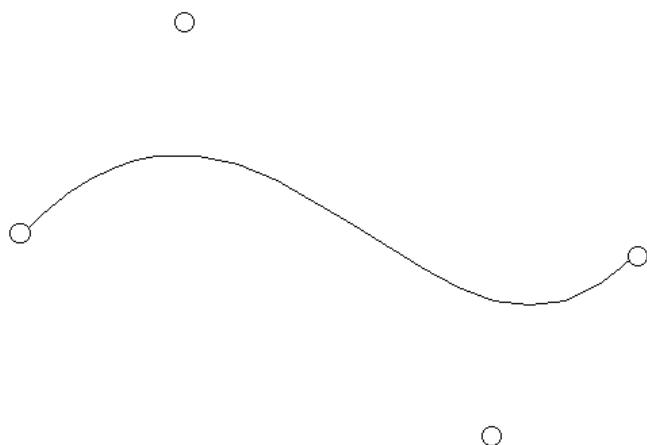


Slumpmässigt bäst; aliasing dränks i brus

Ingemar
Ragnemalm
ingis@isy.liu.se

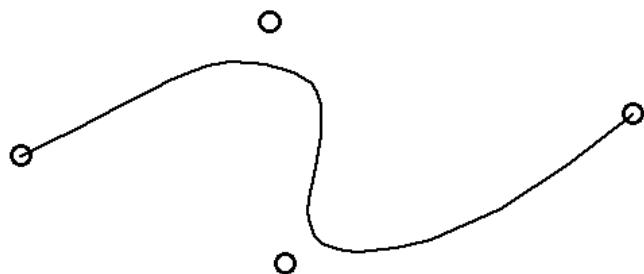
Evaluators

En inbygggd funktion för att hjälpa dig att rita Bezierkurvor. Beräknar kurvan för önskad täthet, och skapar linjesegment.

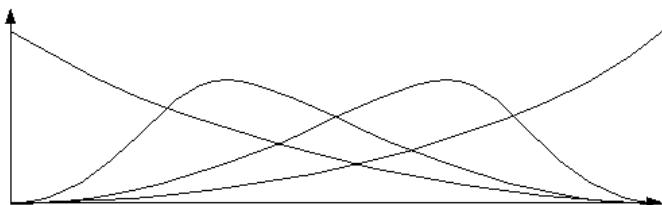


Ingemar
Ragnemalm
ingis@isy.liu.se

Bézierkurvor



Typiskt 4 kontrollpunkter per segment.



$$\begin{aligned} \text{BEZ}_{0,3} &= (1-u)^3 \\ \text{BEZ}_{1,3} &= 3u(1-u)^2u \\ \text{BEZ}_{2,3} &= 3(1-u)u^2 \\ \text{BEZ}_{3,3} &= u^3 \end{aligned}$$

Vägs samman med 4 viktfunktioner
(Bernstein-polynom)

Ingemar
Ragnemalm
ingis@isy.liu.se

Evaluators

Ställs in med glMap

Evalueras element för element med glEvalCoord

eller på en gång med glEvalMesh

```
glMap1f(GL_MAP1_VERTEX_3, u0, u1, 3, 4, &data2[0][0]);
 glEnable(GL_MAP1_VERTEX_3);
 glBegin(GL_LINE_STRIP);
 for (int i = 0; i <= 20; i++)
    glEvalCoord1f(u0 + i*(u1-u0)/20);
 glEnd();
```

Kontrollpunkter

Evaluering, gör
glVertex

Evaluators

glEvalMesh praktiskt om jämna steg duger:

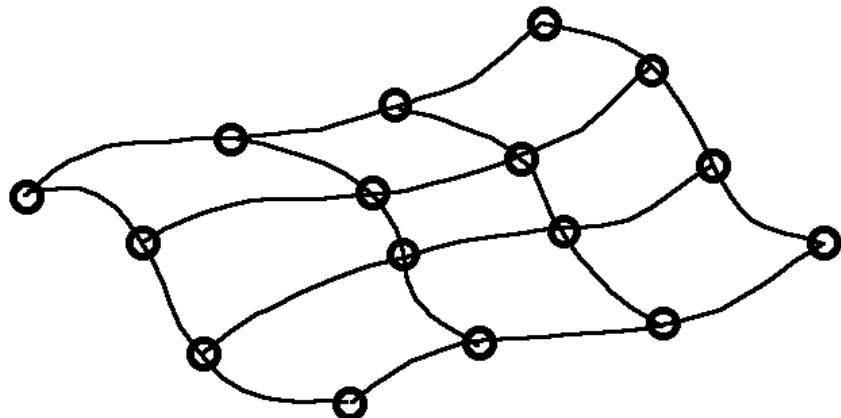
```
glMap1f(GL_MAP1_VERTEX_3, u0, u1, 3, 4, &data2[0][0]);  
glEnable(GL_MAP1_VERTEX_3);  
glMapGrid1f(20, 0, 1); ← Vilket intervall tänker  
glEvalMesh1(GL_LINE, 0, 20); vi jobba med?  
  
Hela loopen i ett anrop  
(nåja, två)
```

Ingemar
Ragnemalm
ingis@isy.liu.se

Evaluators

Samma sak i 2D!

```
glMap2f(GL_MAP2_VERTEX_3, 0, 1, 3, 4, 0, 1, 12, 4, &data2d[0][0][0]);  
glEnable(GL_MAP2_VERTEX_3);  
glMapGrid2f(20, 0, 1, 20, 0, 1);  
glEvalMesh2(GL_FILL, 0, 20, 0, 20);
```



Ingemar
Ragnemalm
ingis@isy.liu.se

Evaluators

Uppenbart bra för att modellera runda former.

Lätt att ha level-of-detail

Bra för t.ex. tygmodellering.

Kan ha beräkningsfördelar genom att GPU'n kan avlasta CPU'n.