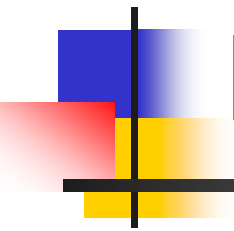
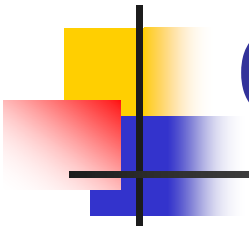


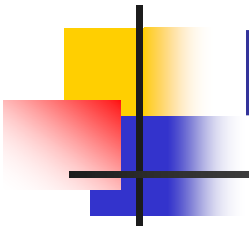
Designing a Shading System

David Larsson



Överblick

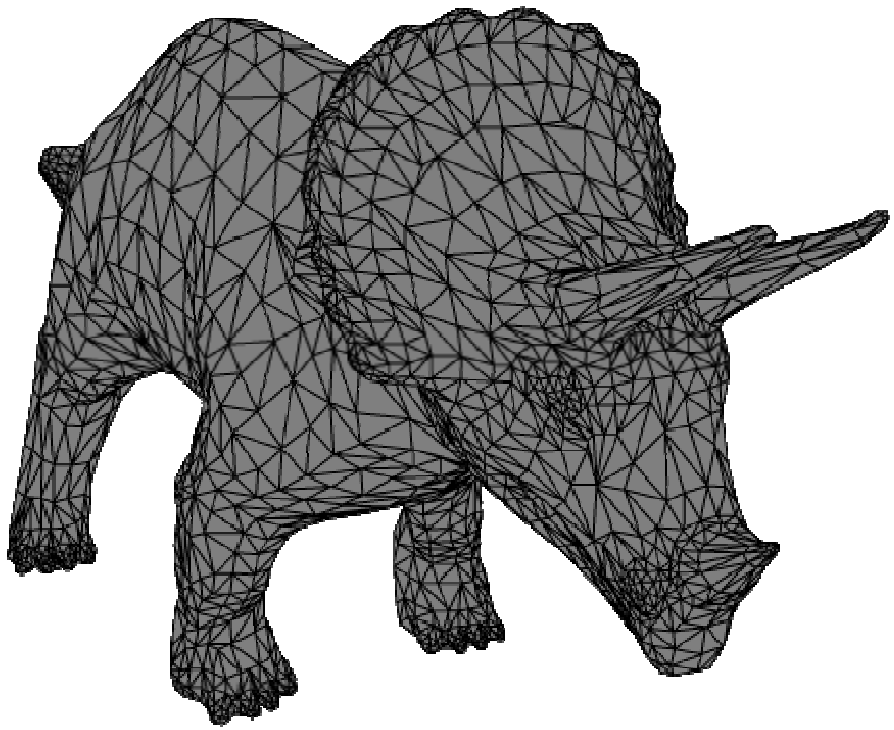
- Genomgång av rendering och shading
- Designval
- Implementationsdetaljer



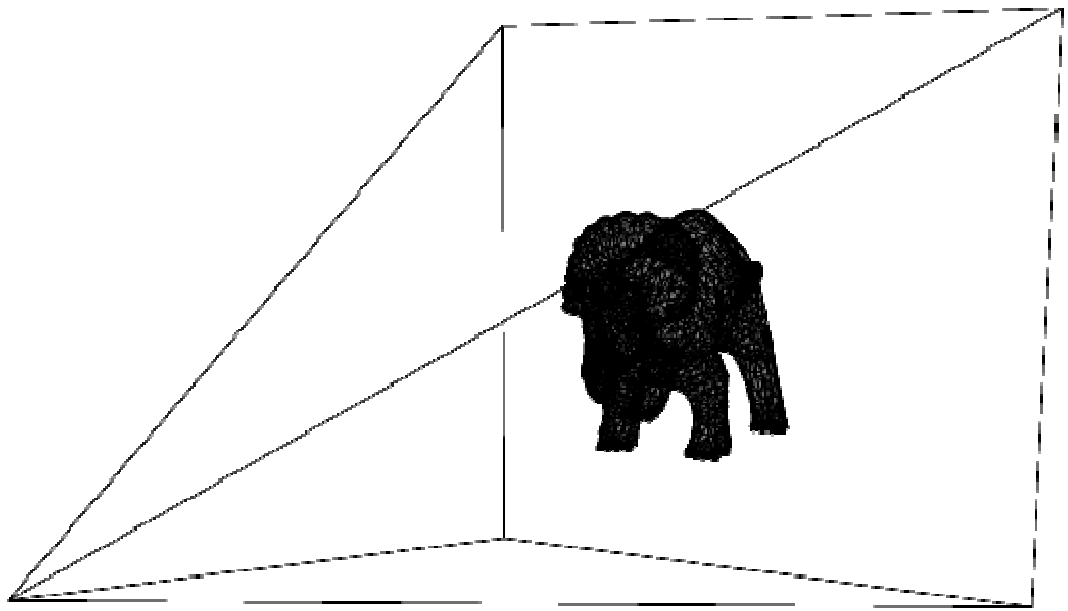
Rendering

- Omvandla en konceptuell 3d-värld till en bild

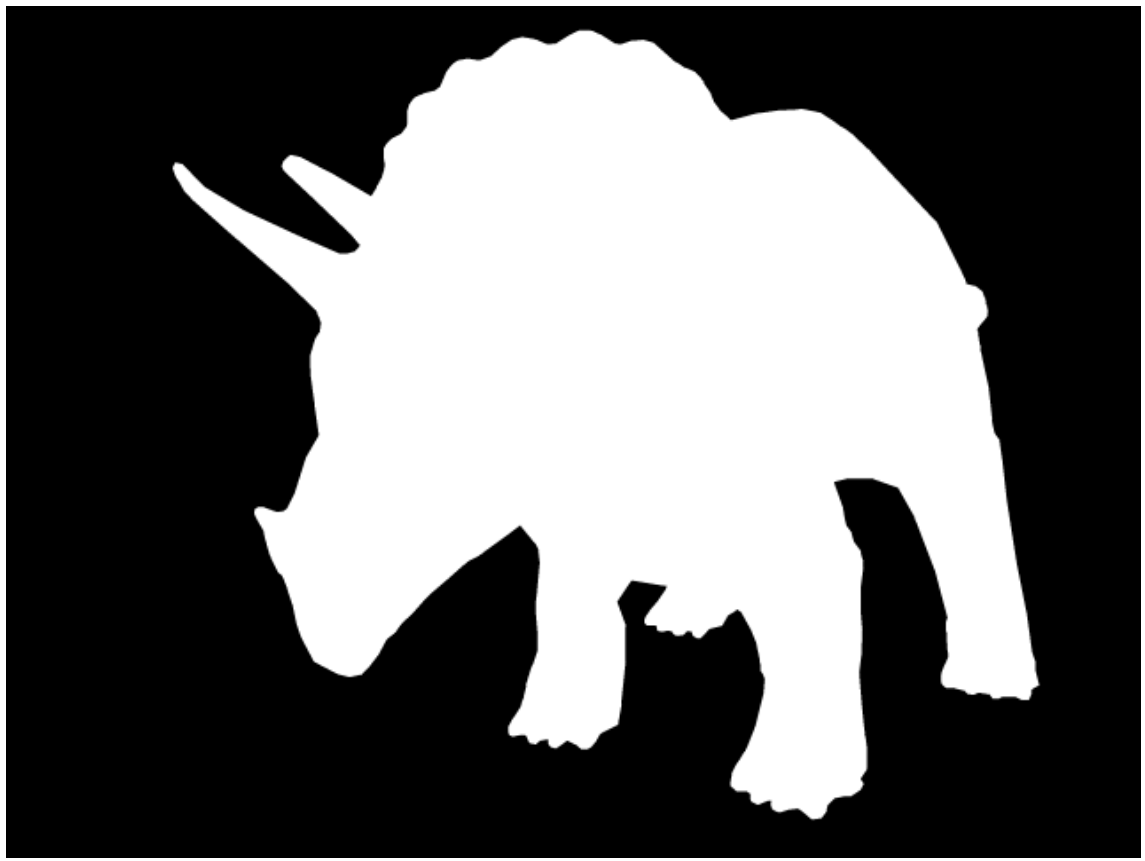
Geometri



Kamera

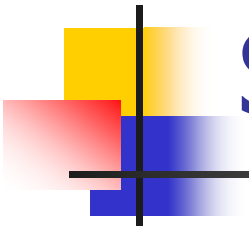


Något saknas?



Något saknas?

- Belysning
- Materialbeskrivningar

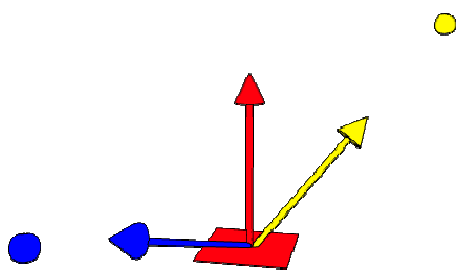


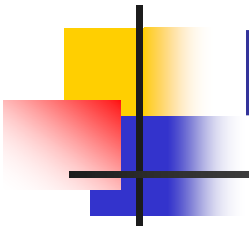
Shading

- Stort begrepp
- Traditionellt sett ljus- och materialinteraktion
- Används ofta om andra saker än material (partiklar, kameror mm)
- I resten av presentationen betyder shading enbart material och belysning

Indata för Shading

- Material
- Fragmentets position i världen
- Ytnormal
- Riktning till belysning
- Riktning till betraktaren
- Annat spännande
- Allt detta tillsammans kallas ytfragment

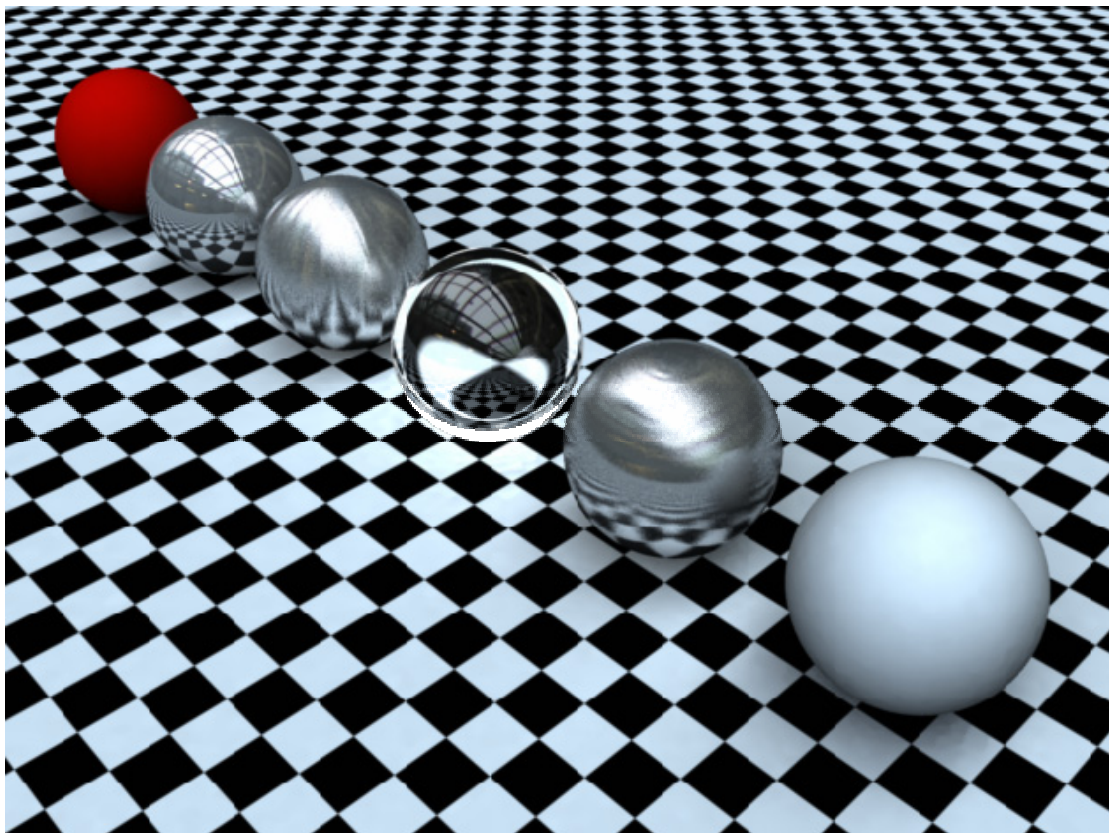




Materialmodeller

- Finns modeller för många olika sammanhang.
- Allt från väldigt enkla till extremt komplexa
- Diffusa, reflektiva, anisotropiska mm.
- Även icke-fysikaliska som ger exempelvis ”tecknad film”-utseende

Materialmodeller

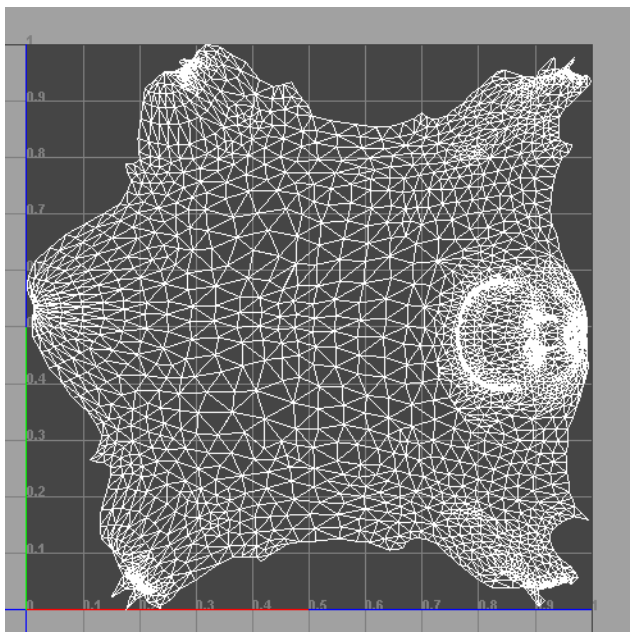


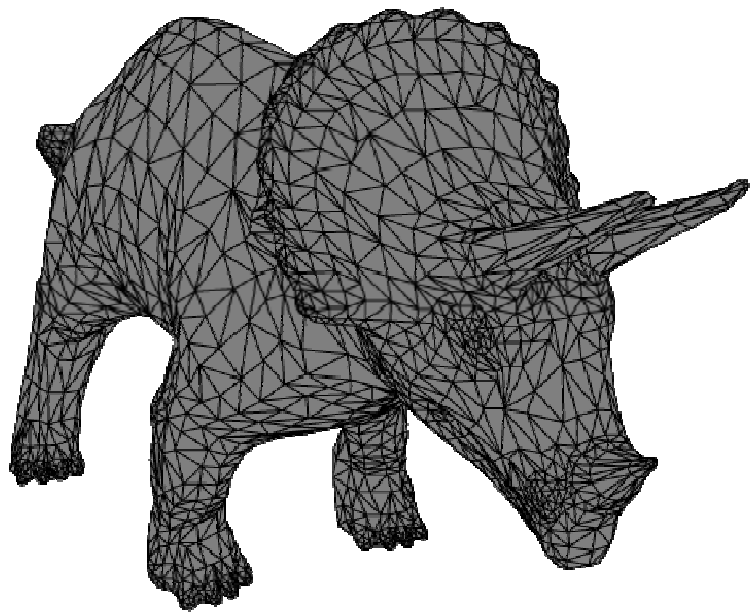
Texturering

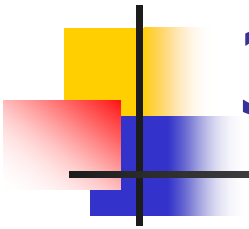
- Material är sällan homogena

Textureringskoordinater

- Platta ut föremålets yta på ett plan



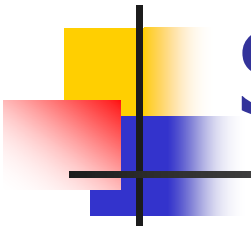




3D-texturering

- Bra för material med djup
- Marmor, trä mm
- Ofta procedurella, svårt att fotografera, enorma datamängder

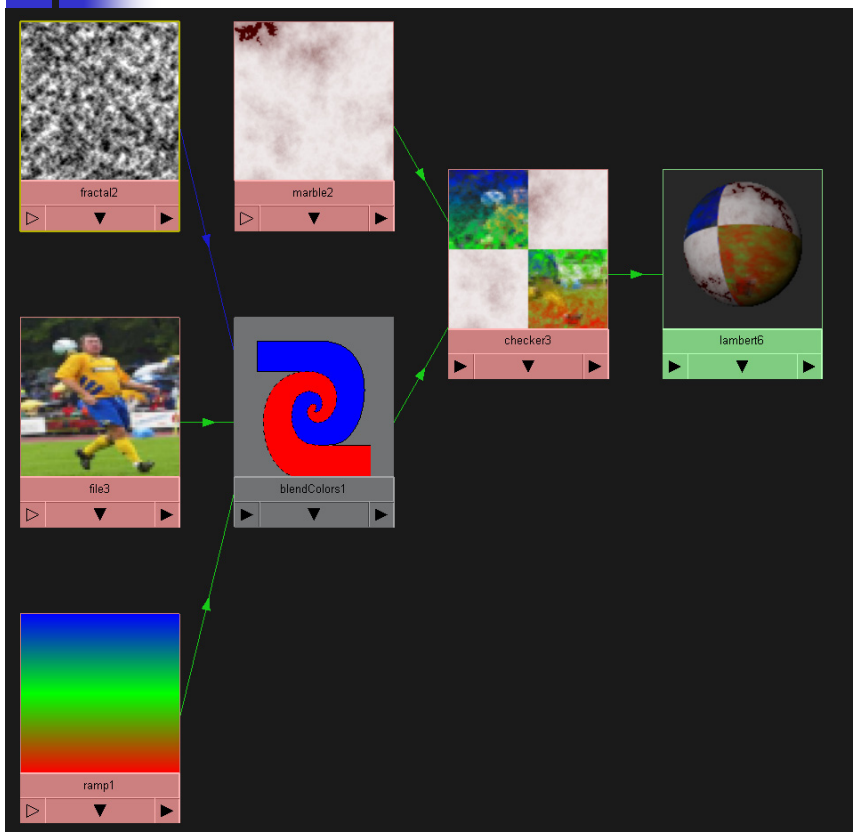
3D-texturering

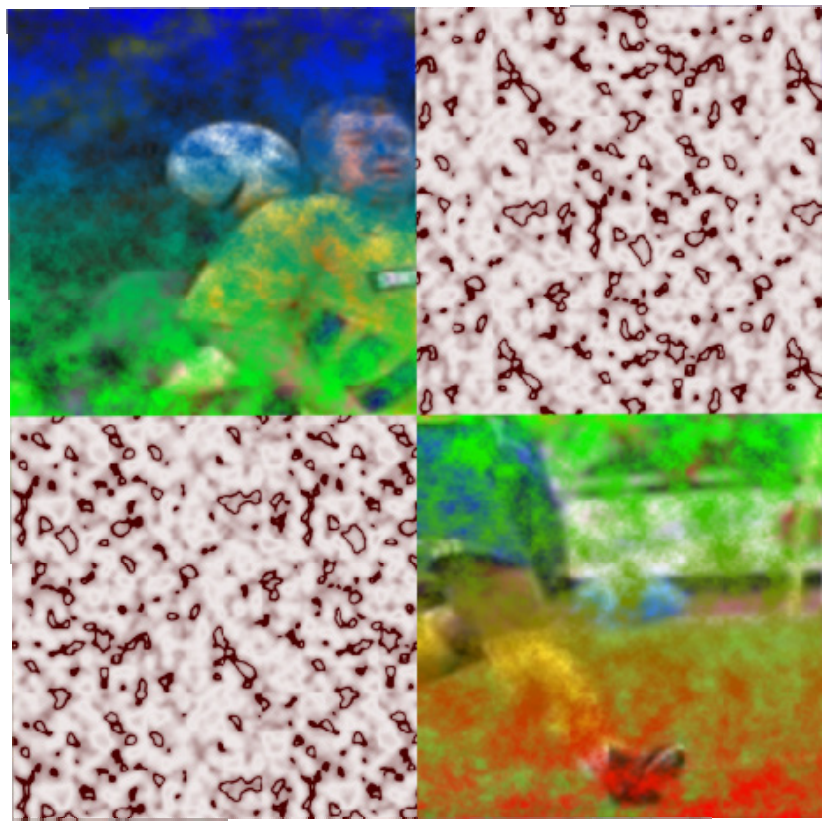


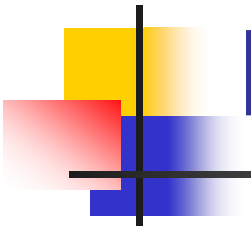
Shading

- Brukade vara fix modell (tänk gamla 3dgrafikkort, gammal opengl-belysning)
- Behöver mer inställningsmöjligheter än andra parametrar i en renderare.

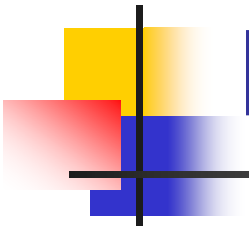
Exempel


fractal2
marble2
checker3
lambert6
file3
blendColors1
ramp1



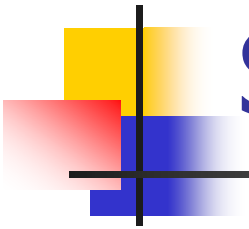


Designval



Exjobbet

- Designa ett shadingsystem för Turtle, en renderare till Maya
- Resultat som liknar Mayas
- Bra prestanda
- Trådsäkert
- Enkelt att göra nya shaders



Shadinglösningar

- Fix modell
- Shading-programspråk
- Träd- och byggklossmodeller

Fix modell

- Oflexibelt
- Slöseri med prestanda

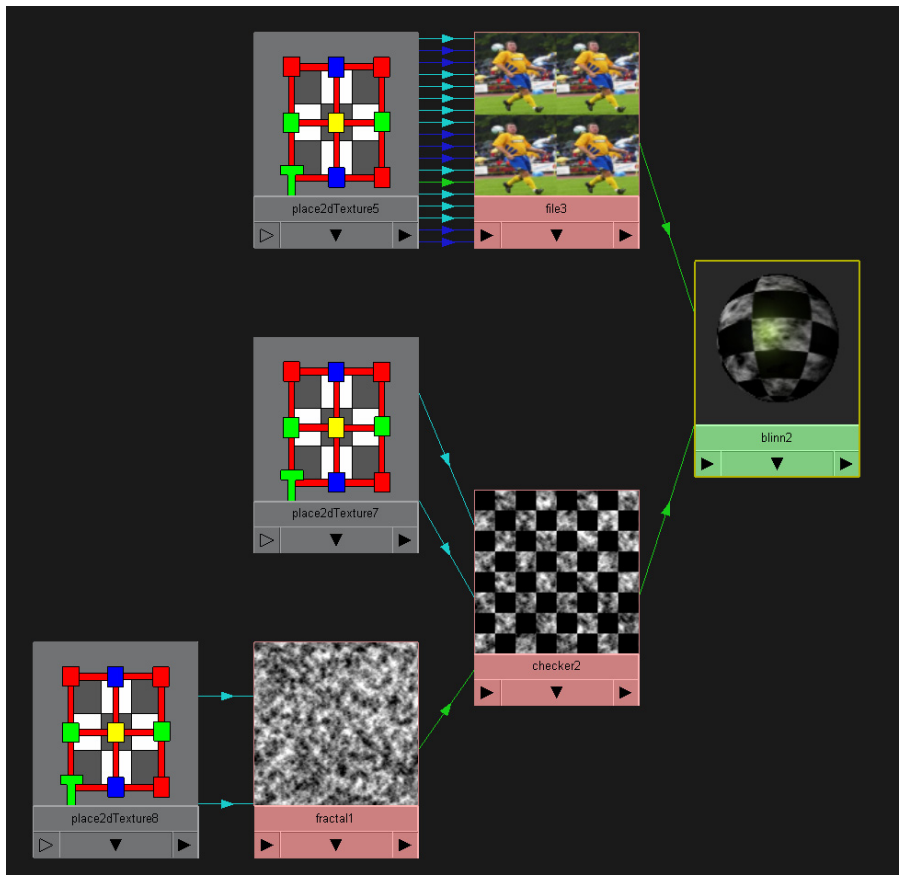
Programspråkslösning

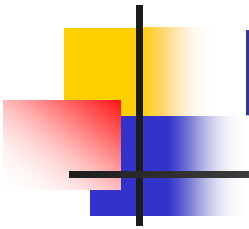
```
surface
plastic( float Ks=.5, Kd=.5, Ka=1, roughness=.1; color
specularcolor=1 )
{
        point Nf, V;
        Nf = faceforward( normalize(N), I );
        V = -normalize(I);
        Oi = Os;
        Ci = Os * ( Cs * (Ka*ambient() + Kd*diffuse(Nf)) +
        specularcolor * Ks * specular(Nf,V,roughness) );
}
```

Programspråkslösning

- Inarbetat (Renderman shading language)
- Svårt för ickeprogrammerare
- Kan användas som back-end till Byggkloss-modell
- Kräver egen kompilator/interpretator

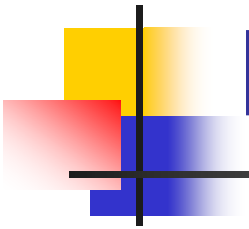
Byggklossmodell





Byggklossmodell

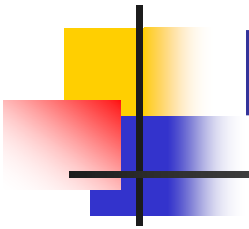
- Enkelt för artister
- Enkelt att stödja Mayas shaders
- Viss overhead



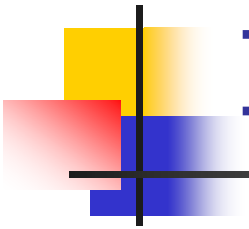
Byggklossar eller språk?

- Måste stödja byggklossar i slutet ändå
- Bygga eget språk tar tid och ger ofta undermåligt resultat
- Använda färdigt språk (lua, python etc.) ger minskad kontroll
- Kompilering i efterhand öppnar optimeringsmöjligheter

Byggkloss-lösning i C++



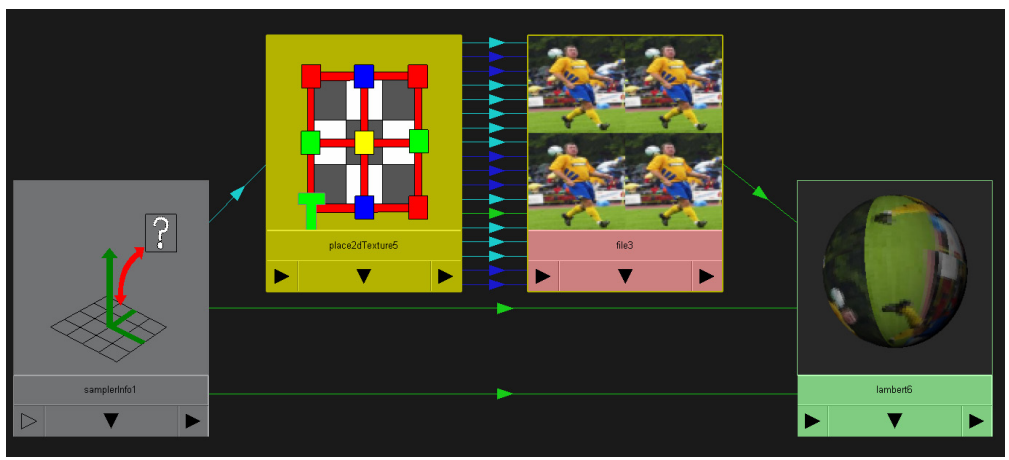
- C++ är bekant och pålitligt
- Inga obehagliga överraskningar
- Enkelt för mig och resten av folket
- Statiskt
- Inget hindrar att en byggkloss exekverar ett programspråk



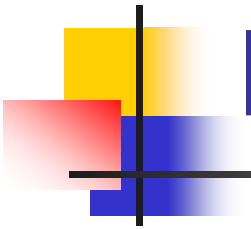
Implementationsdetaljer

Indata

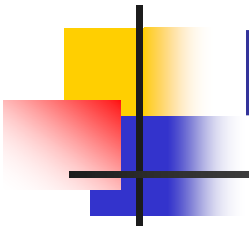
- Fragmentdatan kopplas automatiskt in



Laddande av nätverk

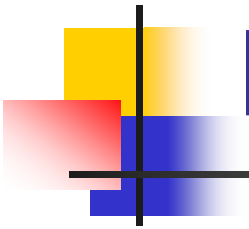


- Laddning – konvertera mayas nätverk till eget format. Det enda mayaspecifika i de bästa av världar.
- Länkning – omvandla textreferenser till riktiga referenser för prestanda
- Post-länkning – förberäkna, propagera konstanter mm.



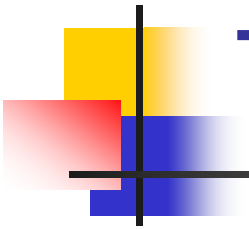
Exekvering av nätverk

- Overhead i att byta shader
- Bättre kod- och data-cache prestanda
- Kan öka prestandan genom att spara undan och sortera fragment
- Kan SIMD-exekvera



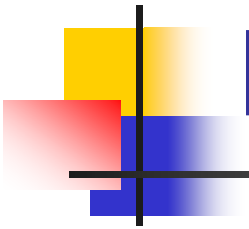
Exekvering av nätverk

- Problem med sekundärstrålar i raytracing
- Reflektioner och liknande går att lösa delvis, men adaptiv sampling är svårt



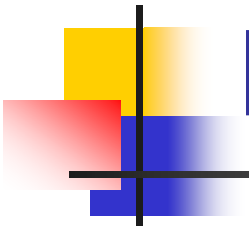
Trådningsproblem

- Många fällor
- Akta sig för statisk data
- Minnesallokeringar är dyra, använd stacken och trådspecifika minnespooler av olika slag
- Farligt att ”cacha” värden i nätverken



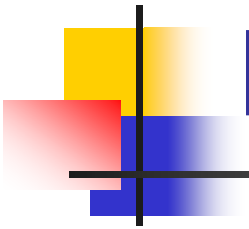
Resultat

- Svårt att jämföra
- Mycket spelar in (strålskickande, texturhantering, kvalitet på koden i noderna)
- Overhead i dataskickande mellan klossar går att mäta
- Står sig bra i de tester jag gjort (resultat i rapporten)



Resultat

- Skalar bra
- Enkelt att skapa nya shaders
- Emulerar Maya bra (med lite hjälp från Alias)
- Används i produktion



Framtida arbete

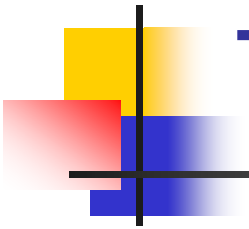
- Smarta cachar
- Automatisk konstantpropagering
- Smartare hantering av implicit fragmentdata (inte beräkna saker som aldrig används)
- Titta på andra anropskonventioner
- SIMD-shading med sekundärstrålar

Resultat

Resultat

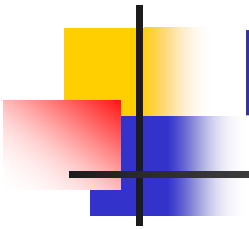
Resultat

Resultat



Tack till

- Illuminate Labs
- Folket bakom bilderna (Engine Room, Vetor Zero, Izmeth Siddek, E3DI, Mats)



Frågor?