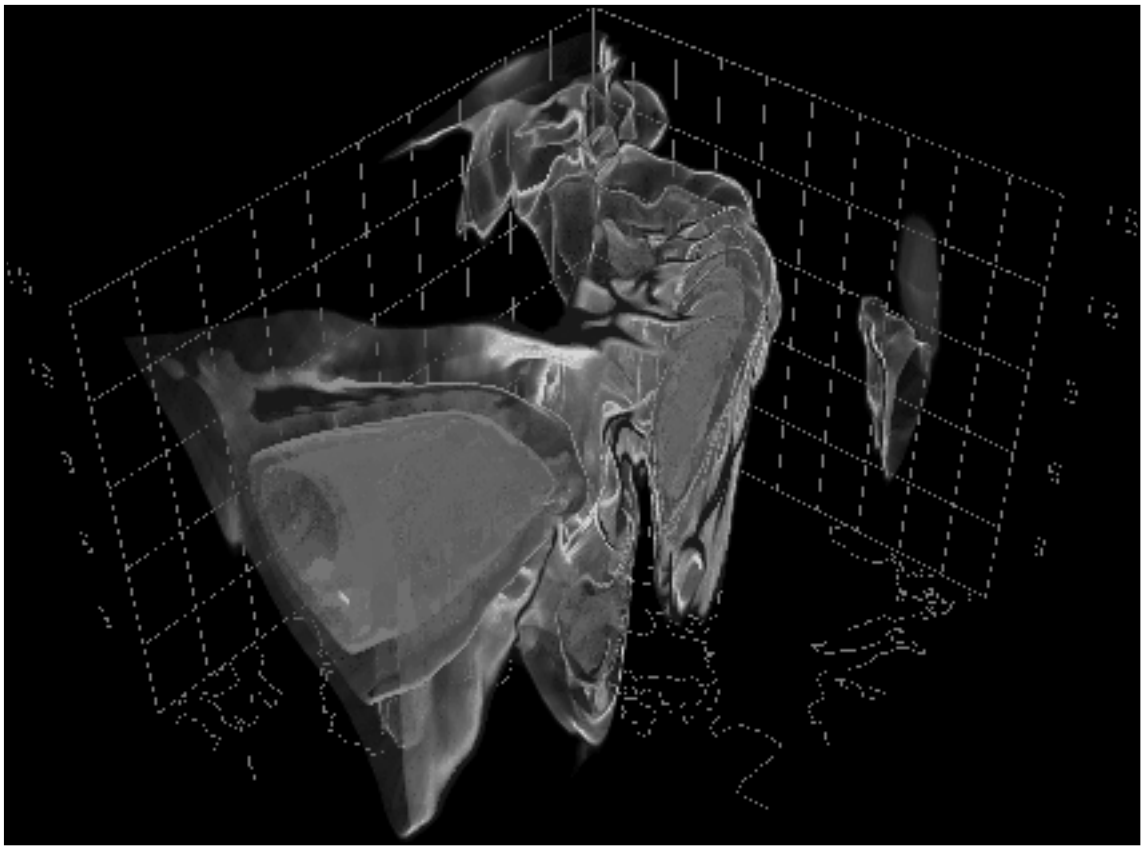
NORSIGD INFO

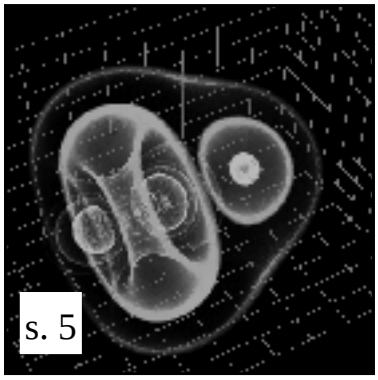
Nummer 3 1994



NORSK SAMARBEID INNEN GRAFISK DATABEHANDLING

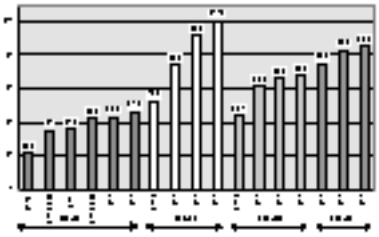
ISSN 0803-8317

Innhold

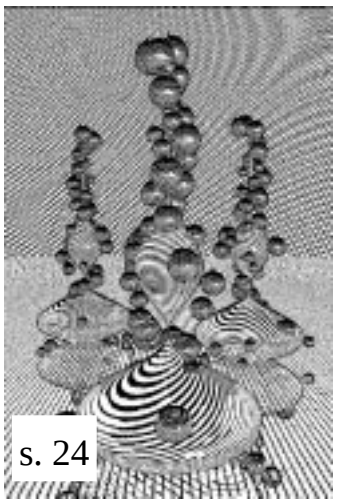


s. 5

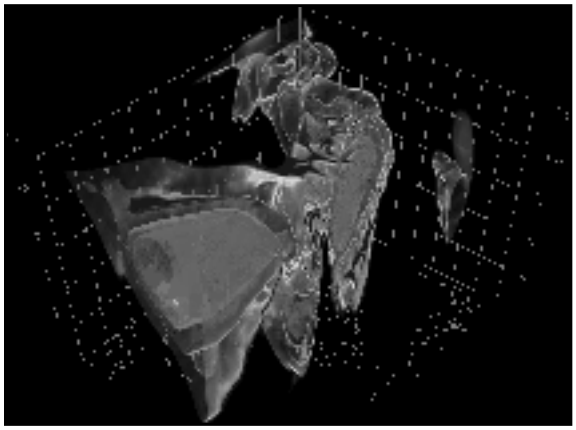
s. 14



s. 18



s. 24



Om forsiden

Bildet, laget av Jo Asplin, viser et atmosfærisk datasett som stammer fra en værprognose for Europa. Gitteret er på 121x97 punkter horisontalt og 18 punkter vertikalt. Toppen av gitteret ligger på 15 km over havet, og gitteret er derfor strukket betydelig i høyden. Figuren viser horisontal vindhastighet v.h.a. fire halvgjennomsiktige ISO-flater (ved 30, 40, 50 og 60 m/s).

NORSIGD Info

- medlemsblad for NORSIGD

Utgitt av:	NORSIGD	
Ansvarlig:	Nils Thune Metronor AS Boks 238 1360 Nesbru	
Utgivelser	1994: 1/9, 20/10, 15/12	
Annonsepriser:	Helside Halvside	kr. 5.000 kr. 2.500
Layout:	Nils Thune FrameMaker 4.0.3	

Hilsen fra styret

Kjære medlemmer,

Vi nærmer oss jul med *storm*skritt og hva passer vel ikke bedre da enn å ha med en artikkel fra Jo Asplin som omhandler blant annet volum visualisering av *vær*prognoser. Jo Asplin holder til daglig til ved Universitetet i Tromsø.

Som nok mange av våre lesere er kjent med så ble GPGS-F introdusert som en grafisk standard allerede tidlig på syttitallet. Nå har det skjedd mye siden den gang, og OpenGL er en av nykommerene på grafikk-bibliotek siden. Vi har vært så heldige å få Mason Woo til å introdusere oss til denne industristandarden. Mason Woo er ansatt ved Silicon Graphics, Mountain View, California.

MIPS, FLOPS, Specs, etc. er kjente begrep når ytelse på en CPU skal måles. Finnes det noe tilsvarende for måling av grafikk-ytelse? Jens Holwech, Digital, gir oss en grundig innføring i “benchmarking av 3D grafikk”.

Fremstilling, visualisering, osv. er ord som flommer ut av oss når vi tenker på data som skal presenteres på en eller annen måte. Wolfgang Leister gir oss noen tanker om det å *illustrere* ting.

Marianne Wallin er gir oss igjen informasjon om hva som skjer på GPGS-F fronten og i Grafikk hjørnet har vi sakset litt fra internettet med tanke på å gi enda litt mer bakgrunnsinformasjon om OpenGL.

Ellers har vi som vanlig en fast spalte med ting som skjer rundt omkring i verden relatert til grafisk databehandling.

En riktig God Jul og et Godt Nytt År!

Hilsen,

Nils Thune

50,— pr. stk.

NORSIGD har et lite restlager av EG '94 T-Shirts med EG '94 logo trykt på.

Benytt den unike sjansen til å få ditt eget eksemplar i dag.

Vi selger ut restlageret til en pris av 50,- pr. stk. (inkl. frakt innen Norge).

Kvaliteten er meget god: 100% bomull, laget og designet i Norge.

Aldri har en femtilapp hatt større verdi!

Jeg bestiller: ___ stk. (XL) ___ stk. (L)

Navn:.....

Firma:.....

Gateadresse: .....

Postadresse:.....

Postnummer/sted:.....

.....

Telefon.....

NORSIGD v/Marianne Wallin

ViaNova AS

Postboks 53

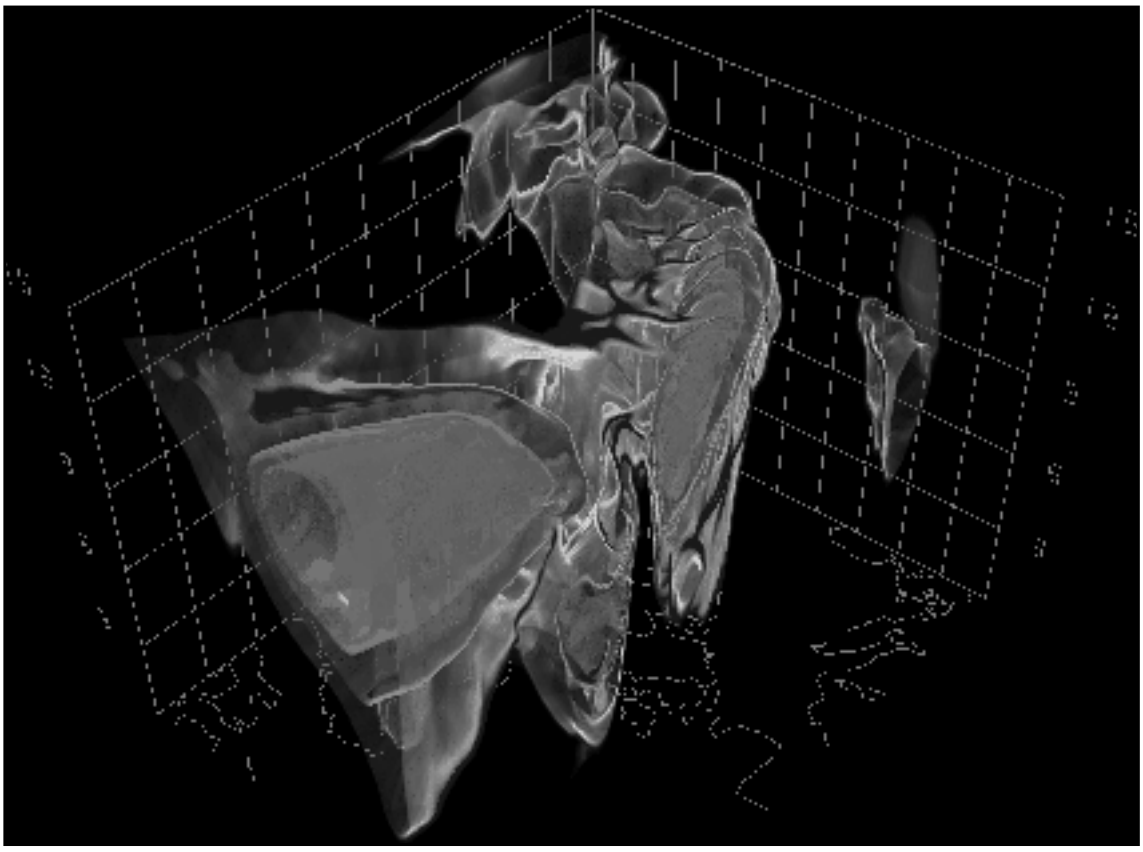
1312 SLEPENDEN

Parallell Volumvisualisering av Atmosfæriske og Molekylære Datasett

Jo Asplin, Seksjon for Informatikk, Universitetet i Tromsø

Innen vitenskap er det ofte interessant å studere størrelser som er fordelt i et tre-dimensjonalt (3D) rom. Slike størrelser kan representeres i volumetriske datasett som ofte er så store og komplekse at 3D visualisering ved hjelp av datagrafikk er nødvendig for en effektiv analyse. Volumvisualisering er en teknikk for visualisering av den indre strukturen til et volumetrisk datasett. I det enkleste tilfellet representerer det volumetriske datasettet skalarverdier, og i denne artikkelen skal et volumetrisk datasett forstås som et slikt enkelt skalarfelt.

Meteorologi er et eksempel på en vitenskap der studiet dreier seg om størrelser i et 3D rom. Størrelsene det er snakk om er fysiske størrelser i atmosfæren. I meteorologi er det nødvendig å representere de atmosfæriske størrelsene i et numerisk gitter med et endelig antall punkter. Slike gitre danner utgangspunkt for simulering av atmosfærens tidsutvikling, men også for volumvisualisering av atmosfæren. Målet med en slik visualisering er i utgangspunktet å forstå hvordan verdien til en størrelse varierer i rommet på et gitt tidspunkt.



Figur 1. I en værprognose for Europa vises horisontal vindhastighet v.h.a. fire halvgjennomsiktige ISO-flater (ved 30, 40, 50 og 60 m/s).

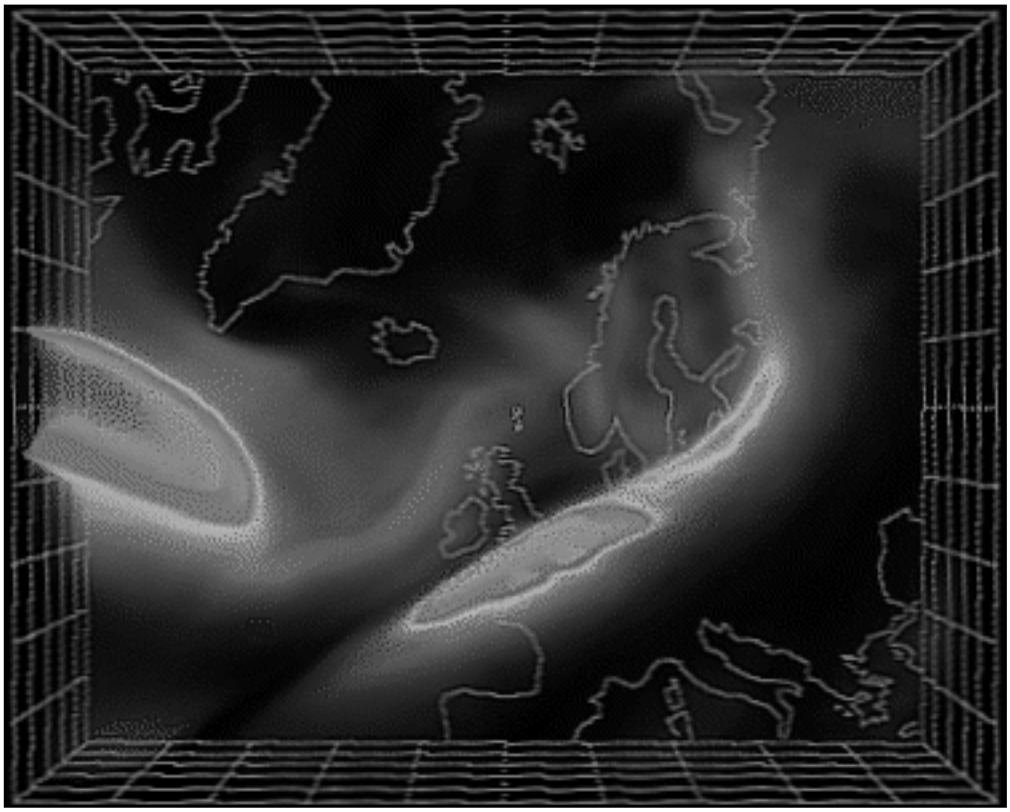
En annen vitenskap som studerer volumetriske størrelser er kjemi. I modellering av kjemiske fenomener inngår ofte bruk av elektronfunksjoner (også kalt orbitaler). En slik funksjon er definert for et molekyl og angir sannsynligheten til enhver tid for at gitte elektroner befinner seg på gitte punkter i rommet som omslutter molekylet. I likhet med atmosfæriske størrelser kan elektronfunksjoner representeres i et gitter med et endelig antall punkter. Hovedforskjellen mellom disse gitrene er selvsagt størrelsen. Et typisk atmosfærisk gitter har en utstrekning på 6050x4850x15 km^3 mens et molekylært gitter ofte ikke omfatter mer enn et par Ångstrøm (10^{-7} mm) i hver dimensjon.

Volumvisualisering er svært ressurskrevende både med hensyn til plass og tid. Mange volumvisualiserings-algoritmer har et stort potensial for parallell utførelse. Hvis dette potensialet utnyttes godt kan ressursbruken
reduseres betydelig.

Denne artikkelen baserer seg på et hovedfagsprosjekt utført ved Seksjon for Informatikk, Universitetet i Tromsø [1]. Artikkelen identifiserer først det grafiske potensialet til volumvisualisering. Deretter beskrives en konkret algoritme som løser disse problemene. Eksempler fra meteorologi og kjemi viser så visualiserings-potensialet til algoritmen. Et avsnitt om parallellitet demonstrerer hvordan tidsforbruket kan reduseres. Artikkelen avslutter med en kort oppsummering.

Volumvisualisering

Volumvisualisering lar oss studere den indre strukturen til et volumetrisk datasett [3]. Dette betyr at vi kan få et inntrykk av den kontinuerlige variasjonen av skalarverdien i rommet. Dette til forskjell fra flatevisualisering (konturering) som bare får frem diskrete ISO-flater med konstant skalarverdi. Disse to klassene av volumetrisk visualisering utfyller hverandre og ofte er det nyttig å anvende begge, enten hver for seg eller samtidig. Det



Figur 2. Viser samme datasett som i figur 1, men med en mer tåkeaktig effekt i områder med svakere vind.

er forøvrig verdt å merke seg at volumvisualisering på mange måter er en mer generell metode som i enkelte tilfeller også kan oppnå brukbare ISO-flate effekter.

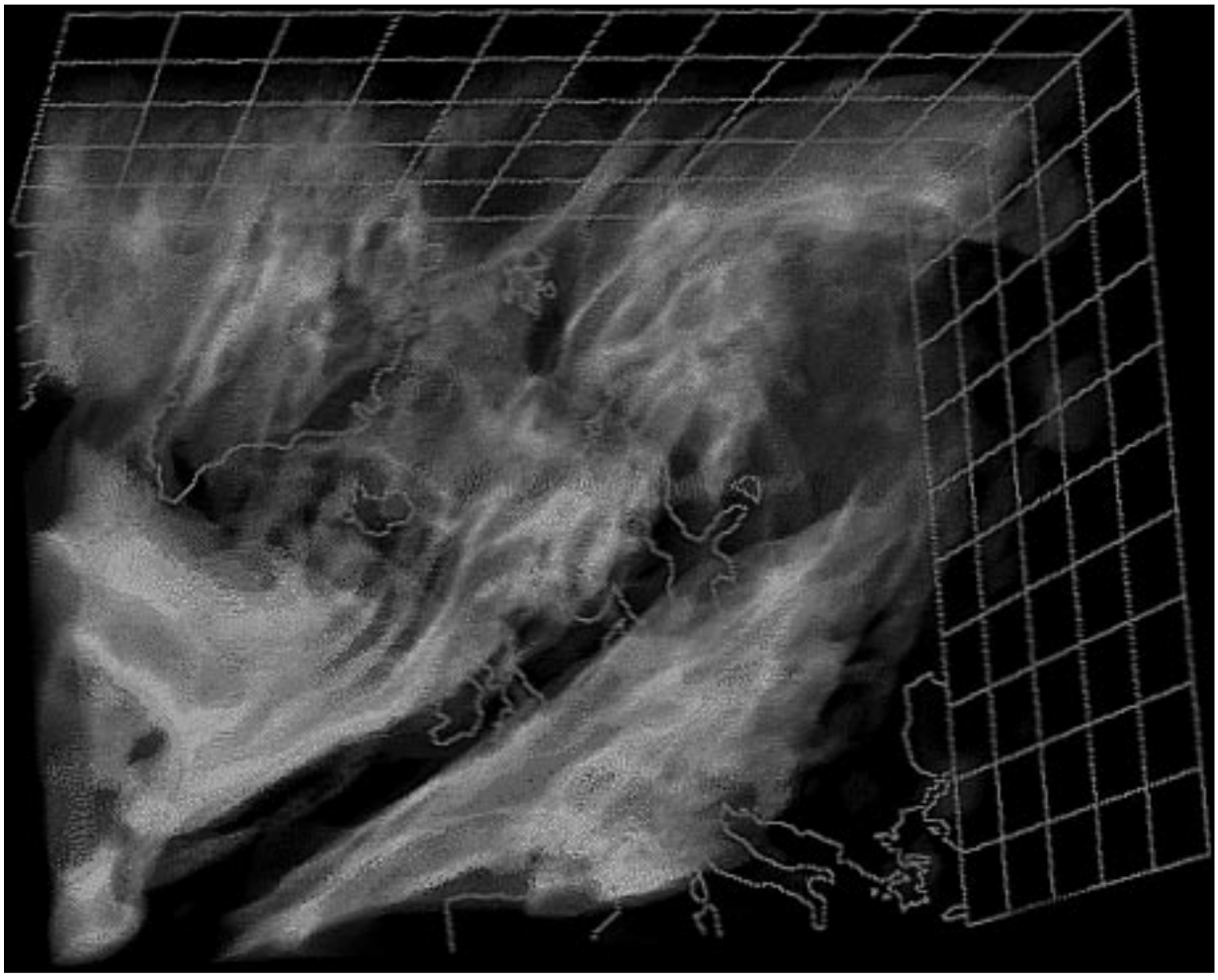
Dataklassifikasjon utgjør en essensiell del av input i volumvisualisering. Her angir brukeren hvordan skalarverdier skal gjenspeiles på farge og ugjennomsiktighet. På den måten kan hun kontrollere hvilke regioner av datasettet som skal representere synlig substans, hvor stor grad av gjennomsiktighet (transparens) de enkelte regionene skal ha, og hvordan den synlige substansen skal fargelegges.

I likhet med andre algoritmer for 3D-grafikk, kan algoritmer for avbildning av volumetriske datasett deles inn i to klasser: *bilderekkefølge* og *objektrekkefølge*. En bilderekkefølge-algoritme beregner den endelige fargen i ett bildeelement før det neste behandles [4]. En objektrekkefølge-algoritme traverserer gitteret region for region mens fargebidrag blir akkumulert i de projiserte bilde-elementene [5].

I dette arbeidet benyttes en bilderekkefølge-algoritme, kalt strålekasting. Denne ble valgt delvis fordi den gir et godt utgangspunkt for-parallellisering på den tilgjengelige system-plattformen (se senere avsnitt om dette).

Datastrukturer

Tre datastrukturer står sentralt i strålekastings-algoritmen. *Gitterobjektet* representerer et volumetrisk gitter med én enkelt skalarverdi i hvert punkt. Gitterpunktene er organisert parallelt i forhold til de tre aksene, men avstanden mellom nabopunkter kan variere fritt. *Avbildingsobjektet* inneholder



Figur 3. Relativ luftfuktighet vist ved hjelp av en kontinuerlig fargeovergang fra 80 til 100% fuktighet.

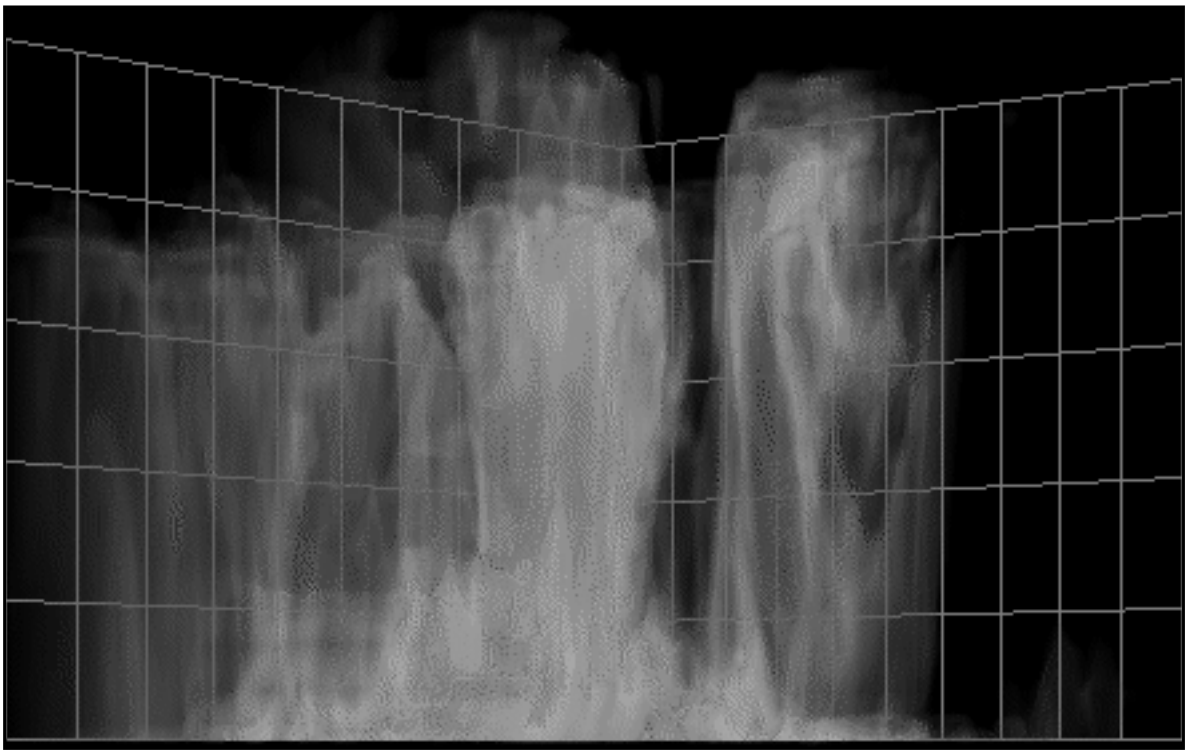
parametre for å kontrollere én enkelt avbildning (visualisering) av gitterobjektet. Den essensielle informasjonen består av 3D-2D transformasjon (dvs. synsvinkel) og dataklassifikasjon, dvs. funksjoner fra skalarverdi til ugjennomsiktighet og farge. Disse funksjonene trenger ikke å være kontinuerlige, og dermed kan vi også oppnå momentane overganger fra f.eks. gjennomsiktig til ugjennomsiktig materiale. *Bildeobjektet* representerer bildet som produseres av en gitt avbildning. Det er en matrise av RGBα-tupler som korresponderer direkte med bildeelementene. R, G og B er intensiteter for rød, grønn og blå fargekomponent, mens α er ugjennomsiktighet. Bildeobjektet inneholder således tilstrekkelig informasjon til å kunne blande inn et bakgrunnsbilde som f.eks. en gitter-referanse eller et landskap.

Algoritme

Totaleffekten av strålekastings-algoritmen er at et bildeobjekt blir generert på grunnlag av et gitterobjekt og et avbildingsobjekt. På øverste nivå traverserer algoritmen bildet og sender en stråle gjennom hvert bildeelement og inn i gitterrommet. RGB-intensiteter og ugjennomsiktighet integreres numerisk langs hver stråle der integranden beregnes på grunnlag av dataklassifikasjon og trilineær interpolasjon av skalarverdier. Integrasjonen fortsetter inntil enten en siktgrense nås eller strålen når baksiden av gitteret. På det tidspunktet kan så RGBα-tuppelet beregnes ut fra integralene.

Eksempler

Strålekastings-algoritmen har i dette arbeidet vært anvendt på datasett fra meteorologi og kjemi. De atmosfæriske datasettene stammer fra en værprognose for Europa. Gitteret er på 121x97 punkter horisontalt og 18 punkter vertikalt. Toppen av gitteret ligger på 15 km over havet, og gitteret er derfor strukket betydelig i høyden. Figur 1 viser horisontal vindhastighet v.h.a. fire halvgjennomsiktige ISO-flater (ved 30, 40, 50 og 60 m/s). Figur 2 viser samme datasett, men med en mer tåkeaktig effekt i områder med svakere vind. I figur 3 - 5 vises relativ luftfuktighet fra tre



Figur 4. Samme som figur 3 men sett fra en annen synsvinkel.

synsvinkler. Her skjer visualiseringen ved hjelp av en kontinuerlig fargeovergang fra 80 til 100% fuktighet.

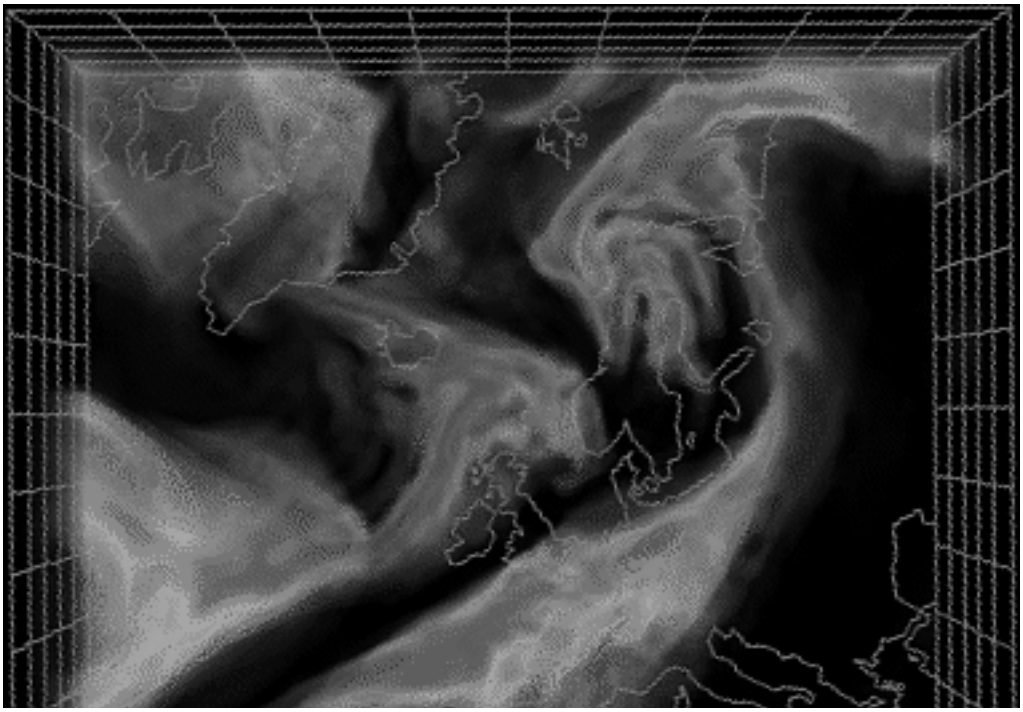
De molekylære datasettene representerer to ulike elektronfunksjoner for gullhydrid (AuH). Gitteret er her på 60x60x60 punkter med jevn avstand mellom nabopunkter. Figur 6 og 7 gir to visualiseringer av samme elektronfunksjon - en med iso-flater og en med kontinuerlig fargeovergang. De to siste figurene (8 og 9) viser en annen elektronfunksjon på samme måte, men bilde 9 viser også effekten av en svak belysning fra høyre.

Parallellitet

Målinger viser at for enkelte datasett og avbildnings-spesifikasjoner kan gjennomsnittlig tidsforbruk for en stråle være så høyt som 40 ms. I det tilfellet tar det altså 2.8 timer å beregne et bilde på 500x500 bildeelementer. For å bøte på denne situasjonen undersøkes det i dette arbeidet hvordan parallell prosessering kan anvendes. Utgangspunktet er at strålekastings-algoritmen har et høyt potensial for parallell utførelse, spesielt hvis samtlige prosessorer kan allokere hele datasettet i sine primærlagre. For å utnytte parallelliteten maksimalt er det imidlertid viktig å få til en god belastningsbalansering [2].

Systemplattform

Systemplattformen for den parallelle implementasjonen består av Hewlett Packard HP-720 arbeidsstasjoner med PA-RISC 7100 prosessor og 32 MB primærhukommelse. Hver arbeidsstasjon kjører operativsystemet HP-UX 9.0, og kommunikasjon skjer på et lokalt 10 Mbit/s Ethernet. Denne systemplattformen egner seg godt for parallelle algoritmer der prosessinteraksjon utgjør en liten andel av det totale tidsforbruket og der de enkelte prosessene har behov for å allokere store datastrukturer. Strålekasting oppfyller denne betingelsen siden de enkelte strålene kan beregnes uavhengig av hverandre så fremt hele gitteret er umiddelbart tilgjengelig.



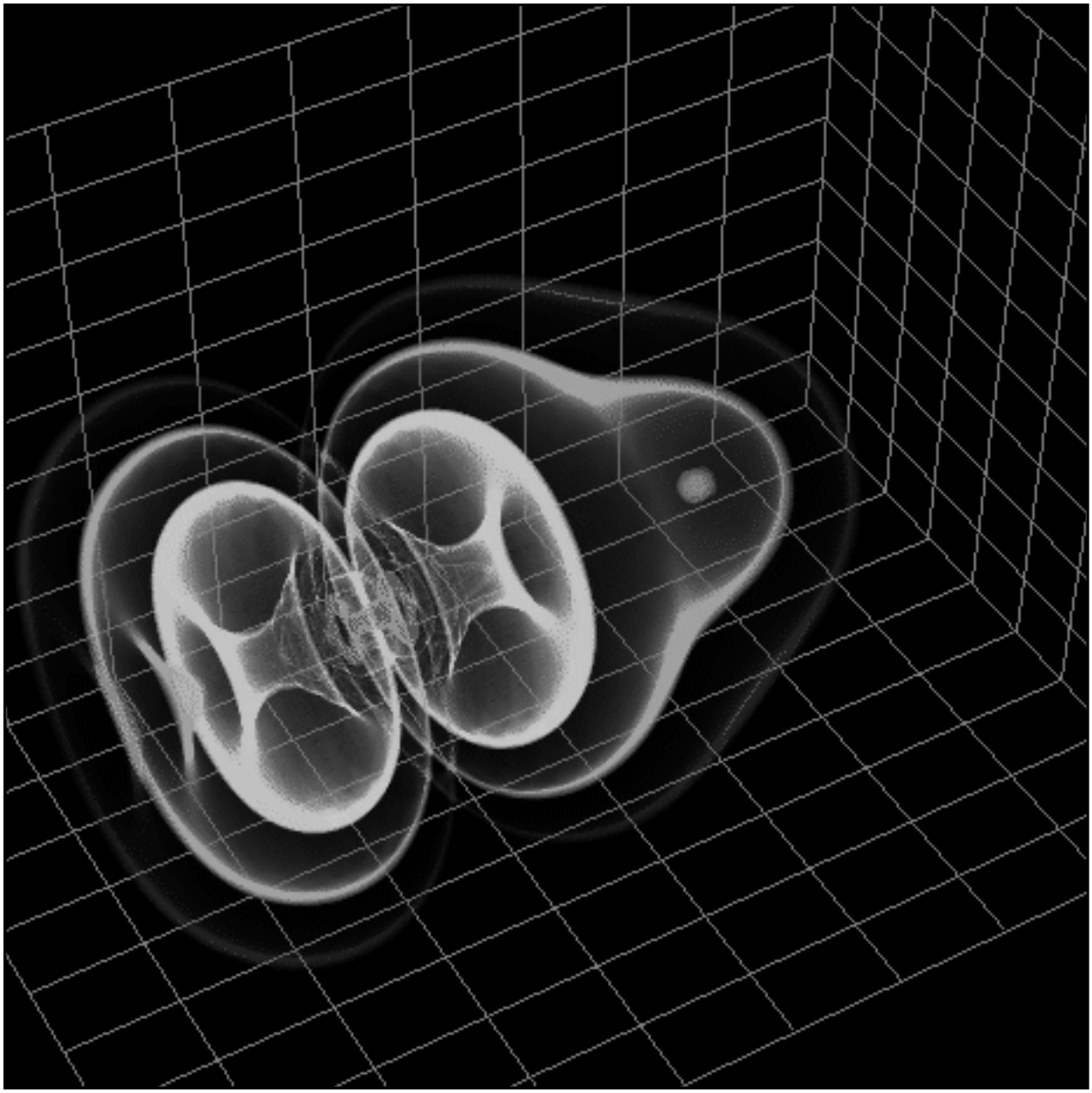
Figur 4. Samme som figur 3 men sett fra en annen synsvinkel.

Arkitektur

Arkitekturen tar utgangspunkt i prosessor-farm-modellen der en *kontroller* har ansvaret for at et sett med *arbeidere* utfører et sett med jobber i parallell. Kommunikasjon skjer kun mellom kontroller og arbeider, og arbeiderne interagerer ikke direkte med hverandre. Resultatet av delberegningene settes sammen til et totalresultat av kontrolleren. Også en tredje type prosess er involvert, nemlig en som implementerer et grafisk brukergrensesnitt og som derfor kommuniserer med kontrolleren.

Algoritme

Parallellisering skjer på bildenivået, dvs. at bildet deles opp i et antall delbilder som svarer til jobber. Typisk er antallet jobber mye større enn antallet arbeidere, så hver arbeider må i utgangspunktet belage seg på å utføre mer enn én jobb. I det følgende antar vi at gitteret på forhånd er replikert hos arbeiderne (dvs. allokert i sin helhet hos hver av dem). Når brukeren har spesifisert et nytt avbildingsobjekt, sendes dette via kontrolleren og ut til arbeiderne. Den parallelle algo-

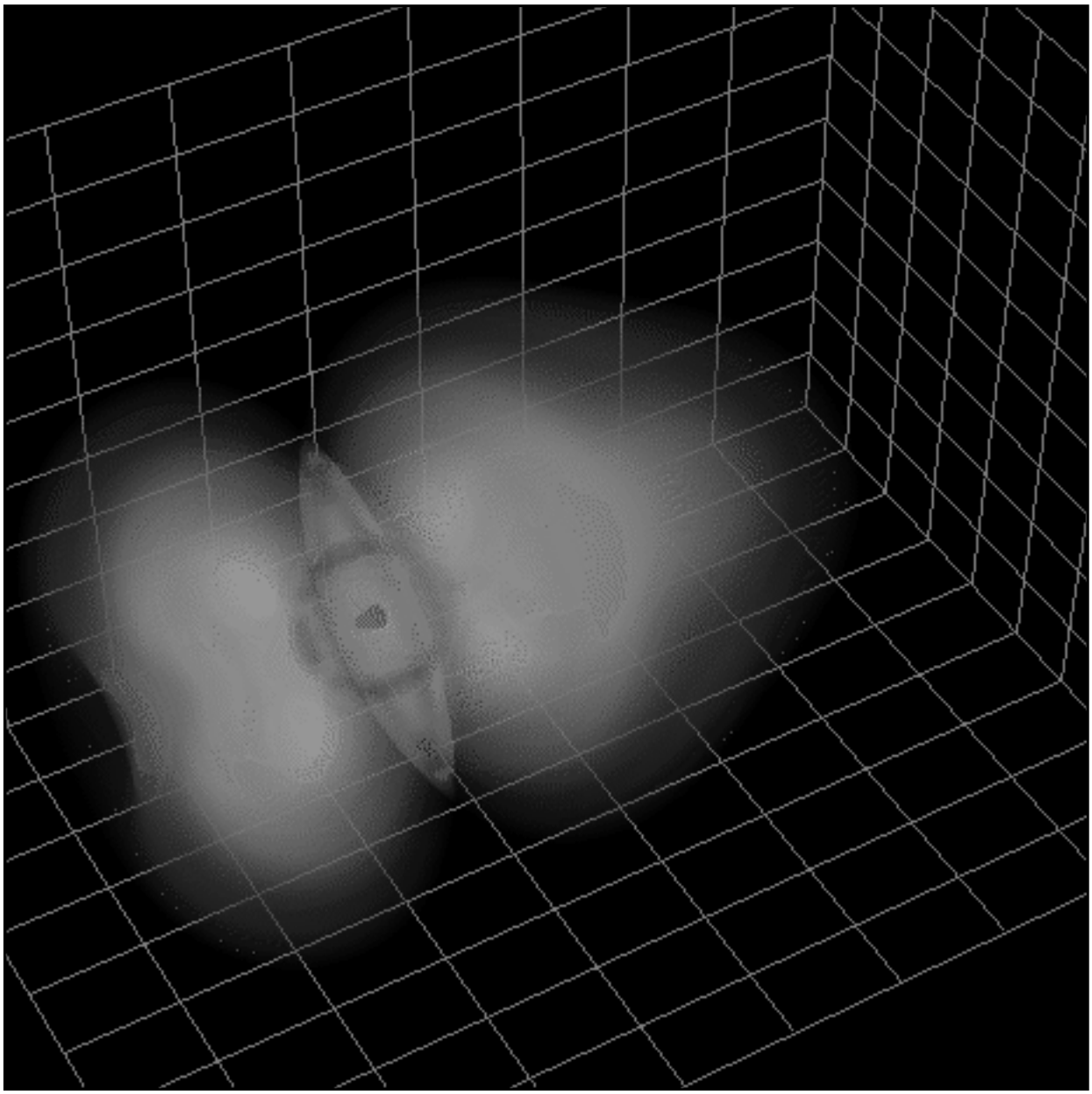


Figur 6. Visualiseringer av en elektronfunksjon for gullhydrid (AuH) - ved hjelp av iso-flater.

ritmen går nå inn i en fase hvor arbeiderne selv må ta initiativ til å be kontrolleren om jobber. En slik jobbforespørsel vil også inneholde resultatet fra forrige jobb arbeideren utførte slik at kontrolleren kan registrere dette bidraget før en ny jobb deles ut til arbeideren. Når samtlige bidrag er samlet inn, gir kontrolleren arbeiderne beskjed om å vente på neste avbildingsobjekt før den leverer det ferdige bildet til brukergrensesnittet.

Dersom jobbstørrelsen velges med omhu, vil denne algoritmen automatisk oppnå balansert utførelse i den forstand at samtlige arbeiderne holdes i arbeid hele tiden. Arbeidere som enten utføres på hardt belastede prosessorer eller får tildelt jobber som krever lang beregningstid, vil sende færre forespørsler og dermed bli tildelt færre jobber. Intuisjonen for at jobbstørrelsen er kritisk er at små jobber fører til mer kommunikasjon mens store jobber fører til et dårligere balanseringspotensiale.

Resultater



Figur 7. Visualiseringer av en elektronfunksjon for gullhydrid (AuH) - ved hjelp av kontinuerlig fargeovergang.

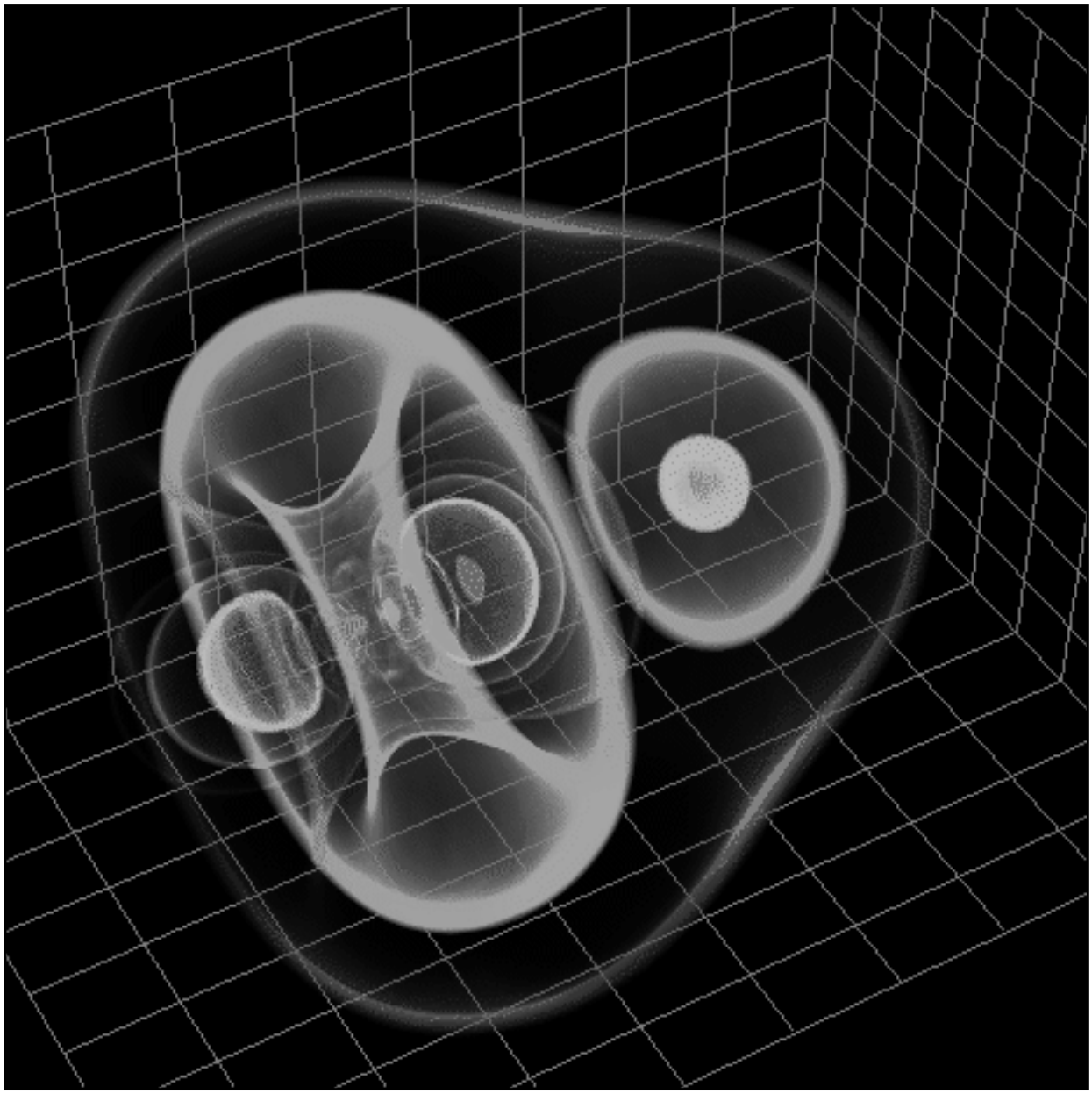
Eksperimenter med den parallelle algoritmen på 35 arbeidsstasjoner viser at den oppnår tilnærmet lineær speedup opp til dette antallet. Dette resultatet var avhengig av en jobbstørrelse på rundt 250 bildeelementer.

Oppsummering

Erfaring med programsystemet som er utviklet i dette arbeidet viser at volumvisualisering kan være et nyttig analyseverktøy for både meteorologer og kjemikere. Det er dessuten mulig å implementere en effektiv parallell algoritme på et regulært nett av arbeidsstasjoner. Et slikt system kan derfor være et alternativ til mer spesialiserte og kostbare plattformer for volumvisualisering.

Kreditering

Forfatteren ønsker å rette en takk til følgende personer som på ulike måter har bidratt til arbeidet: Bård Fjukstad, Anstein Foss, Trond Saue, Dag Johansen, Karl Henrik Eggestad



Figur 8. Visualiseringer av en elektronfunksjon for gullhydrid (AuH) - ved hjelp av iso-flater.

og William Hibbard.

Litteratur

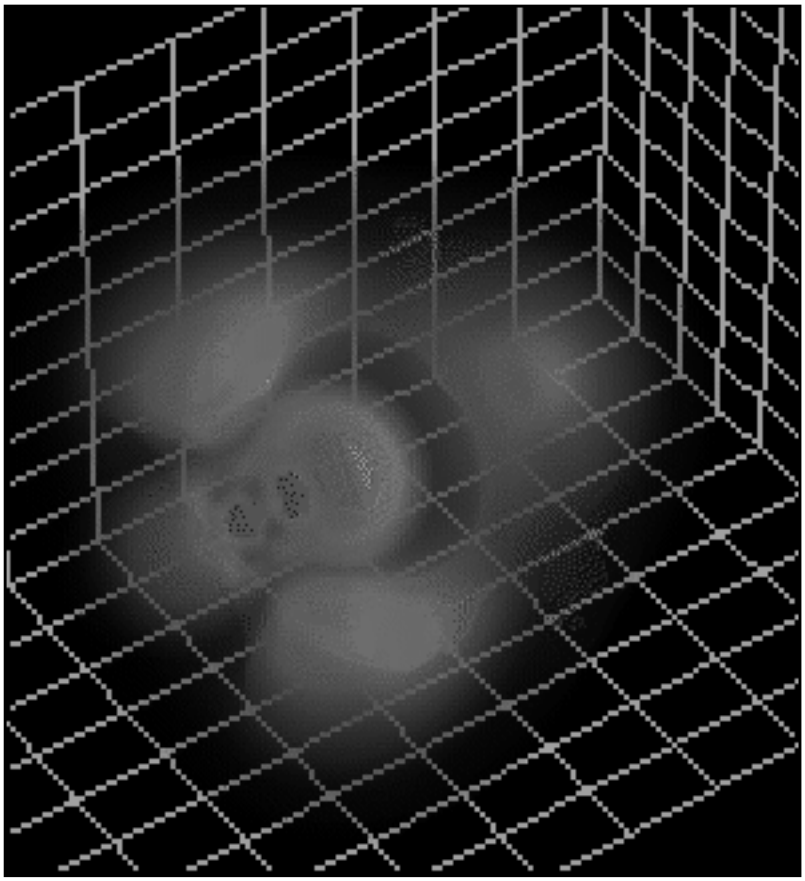
[1] Jo Asplin. *Distribuert Parallell Volumvisualisering av Skalarfelt fra en Atmosfæremodell*. Hovedoppgave i Informatikk, Universitetet i Tromsø, Mars 1994.

[2] Stuart Green. *Parallel Processing for Computer Graphics*. Pitman, 1991.

[3] Arie Kaufman. Introduction to Volume Visualization. In Arie Kaufman, editor, *Volume Visualization*, pages 1 - 18. IEEE Computer Society Press, 1991.

[4] Craig Upson and Michael Keeler. V-BUFFER: Visible Volume Rendering. *Computer Graphics*, 22(4):59--64, August 1988.

[5] Lee Westover. Footprint Evaluation for Volume Rendering. *Computer Graphics*, 24(4):367--376, August 1990.



Figur 9. Visualiseringer av en elektronfunksjon for gullhydrid (AuH) - ved hjelp av kontinuerlig fargeovergang.

The Design Philosophy of the OpenGL Graphics API

Mason Woo, Silicon Graphics

The world is not flat, but computers have tended to visualize the world as a flat, two-dimensional computer screen. This visual model has worked for desktop publishing, but as low-cost technology makes high performance computing more accessible, users will want to use a more realistic model of the world. Doctors who first used X-rays, then used CAT scans, will now see a solid model of their patient, before surgery. An automobile designer will see if changes to a "virtual" design make a car more safe, without building and destroying dozens of real and costly car bodies.

The real world is a 3D world, and the desktop computer will become more able of operating with 3D data. The power and price of the 3D hardware will vary for different manufacturers and platforms. But the combined efforts of many companies has created a standard software interface for 3D graphics: the OpenGL(TM) graphics library. With OpenGL, advanced software, such as virtual reality, scientific visualization, and flight simulation will be portable between different companies, from PC to workstation. The end-user will win, as the end-user will be able to choose the correct hardware at the affordable price and be certain that their software application will run on it.

A simple description of the OpenGL™ graphics API (Application Programming Interface) must include these characteristics:

- The operation of OpenGL is only for rendering graphics. OpenGL is not for performing window management or handling events or input devices, which already have a variety of well-defined programming interfaces.
- OpenGL has a large set of graphics functions, which can be rendering interactively with state-of-the-art graphics hardware. On less powerful or frame-buffer-only graphics hardware, some OpenGL features will render slowly.

Those two restrictions would define almost any graphics library, but the philosophy behind the design of OpenGL is also greatly influenced by:

- the five graphics primitives which can be rendered, and
- the state machine (or modes), which specify which operations will take place to the primitives.

It is the elegance of the OpenGL state machine design which enables the variety of graphics images which can be rendered with this API.

The Primitives

Computer graphics can be used to create images of automobiles, airplanes, houses, human bodies, entire industrial plants, or even prehistoric dinosaurs. At the simplest level, a graphics library must allow the rendering of any of these objects. With an emphasis on simplicity, the design for the OpenGL API provides for only five drawing primitives. *Points, line segments, filled polygons* are the only geometric primitives. These vertex-based primitives are the basic elements of all 3D objects. There are two additional raster primitives: *the bitmap* and

the image rectangle (also known as a pixmap, or simply as an image). The bitmap is a single color rectangle of pixels, which is used for fonts and polygon stipples. The image is a set of pixels of many colors. Typically, photographs, textures, or the contents of the frame buffer are stored in an image rectangle.

OpenGL supports some higher-level primitives indirectly. There are OpenGL commands for *quadrics* (e.g., spheres, cylinders, conics) and *NURBS* (non-uniform rational B-splines for curves and surfaces), but those objects are ultimately reduced to the aforementioned points, lines, and polygons. A 3D OpenGL object is not *solid*. Only the surfaces are modeled, and if the interior is examined, there is *nothing* inside. OpenGL currently does not have any *voxel* primitives, which reflects the current technology of graphics hardware architecture, which is primarily vertex based.

For geometric drawing primitives, a programmer specifies an ordered set of vertexes. Each vertex command specifies its coordinate position in a 3D world. This list of vertexes must be surrounded by calls which specify which geometric primitive is constructed. For example, the following code would specify a three-sided filled polygon to be drawn.

```
glBegin (GL_POLYGON);
glVertex3f (x1, y1, z1);
glVertex3f (x2, y2, z2);
glVertex3f (x3, y3, z3);
glEnd ();
```

However, if `GL_LINE_LOOP` replaced `GL_POLYGON` in the previous code, then the code specifies drawing three lines, connected into a closed polygon. If it said `GL_POINT`, then only three, unconnected vertexes would be specified.

For raster drawing primitives, a position where the next bitmap or image will be drawn must be specified. We can expect to see code which looks like:

```
glRasterPos2i (x1, y1);
glBitmap (w, h, xorigin, yorigin,
xinc, yinc, bitmap);
or
glRasterPos2i (x1, y1);
glDrawPixels (w, h, format, type,
pixels);
```

The OpenGL State Machine

The other major design principle of OpenGL is that the graphics operations are affected by the current states (or modes) that remain in effect, until they are changed. The OpenGL *state machine* can be thought of as a large array of state values. Some of the values are Boolean, either ON or OFF. (Do I turn on lighting? Texturing? Hidden Surface Removal? Alpha Blending? Culling?) In most cases, enabling a state (i.e., turning it on) increases the amount of work to be done. Some of the state machine require more values. For example, if fog is enabled, the fog equation, density, and color may also need to be provided.

The programmer must prepare the values in the state machine, such as the camera position and lighting, **before** the primitives are rendered. The state may be changed between rendered objects. For example, if you want to draw one red object and one green object, you might change the value of the color state variable between objects:

```
//Parameters are (red, green, blue).
glColor3f (1.0, 0.0, 0.0);

//Draw object, current color is red
...
//Change current color to green
glColor3f (0.0, 1.0, 0.0);
//Draw green object
...
```

In the previous section, the `glBegin (GL_POLYGON)`, three vertex commands, and `glEnd()` were discussed, as describing a three-sided filled polygon. However, different states can drastically alter the appearance of the final drawing. If these two commands

precede the polygon, then the polygon will not be solidly, but be filled by the repeating pattern, specified by the array `bitmask`:

```
glEnable (GL_POLYGON_STIPPLE);
glPolygonStipple (bitmask);
```

If the bitmask is a fine pattern, the polygon may look like a window screen.

Another example of defining a state is polygon culling, which is a method which can reduce the number of polygons which are drawn. For instance, when a sphere is drawn, you may wish to draw the polygons which are facing the viewer, but not to draw the polygons which are *facing away* (on the back side).

`glEnable(GL_CULL_FACE)` is enough to eliminate all *back-facing* polygons, with **front**-facing defined as a polygon rendered with its vertexes in counter-clockwise order. Suppose your data for your polygons had all vertexes for front-facing polygons in *clockwise* order. There is no need to try to reorder your data. Just use the OpenGL routine `glFrontFace(GL_CW)` to redefine what front-facing is!

Continuing with a similar example, the `glPolygonMode()` routine alters the state machine values for how polygons are rasterized (filled). The default is `glPolygonMode(GL_FRONT,GL_FILL)`, which says that front-facing polygons should be filled. By calling `glPolygonMode(GL_FRONT,GL_LINE)`, a front-facing polygon will now be drawn as connected line segments and appear hollow. `glPolygonMode(GL_FRONT,GL_POINT)` would only reduce a polygon to only three vertexes.

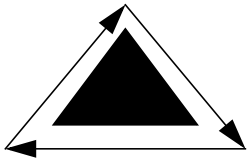
These effect of these states are designed to be **orthogonal**, that is, the effect of turning on several states is to be cumulative. In the previous example, we discussed, polygon stippling, culling, and rasterization. Each state is independent, but the cumulative effect should result, wherever possible, in predictable and useful graphics. If our polygon was a three-sided, clockwise ordered polygon, the different states would look as

NORSIGD Email liste!

Dette er en liste over personer/organisasjoner/firmaer/bedrifter som ønsker å få elektronisk informasjon om kommende seminarer arrangert av NORSIGD, status på GPGS-F, og andre ting som har med NORSIGD å gjøre.

Så langt det er mulig prøver vi også å bidra med informasjon om grafikk-relaterte spørsmål via denne listen.

Kunne du tenke deg å komme med på NORSIGDs Email liste? Send i så fall en email til *thune@oslonett.no.*



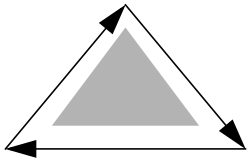
default

```
glBegin (GL_POLYGON);
glVertex3f (x1, y1, z1);
glVertex3f (x2, y2, z2);
glVertex3f (x3, y3, z3);
glEnd ();
```

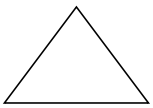
```
glEnable(GL_POLYGON_STIPPLE);
glPolygonStipple(bitmask);
```

'disappears'
polygon culled

```
glEnable(GL_CULL_FACE)
```



```
glEnable(GL_CULL_FACE)
glFrontFace(GL_CW)
glEnable(GL_POLYGON_STIPPLE);
glPolygonStipple(bitmask);
(front-facing is redefined)
```



```
glPolygonMode(GL_FRONT,GL_LINE)
culling & line stipple affect this
    polygon stipple does not
```

shown above.

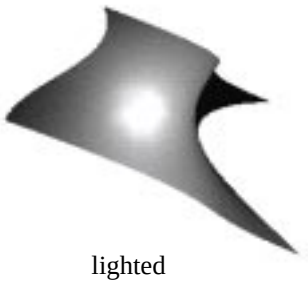
Orthogonality allows for the combination of different OpenGL functions. For example, lighting, texturing, and hidden surface removal can be combined to create a very pleasing image. In the next illustration, there is a NURBS surface, first rendered with only lighting, then with only texture mapping, and then with both lighting and texture states enabled.

Publications

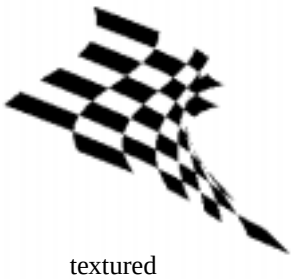
This article is intended to be a small introduction to OpenGL. For more technical information, the following manuals are available:

Neider, Jackie, Tom Davis, and Mason Woo, OpenGL Programming Guide: The Official Guide to Learning OpenGL, Addison-Wesley Publishers.

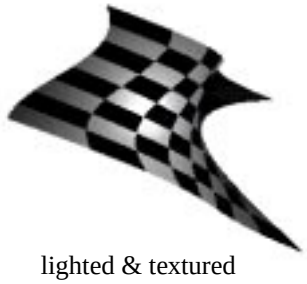
OpenGL Architecture Review Board, OpenGL Reference Manual: The Official Reference Document for OpenGL, Addison-Wesley Publishers.



lighted



textured



lighted & textured

Benchmarking av 3D-grafikk

Jens Holwech, Digital Equipment Corporation A/S

Bruken av arbeidsstasjoner med 3D-grafikk har økt kraftig i den senere tid. DAK er fortsatt det viktigste bruksområdet, men bruken av applikasjoner for annen data-visualisering øker kraftig. Det er i dag på ingen måte enkelt for en kjøper å orientere seg i jungelen av ytelses-tester som benyttes av de forskjellige leverandørene, både innenfor generell CPU-ytelse og spesielt innenfor grafikk-ytelse. Grafikk-området har vært spesielt vanskelig, idet det ikke har eksistert standard-tester på dette området. Forvirringen blir ikke mindre av at forskjellige leverandører oppgir tilsynelatende sammenlignbare måleresultater, men hvor det opereres med subtile variasjoner i detaljene.

Den viktigste leveregelen innenfor dette området bør være den samme som ved all annen benchmarking:

> **Test den aktuelle applikasjonen med dine egne data!**

Dersom dette ikke er praktisk mulig, så bør man undersøke hvilke standard-benchmarker som ligger nærmest opptil den aktuelle bruken. På dette området har det faktisk skjedd ganske store endringer i løpet av det siste året. Arbeidet som utføres av *Graphics Performance Characterization Committee (GPC)* har medført en markant dreining fra ensidig fokus på teoretisk maksimal hardware-ytelse og over til bruk av mer realistiske tester, utviklet for å simulere reell applikasjonsbruk. GPC er opprettet av *NCGA (National Computer Graphics Association)* i USA. Komiteen er åpen for alle interesserte, men initiativet ble tatt av de store maskin-leverandørene, og disse er fortsatt den viktigste drivkraften i arbeidet. De fleste aktørene i grafikk-bransjen har representanter i komiteen: Digital Equipment Corporation, Evans & Sutherland, Fujitsu, Hewlett-Packard, IBM, Intel, Kubota Graphics Corporation, Megatek, Silicon Graphics, Sun Microsystems, Tektronix. Komiteen oppfordrer selv til økt deltagelse av representanter fra andre fagområder, f.eks. program-utviklere og slutt-brukere.

PLB-tester

Hva har så GPC funnet på? De har utviklet en serie av tester som har fått betegnelsen *Picture Level Benchmarks (PLB)*. PLB-testene forsøker, som navnet indikerer, å måle hastigheten ved opptegning av komplette bilder, i motsetning til enkle primitiver. De forskjellige standard-bildene er laget med utgangspunkt i behovene fra forskjellige applikasjonsmiljøer. Det er laget tre nye, industri-standard benchmark-tester: *PLB2d*, *PLBwire* og *PLBsurf*. Ved måling av CPU-ytelse har bransjen etterhvert beveget seg bort fra MIPS-målinger og over til bruk av SPECint92/SPECfp92. GPCs intensjon er at PLB-testene skal bidra til tilsvarende utvikling innenfor grafikk-området.

Primitiv-tester

Tidligere har brukerne oftest vært nødt til å utføre sin vurdering med utgangspunkt i leverandørenes oppgaver over opptegningshastighet for forskjellige primitiver (vektorer, trekanter, polygoner). Fortsatt vil nok primitiv-tester måtte brukes i en del sammenhenger, og det er derfor nyttig å vite litt om hva som ligger bak disse tallene. Det er ellers fort gjort å ende opp med en sammenligning av “epler og bananer”.

Vendor	*Size (no of pixels)*	*Shading*	*Lighting*	*Comments*
Digital	50	Gouraud	1 ambient 1 directional	orthographic projection, clip checked (none clipped), back-face cull checked (none culled)
Hewlett-Packard	50	flat	none	clip checked and trivially accepted on all six clip planes; solid interior style; no culling, edging or additional data per vertex
Silicon Graphics	50	flat	unlit	
Sun	25/50	Gouraud/-flat	unlit and 1 ambient 1 directional	clip checked and trivially accepted on all six clip planes; solid interior style; backface cull checked (none culled); no edging; no additional data per vertex

Tabellen viser eksempler på leverandørenes forskjellige målemetoder for hastighet ved opptegning av 3D trekanter. I tillegg til disse forskjellige varierende attributtene ser alle leverandørene ut til å operere med en del felles attributter i målingene: "random orientation, connected triangles, constant color, z-buffering". Kilde: The GPC Quarterly, Vol. 4, No 1, unntatt for SGI som er hentet fra SGIs egen Workstation/Client Periodic Table, Aug. 1994.

Ved måling av antall vektorer/sekund er man ganske trygg. Det er ikke mange variasjoner mellom leverandørene her, men det er dessverre heller ikke så mange applikasjoner hvor dette forteller "alt" om ytelsen... Straks man begynner å se på trekanter/sekund begynner variasjonene å bli større. Aktuelle spørsmål man bør stille er f. eks.: Hvor store er trekantene (noen leverandører benytter 50 pixler, andre 25)? Hvordan er de orientert i rommet? Er de sammenhengende eller frittstående? Har de en eller flere farver? Benyttes z-bufring ved opptegningen? Størst variasjon ser det ut til å være i bruken av skyggelegging. Noen leverandører oppgir data ved bruk av enkel (flat) skyggelegging, mens andre benytter Gouraud-skyggelegging. Gouraud-skyggelegging gir mer realistiske bilder, men enkel skyggelegging kan ofte være akseptabelt mens man arbeider med utviklingen av en modell.

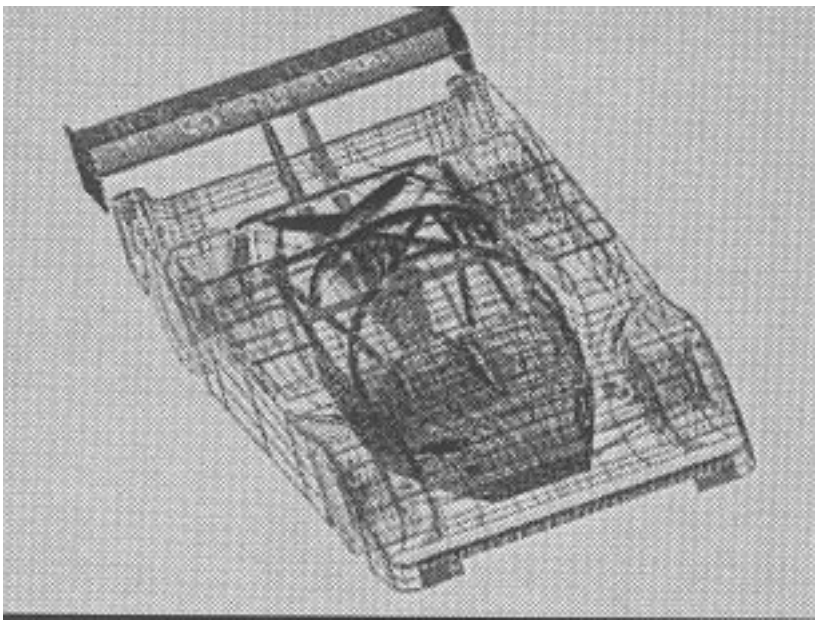
Tilsvarende problemstillinger gjelder også for antall polygoner/sekund. Her er det lyssettingen som ofte varierer. Det er dessverre lite sannsynlig at alle de aktuelle leverandørene har brukt samme sammensetning av bakgrunnslys, spot-belysning, og refleksjon. Moralen er: sjekk hvordan målingen er utført før man sammenligner "rått"!

Reelle bilder

Hva er det så som skiller PLB-testene fra primitiv-målinger? De tar som nevnt utgangspunkt i opptegning av en rekke bilder som er relevante innenfor hvert sitt fagfelt: PLB2d tar utgangspunkt i opptegning av en rekke 2-dimensjonale DAK/GIS-bilder. PLB2d er minst brukt av de tre standard-testene. For vurdering av generell 2D-ytelse gir antagelig den nye testen, Xmark93, et mer komplett bilde enn PLB2d. Vi kommer tilbake med nærmere omtale av Xmark93 i et senere nummer av NORSIGD Info. PLBwire og PLBsurf benytter begge 3-dimensjonale data, og er de mest brukte testene.

PLBwire

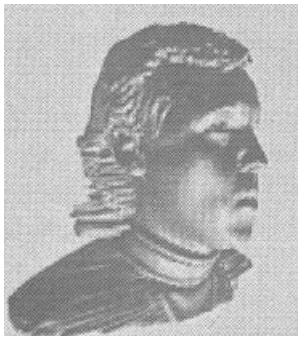
PLBwire måler ytelsen ved opptegning av 3D-vektorer. Grunnlaget stammer også her fra DAK- og GIS-modeller. Testen benytter tre bilder: sys_chassis, race_car og seafloor. Sys_chassis er en modell av chassiset til en datamaskin, og består av 6.107 heltrukne og 158 stiplede 3D polylinjer. Totalt er det 19.064 vektorer i modellen. Under testen blir modellen rotert, translert, skalert (zoomet) og vist i forskjellig perspektiv. Race_car er en 3D-trådmodell av en racerbil, opprinnelig modellert med Unigraphics. Under testen vises animasjon av modellen i fire samtidige vinduer på skjermen. I gjennomsnitt vises 17.917 polylinjer (141.934 vektorer) i hvert bilde. Animasjonen består av 600 bilder. Seafloor representerer en modell av havbunnen med konturlinjer. Modellen roteres 360 grader under testen, og dette gir 229.682 vektorer i gjennomsnitt for hvert av de 540 bildene.



Et bilde fra race_car-testen som inngår i PLBwire.

PLBsurf

I PLBsurf benyttes fire modeller: cyl_head, head, shuttle og studio. Cyl_head er en modell av en motorblokk. Under testen vises den fra forskjellige retninger, på tilsvarende måte som sys_chassis for PLBwire. I motsetning til sys_chassis er cyl_head en volummodell. Under testen vises den med Gouraud-skyggelegging, påvirket av tre forskjellige lyskilder med forskjellige type lys. Head er en modell av et menneske-hode, generert av en laser-scanner. Det er over 60.000 trekanter i denne modellen. Under testen rote-



Slik ser hodet som benyttes i head-testen ut. Head er en del av PLBsurf.

rer hodet tre ganger, og vises under påvirkning fra fire lyskilder. Shuttle er en "enkel" simuleringsmodell av en romferge som møter en sattellitt og beveger robotarmen under dette møtet. Studio er en virtuell vandring gjennom en arkitekt-modell av et studio. Lyset er forhåndsberegnet v.h.a. radiositet.

Generelt rammeverk

Alle testene benytter det samme rammeverket, som består av:

- et filformat for lagring av geometri- og bevegelsesdata (BIF - Benchmark Interface Format)
- rutiner for tidtagning (BTM - Benchmark Timing Methodology)
- et rapporteringsformat for resultatene (BRF - Benchmark Reporting Format)
- selve benchmark-programmet for å lese BIF-formatet, og gjennomføre testene
- test-programmer for å verifisere at implementasjonen er i henhold til spesifikasjonene
- standard BIF-filer for de forskjellige testene

Det er ingenting i veien for å lage sine egne BIF-filer, og gjennomføre tester med disse. Slik bruk oppmuntres av GPC, men er nok fortsatt ganske sjelden. De fleste kommer langt ved å sammenligne resultatene fra de mest relevante standard-testene. Resultatene for hver enkelt test inngår i BRF, og denne informasjonen er tilgjengelig fra leverandørene, eller direkte fra GPC. Det er altså fullt mulig å trekke ut de mest aktuelle del-testene når man gjennomfører sin vurdering. Alle test-resultater blir offentliggjort gjennom The *GPC Quarterly*, som er GPCs nyhetsbulletin. Siste utgave ble publisert i sommer, og neste utgave er ikke langt unna. Komplett oversikt over alle de siste PLB- og Xmark93-resultater vil bli offentliggjort i neste nummer av NORSIGD Info.

Leverandør-støtte

De fleste leverandører oppgir nå resultater fra PLB-testene for sine forskjellige maskiner og grafikk-kontrollere, med ett viktig unntak: Silicon Graphics nekter å oppgi resultatene for sine maskiner. Ettersom SGI er markedslederen innenfor dette området, så er dette et alvorlig tilbakeslag for utbredelsen av disse testene. SGI støtter initiativet for å utvikle bedre benchmark-tester, og deltar som nevnt i GPC-komiteen. Men de mener at PLB-testene gir et fortrinn til leverandører som benytter PHIGS eller PEXlib som programmeringsgrensesnitt (API). SGI har som kjent basert sin utvikling på bruk av Iris GL og baserer seg fremover på bruk av OpenGL. Det er riktig at PHIGS og PEXlib er mest brukt i disse testene, men de er ikke enerådende. IBM oppgir i dag resultater oppnådd både gjennom PEXlib og graPHIGS. HP gjennomfører testene både med sin egen pakke, Starbase, og HP PEX. Sun benytter sitt eget grensesnitt, XGL i sine tester. GPC innser at manglende støtte fra SGI er en alvorlig hindring for utbredelsen av PLB-testene. Det ble derfor allerede i 1993 gjennomført en studie for å undersøke om testene favoriserte bestemte APIer. Fil-formatet som benyttes i testene har sterke likhetstrekk med Clear Text Archive-formatet som er definert i PHIGS. Likevel var konklusjonen fra denne undersøkelsen faktisk at man ikke kunne se at PHIGS var spesielt favorisert. Det ble også satt i gang et arbeide for å utvikle programmer som kan gjennomføre PLB-testene med bruk av OpenGL. Det er nå uklart om dette arbeidet vil bli sluttført, idet det arbeides aktivt for å utvikle alternative tester for OpenGL.

OpenGL

Det er nylig opprettet en egen gruppe innenfor GPC, OpenGL Performance Characterization (OPC), for å utvikle tester basert på OpenGL. Man arbeider med to klasser av tester, først en Picture Level-test (viewperf), og deretter en primitiv-test (GLperf). Dette arbeidet er ikke sluttført, men det foreligger versjoner for utprøving av viewperf-konsep-

tet. IBM oppgir f. eks. resultater av sine interne OpenGL-tester i sitt brosjyremateriell. En vesentlig del av viewperf er sette sammen representative tester for forskjellige applikasjonsområder. På dette området samarbeider man med utviklere og sluttbrukere. Man håper å kunne offentliggjøre resultater tidlig i neste år.CPU-bruk

CPU-bruk

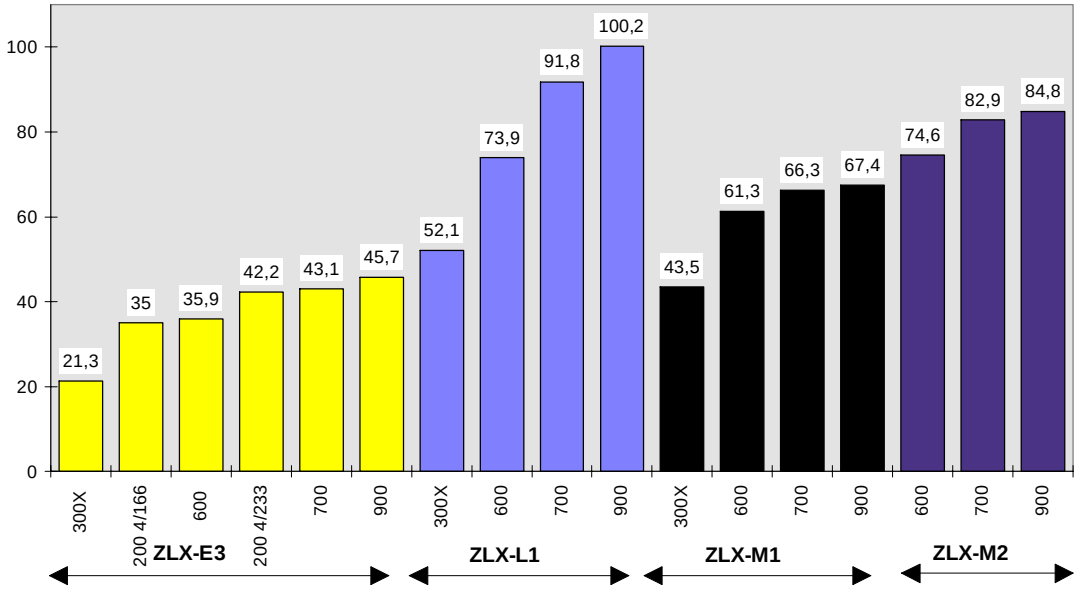
Selv om PLB-testene og andre høynivå-tester gir et bedre vurderingsgrunnlag enn tidligere, så er det fortsatt viktig å inkludere andre momenter i vurderingen. Jeg skal ikke her gå nærmere inn på områder som pris, garanti- og service-betingelser, etc., men nøye meg med å nevne andre forhold som berører vurdering av utstyrets total-ytelse. PLB-testene viser ytelsen av grafikk-systemet, og primært da grafikk-kontrollerne. I reell applikasjonsbruk vil grafikk-ytelsen ha varierende betydning. Selv i grafikk-intensive applikasjoner, som DAK-, GIS- eller visualiserings-systemer, vil det være viktig å vurdere ytelses-behovet også for andre deler av systemet. Det er f.eks. viktig å vurdere CPU-ytelse når man skal danne seg et totalbilde. Ved utvikling av grafikk-systemer kan man velge forskjellige løsninger for kontroller-arkitekturen: Kontrollerne kan f. eks. designes for å utnytte CPU-kraft, eller for å avlaste CPUen. Dette vil i ytterste konsekvens kunne bety at systemer med sammenlignbare PLB-resultater, vil kunne gi svært forskjellig applikasjons-ytelse når systemet skal utføre modelleringsoppgaver etc., i tillegg til å vise frem resultatene grafisk. Slike forhold fremgår ikke av grunnlaget for PLB-testene. Som et eksempel kan nevnes at ved test av Digitals arbeidsstasjoner med grafikk-kontroller av typen PXG+ gikk CPUen på tomgang kun i 5% av tiden, mens resultatene for ZLX-M1 ble oppnådd med 60% ledig-tid, og 70% med ZLX-M2. Ledig-tid i denne sammenheng er tid hvor CPUen kan benyttes til andre oppgaver av applikasjonen. Arkitekturen for ZLX-M avlaster CPUen vesentlig i forhold til den eldre PXG-arkitekturen. Til tross for drahjelp fra en kraftig CPU gir PXG-kontrolleren likevel dårligere ytelse enn ZLX-M. La det ikke dermed være sagt at avlastning av CPUen alltid er å foretrekke. For applikasjoner som ikke utnytter ledig CPU-tid mens opptegningen foregår vil det antagelig være en fordel å la CPUen delta aktivt i grafikk-opptegningen når man benytter kraftige maskiner, mens forholdet kan være omvendt for mindre maskiner. HP utnytter f. eks. CPUen kraftig gjennom sine CRX/HCRX-kontrollere, og Digital har nettopp lansert en ny kontroller, ZLX-L1, etter de samme prinsippene. ZLX-L1 tilhører samme familie som ZLX-M, men den mangler en del av de dedikerte grafikk-kretsene som finnes på ZLX-M1/M2. PLB-resultatene er gjennomgående høyere for denne nye kontrolleren, noe som indikerer at denne løsningen er å foretrekke ved ren grafikk-ytelse, og for applikasjoner som ikke utnytter ledig CPU-tid under opptegningen.

Arkitektur

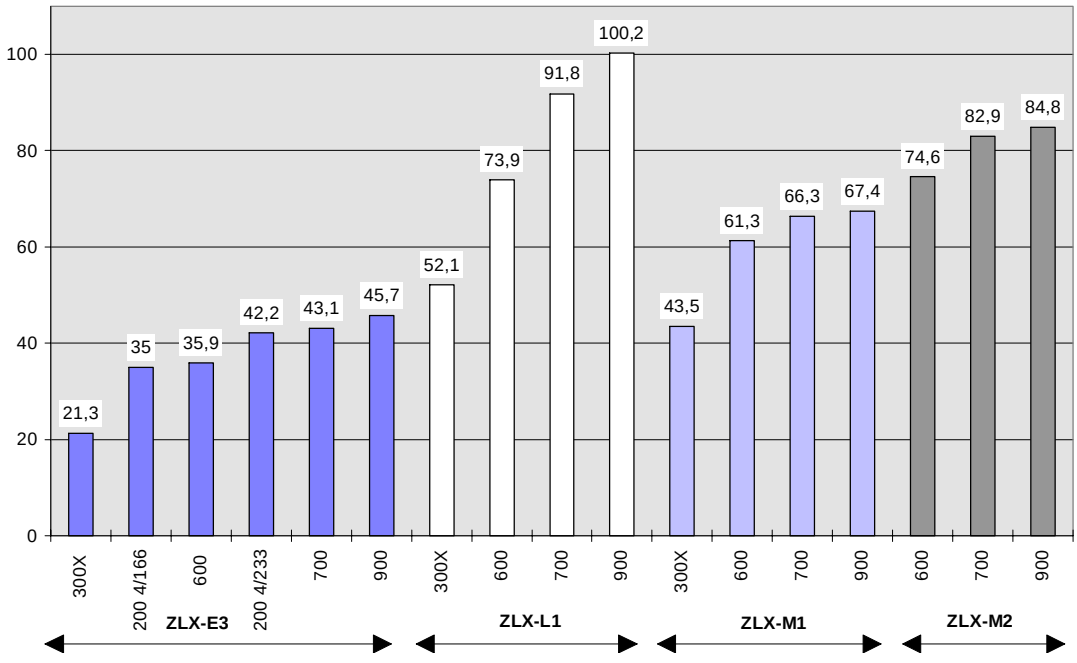
Det har vist seg praktisk umulig å produsere den perfekte kontroller, med ledende ytelse for 2D (imaging, vektor-tegning) og 3D (vektor-tegning, flate/volum-modellering), til en lav pris. Man må derfor vanligvis nøye seg med et passende kompromiss: f.eks. god 3D-ytelse, og akseptabel 2D-hastighet. De fleste leverandører tilbyr kontrollere med styrker innenfor forskjellige områder. Figur 1 og figur 2 viser hvordan Digital har valgt å løse dette for 3D-tegning. Dersom man primært har behov for vektor-tegning (PLBwire), og noe flate-modellering (PLB-surf) skal man velge en kontroller i ZLX-E-familien. Dersom behovet for flate/volum-modellering er stort bør man vurdere å velge et av medlemmene i ZLX-L/ZLX-M-familien. Andre leverandører har valgt løsninger som gir tilsvarende, eller kanskje helt forskjellige kompromisser. Dette understreker igjen viktigheten av å definere sitt eget

behov før man velger en løsning.

Som man skjønner, det er ikke tilstrekkelig å slå opp i første og beste brosjyre for å finne "Sannheten" om grafikk-ytelse! Jeg håper dette har gitt litt bakgrunn om det arbeidet som er i gang i bransjen. Det viktigste vil fortsatt være å gjennomføre en vurdering utfra egne behov, og følgelig at man velger tester som reflekterer dette behovet!



Figur 1 PLBwire-resultater for Digitals maskin/kontroller-kombinasjoner.



Figur 2 PLBsurf-resultater for Digitals maskin/kontroller-kombinasjoner.

Om å illustrere

Wolfgang Leister, Metronor AS

Innen fagområdet datagrafikk anvendes nye begreper som virtuell virkelighet, multimedia og visualiseringsteknikk. Disse heller tekniske begrepene tar mye av oppmerksomheten fra bruken av de nye mulighetene som datagrafikken gir oss. Det kan synes som om det mangler et hovedbegrep som dekker følgende områder: Grafikk til daglig bruk, reklamefilmer, animasjon, kunst, skisser, avbildninger, illustrasjoner i bøker og brosjyrer. Jeg foreslår å introdusere begrepet *å illustrere* for dette og jeg vil i denne artikkelen begrunne forslaget fra forskjellige synsvinkler ved hjelp av eksempler.

Teknikker fra 1400-talls trykke- og billedkunst, f.eks. perspektivlæren, har preget vår tenkning innen datagrafikk. Disse teknikkene er mer innskrenkende i virkemidlene enn malerkunsten og var av den grunn omfattet av vitenskapen om *maleriet*, slik det fremgår av Albrecht Dürers *Underweysung (1525)* og Leonardo da Vincis *Traktat om det å male (1498) [1]*. På den tiden ble begrepene malerkunst og geometri ansett som synonyme.

Fra *Leonardo da Vinci: Traktat om det å male, Del I, Faszikel 1, Nummer 5*:

> Malerkunstens vitenskap omfatter alle farger av flater, figurene som er kledt inn av disse og deres nærhet og avstand med tilsvarende sjatteringer etter graden av avstand. Og det er denne vitenskapen som er mor til perspektiv, dvs. læren av synslinjene.

Mens hele spekteret av geometri og farger står til rådighet i malerkunsten, medførte trykketeknikken begrensninger, ikke minst av kostnadsmessige grunner. Begrensningene var særlig tydelige før oppfinnelsen av rastereringsteknikken. Det ble dannet spesielle grafikkformer, bla. kobberstikk og tresnitt, som var egnet til å mangfoldiggjøres v.h.a. den tids trykketeknikk.

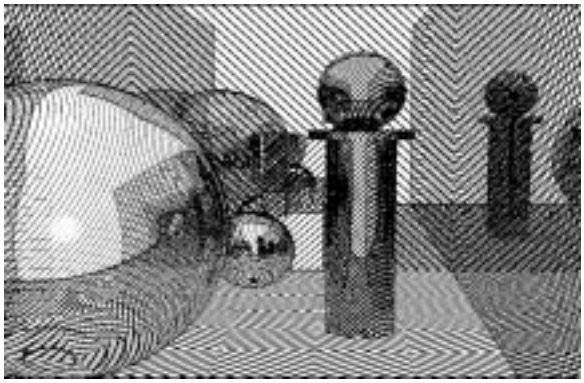
Innskrenkningen til svart-hvitt og linjegrafikk betydde ikke bare en begrensning men også en økning av pregnans i presentasjonen. En sammenligning i illustrasjoner i vitenskapelige artikler som ble skrevet rundt århundreskiftet med slike som ble fremstilt på 50-tallet (med svart-hvitt reproduksjoner - rastereringer av fotografier) viser at de gamle linjetegninger var mere uttrykksfulle.

Datagrafikk muliggjør det å etterlikne og videreutvikle forskjellige stilarter og teknikker. Desktop-Publishing og PC-nettverk øker mulighetene ytterligere. Ved nærmere øyensyn synes imidlertid datamaskinen å ikke være mer enn en elektronisk settemaskin, hvor hoveddelen bestå i å legge inn rasterbilder og skjemategninger i tekster. Også i tekster, som skal vise de nye mulighetene som ligger i teknikken, er figurene ofte laget med enkle tegneprogrammer.

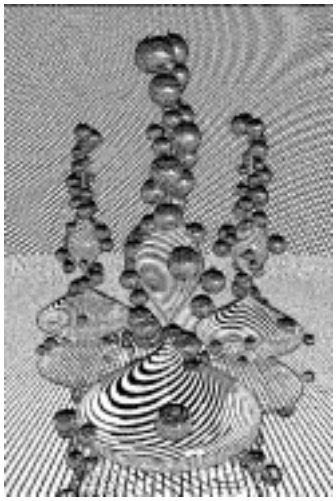
I områdene DAK eller arkitektur blir prosjekter allerede fremvist på fotorealistisk vis. Dette medfører at karakteren til det planlagte og uferdige går tapt. Aspektet ved å *illustrere* bør igjen komme mere i forgrunnen, ved å bla. fremstille figurer som ligner mer på håndtegninger eller skisser [2].

Nærmere forklaringer på begrepet *å illustrere* finner vi i et etymologileksikon [3] (utdrag oversatt og avkortet):

> **illuster:** glansfull, fornem, utsøkt. Adjektivet ble på 1800-tallet lånt fra det franske *illustre* med samme betydning, som går til-



Figur 1. Clip 9, datagenerert kobberstikk.



Figur 2. Marsbananer, datagenerert kobberstikk (Achim Stöβer).

bake til lat. *illustris*. De ordene som kommer fra lat. *lustrare* dukker opp på 1600- og 1700-tallet i Mellomeuropa som å illustrere og *illustrasjon*. Den moderne betydningen "å utsmykke en bok eller tidskrift med bilder" kommer derimot først på 1800-tallet da man begynte å lage tekstutgaver med bilder. Dette gjelder særlig for formene *illustrert* (1800-tallet) og dens substantivering en *illustrert* (1900-tallet).

For datastøttet illustrering blir det i tillegg til parametrene som brukes for å lage realistiske bilder brukt data til å fremheve eller å forsterke og individuelle egenskaper, som gir en illustrasjon en spesiell karakter og legger til tolkninger. Dette kan skje i det todimensjonale billedrom så vel som i de tredimensjonale scenebeskrivelsene.
Med metoder som brukes i de ikke-realistiske metodene i billedgenerering (nonrealistic rendering) kan det oppnås slike effekter. En mulig fremgangsmåte er å fremstille datagenererte kobberstikk [4]. Denne fremgangsmåten har overraskende nok sitt utspring i den fotorealistiske billedgenereringen[5]. Ved en utvidelse av overflatebeskrivelsene til objektene og en etter-prosessering av pikselbildet kan det genereres grafiske bilder som ligner kobberstikk. Fremgangsmåten kan dessuten utvides til å fremstille kobberstikk i farger som ligner illustrasjonene på f.eks. frimerker eller sedler.

Avslutningsvis vil jeg komme med et ønske om at design og kvalitet igjen rykker i forgrunnen og at publikasjonsverktøy vurderes ut fra disse kriteriene. Utprøving av det teknisk mulige og farget realistiske bør ikke lenger være eneste mål. Et tilbakeblikk på teknikker som allerede har vist seg å være egnet er ønskelig. Det beste fra tidligere teknikker må hentes frem og overføres til de nye mediene og fremgangsmåtene.

Litteratur

[1] Leonardo da Vinci. *Traktat von der Malerei*. Eugen Diederichs Verlag, München, 1989. Nachdruck der Ausgabe Jena, 1925.

[2] T. Strothotte, B. Preim, A. Raab, J. Schumann, and D. Forsey. How to render frames and influence people. *Computer Graphics Forum*, 13(3):C455--C466, 1994.

[3] G. Drosdowski, editor. *Duden - Etymologie (Bd. 7)*. Dudenverlag, Mannhaim, Wien, Zürich, 1989.

[4] W. Leister. Computer generated copper plates. *Computer Graphics Forum*, 13(1):69--77, 1994.

[5] W. Leister, H. Müller, and A. Stöβer. *Fotorealistische Computeranimation*. Springer-Verlag, Heidelberg, 1991.

GPGS-F hjørnet

Marianne Wallin, ViaNova AS

Brukerforum

Brukerforum for GPGS-F under MS Windows ble holdt i ViaNova's lokaler i Slependvn. 48 i Bærum onsdag 7. desember kl 17.00. Er du allerede erfaren bruker, har du tittet litt på Windows-versjonen eller kan tenke deg å ta den i bruk? Kom og delta på et slikt brukerforum og diskuter med andre brukere! Påmelding til Stein Slaatsveen, tlf. 67 56 46 00 eller email steins@oslonett.no.

HP-GL2 driver

Vi nevnte i forrige Norsigd Info at det var utviklet en ny HP-GL2 driver. Nå er den tilgjengelig på alle plattformer og koster kr. 3000,-. HP-GL2 driveren går mot alle HP's nye penn- og rasterplottere samt laserskrivere. På flere plattformer vil Postscript- og HP-GL2 driverene bli de viktigste skriverdriverene framover.

GPGS utover landegrensen

Applikasjoner med GPGS-grafikk brukes allerede langt utover Norges grenser, og nå vurderes det å legge inn det russiske alfabetet i GPGS for applikasjoner som skal benyttes i Russland!

Antall GPGS-brukere

Noen er bekymret over at GPGS har en liten brukermasse. La oss derfor ta en titt på hva som fantes av GPGS lisenser i 1993. Unix-versjon 33, VAX 15, ND-500 10, ND-100 3, IBM OS/MVS 1, IBM VM/CMS 2 og DOS 15. MS Windows versjonen er først tatt i bruk i år, og vi regner med å få flere lisenser i løpet av 94/95. På sluttbrukernivå er det tusenvis av brukere, da mange av GPGS-applikasjonene er programvare som selges!

Vi vil gjerne ha hjelp

Har du informasjon, tips eller spørsmål du vil vi skal ta opp i GPGS-hjørnet? Ring oss, så skal vi gjøre vårt for at denne spalten blir interessant å lese!

Grafikk hjørnet

- Frequently Asked Questions about OpenGL (TM)

Mason Woo, Silicon Graphics

Q1: What is OpenGL?

Q2: What is the relationship between IRIS GL and OpenGL?

Q3: What does the .gl or .GL file format have to do with OpenGL?

Q4: Which vendors are supporting OpenGL?

Q5: What OpenGL implementations are available?

Q6: What documentation is available for OpenGL? (A bibliography of OpenGL documents is listed here.) Where is a written version of the OpenGL specification? Where is the source code which accompanies that documentation?

Q7: Why doesn't SGI provide a free implementation of OpenGL?

Q8: What are the conformance tests?

Q9: How do I contribute OpenGL code examples to a publicly accessible archive?

Q10: Will OpenGL code be source code or binary code compatible with IRIS GL code?

Q11: Why should I port my IRIS GL application to OpenGL?

Q12: How much work is it to convert an IRIS GL program to OpenGL? What are the major differences between them?

Q13: How does a university or research institution acquire access to OpenGL source code?

Q14: Who needs to license OpenGL? Who doesn't? How is a commercial license acquired?

Q15: How is the OpenGL governed? Who decides what changes can be made?

Q16: Who are the current ARB members?

Q17: What is the philosophy behind the structure of the ARB?

Q18: How does the OpenGL ARB operate logistically? When does the ARB have meetings?

Q19: How do additional members join the OpenGL ARB?

Q20: So if I'm not a member of the ARB, am I shut out of the decision making process?

Q21: What is the OpenGL Advisory Forum?

Q22: Are ARB meetings open to observers?

Subject: Q1: What is OpenGL?

OpenGL is the software interface for graphics hardware that allows graphics programmers to produce high-quality color images of 3D objects. OpenGL is a rendering only, vendor neutral API providing 2D and 3D graphics functions, including modelling, transformations, color, lighting, smooth shading, as well as advanced features like texture mapping, NURBS, fog, alpha blending and motion blur. OpenGL works in both immediate and retained (display list) graphics modes. OpenGL is window system and operating system independent. OpenGL has been integrated with Windows NT and with the X Window System under UNIX. Also, OpenGL is network transparent. A defined common extension to the X Window System allows an OpenGL client on one vendor's platform to run across a network to another vendor's OpenGL server.

Subject: Q2: What is the relationship between IRIS GL and OpenGL?

IRIS GL is the predecessor to OpenGL. After other implementors had experience trying to port the IRIS GL to their own machines, it was learned that the IRIS GL was too tied to a specific window system or hardware. Based upon consultations with several implementors, OpenGL is much more platform independent.

IRIS GL is being maintained and bugs will be fixed, but SGI will no longer add enhancements. OpenGL is now the strategic interface for 3-D computer graphics.

Subject: Q3: What does the .gl or .GL file format have to do with OpenGL?

.gl files have nothing to do with OpenGL. It's a file format for images, which has no relationship to IRIS GL or OpenGL.

Subject: Q4: Which vendors are supporting OpenGL?

OpenGL is supported by SGI and many other hardware vendors. As of April, 1994, OpenGL has been licensed to:

AT&T, Cirrus Logic (which purchased the A1060 technology from Austek Microsystems), Cray Research, Daikin, Digital Equipment, Du Pont Pixel Systems (supporting Sun and a Du Pont Pixel graphics accelerator board), Evans & Sutherland, Harris Computer, Hitachi, IBM, the Institute for Information Industry, Intel, Intergraph, Kendall Square Research, Kubota Pacific, Media Vision, Microsoft, miro, NEC, Portable Graphics (formerly Nth Portable Graphics; supporting Sun and HP), RasterOps, SPEA, Samsung, Sony, and Univel.

The machines supported by OpenGL licensees constitute over 95% of the graphics workstation marketplace, as well as the majority of the PC market.

Subject: Q5: What OpenGL implementations are available?

On SGI machines, OpenGL is available on both Onyx and Indy. Availability on at Indigo, Indigo 2, Crimson (except VGX(T)), and other SGI workstations will be part of the next release of IRIX. The graphics options covered by OpenGL include the Entry system, XS, XS24, XZ, XL, Elan, Extreme, VTX, Reality Engine, and Reality Engine 2.

Digital Equipment has shipped its first OpenGL product as part of the Open3D 2.0 layered product for Alpha AXP OSF/1. The server support in this release is only for the PXG graphics adaptor, but the client libraries are of course universal if thats all you need. For information on ordering the Open3D 2.0 product from Digital contact your Digital sales representative.

SGI does not speak for any other company. However, this space is available for any company who wishes to state status reports or release dates for their OpenGL implementation. Please send e-mail to woo@sgi.com to add to this section.

Subject: Q6: What documentation is available for OpenGL?

(A bibliography of OpenGL documents is listed here.) Where is a written version of the OpenGL specification? Where is the source code which accompanies that documentation?

A 2 volume set, The OpenGL Technical Library, is being published by Addison-Wesley. The OpenGL Reference Manual is ISBN 0-201-63276-4. The OpenGL Programming Guide is ISBN 0-201-63274-8.

You can purchase the books in extremely large volume by calling Robert Shepard of Addison-Wesley (617) 944-3700 ext 2435.

The man pages for the OpenGL API, its Utility Library (GLU), and the X server extension API (GLX) and a PostScript version of the OpenGL specification are available via anonymous, public ftp, on the machine sgigate.sgi.com in ~ftp/pub/opengl/doc. A paper by Allen Akin comparing PEX 5.1 to OpenGL is also in this directory.

The spec (Spec.tar.Z) is contained in a directory with several files which have been tar'd and compressed. The OpenGL, OpenGL Utility Library, X extensions and GLX protocol specifications are all here. Please read the README file in the directory, which explains the copyright and trademark rules for usage of the specification. Possession of the OpenGL Specification does not grant the right to reproduce, create derivative works based on or distribute or manufacture, use or sell anything that embodies the specification without an OpenGL license from SGI.

You can also get the source code examples which are found in the OpenGL Programming Guide via anonymous, public ftp on sgigate.sgi.com in the file ~ftp/pub/opengl/opengl.tar.Z

What follows is a bibliography of articles, books, and papers written about OpenGL.

Magazine articles and books

Davis, Tom. "Moving to OpenGL," IRIS Universe, Number 25, Summer, 1993.

Glazier, Bill. "The 'Best Principle': Why OpenGL is emerging as the 3D graphics standard," Computer Graphics World, April, 1992.

Karlton, Phil. "Integrating the GL into the X environment: a high performance rendering extension working with and not against X," The X Resource: Proceeding of the 6th Annual X Technical Conference, O'Reilly Associates, Issue 1, Winter, 1992.

Kilgard, Mark J. "OpenGL & X: An Introduction," The X Journal. November-December, 1993, page 36-51.

Kilgard, Mark J. "Using OpenGL with Xlib," The X Journal. January-February, 1994, page 46-65.

Neider, Jackie, Tom Davis, and Mason Woo, OpenGL Programming Guide: The Official Guide to Learning OpenGL, Release 1, Addison-Wesley, Reading, Massachusetts, 1993.

OpenGL Architecture Review Board, OpenGL Reference Manual: The Official Reference Document for OpenGL, Release 1, Addison-Wesley, Reading, Massachusetts, 1992.

"OpenGL Programs a New Horizon for Sun," SunWorld, January, 1994, page 15-17.

Technical reports

Akin, Allen. Analysis of PEX 5.1 and OpenGL 1.0. Technical report, Silicon Graphics Computer Systems, Mountain View,
California, August 3, 1992.

Segal, Mark and Kurt Akeley. The OpenGL Graphics System: A Specification. Technical report, Silicon Graphics Computer Systems, Mountain View, California, 1992, revised 1993.

Segal, Mark and Kurt Akeley. The OpenGL Graphics Interface. Technical paper, Silicon Graphics Computer Systems, Mountain
View, California, 1993.

Subject: Q7: Why doesn't SGI provide a free implementation of OpenGL?

The intent of licensing is to enhance conformity and portability of applications code. In the days before OpenGL, IRIS GL was supported differently and enhanced differently for each machine on which it was implemented. This included machines from IBM, HP, Sun, as well as SGI. Applications developers found this to be a nightmare, and convinced
SGI that OpenGL implementations needed to be more consistent. This led us to develop specifications and conformance tests, and to seek some way to ensure that OpenGL implementors would honor them. There are only a few legal mechanisms that can be used for this purpose, and licensing is one of the most effective and widely accepted. SGI thinks the lack of a public-domain implementation is a trade-off for a guarantee of consistent OpenGL implementations from many different vendors, and that in the long run this is the best interests of application developers, who seek ease of portability, and end-users, who seek to be able to choose among hardware vendors for their chosen
software solution.

Licensing money goes to creating conformance tests to control variants and subsets, maintaining the specification and sample implementation. Because of the overhead of having the dozen or so engineers working
on OpenGL at SGI, SGI is not making money from the OpenGL effort. Licensing is one way to help offset those costs.

Subject: Q8: What are the conformance tests?

The conformance tests are a suite of programs which judge the success of an OpenGL implementation. Each implementor is required to run these tests and pass them in order to call their implementation with the trademark OpenGL. Passing the conformance tests ensures source code compatibility of applications across all OpenGL implementations.

Subject: Q9: How do I contribute OpenGL code examples to a publicly accessible archive?

To contribute to the public OpenGL archive, send mail to opengl-contrib@sgi.com. Your mail should contain:

- The material to be archived, or instructions for obtaining it.
- An announcement suitable for posting to comp.graphics.opengl.

SGI will place the material in the OpenGL/contrib directory on sgi.com and post the announcement to this newsgroup.

To retrieve something from the archive, use anonymous ftp to sgi.com. Once connected, cd to the directory OpenGL. (Case is significant.) Currently there are two subdirectories:

- doc: Manual pages for OpenGL and related libraries.
- contrib: Contributions from the public.

Note that all contributions are distributed as-is; neither SGI nor the other companies on the OpenGL Architecture Review Board make any legally valid claims about the robustness or usefulness of this software.

If you do not have access to anonymous ftp, consider using an "ftp-by-mail" server. For information on one such server, send mail to ftpmail@decwrl.dec.com with a message body containing only the word "help".

Subject: Q10: Will OpenGL code be source code or binary code compatible with IRIS GL code?

OpenGL code is neither binary nor source code compatible with IRIS GL code. It was decided to bite the bullet at this time to make OpenGL incompatible with IRIS GL and fix EVERYTHING that made IRIS GL difficult to port or use. For example, the gl prefix has been added to every command: glVertex(), glColor(), etc.

Subject: Q11: Why should I port my IRIS GL application to OpenGL?

SGI will be maintaining the old IRIS GL, but not enhancing it. OpenGL is the API of choice on all new SGI machines.

OpenGL has no subsets. You can use the same functionality on all machines from SGI or from other vendors.

OpenGL is better integrated with the X Window System than the old IRIS GL. For example, you can mix OpenGL and X or Display PostScript drawing operations in the same window.

The OpenGL naming scheme, argument list conventions, and rendering semantics are cleaner than those of IRIS GL. This should make OpenGL code easier to understand and maintain.

Subject: Q12: How much work is it to convert an IRIS GL program to OpenGL? What are the major differences between them?

There is a fair amount of work, most of which is in substituting for window management or input handling routines, for which the equivalents are not OpenGL, but the local window system, such as the X Windows System or Windows NT. And all routine names have changed, at least, minimally; for example: ortho() is now glOrtho().

To help ease the way, port to "mixed model" right away, mixing the X Window System calls to open and manage windows, cursors, and color maps and read events of the window system, mouse and keyboard. You can do that now with IRIS GL, if you are running IRIX 4.0.

In the X Window System, display mode choices (such as single or double buffering, color index or RGBA mode) must be declared before the window is initially opened. You may also substitute for other IRIS GL routines, such as using a OSF/Motif menu system, in place of the IRIS GL pop-up menus. You should use glXUseXFont(), whenever you were using the font manager with IRIS GL.

Tables for states such as lighting or line and polygon stipples will be gone. Instead of using a def/set or def/bind sequence to load a table, you turn on the state with glEnable() and also declare the current values for that state.

Colors are best stored as floating point values, scaled from 0.0 to 1.0 (0% to 100%). Alpha is fully integrated in the RGBA mode and at least source alpha will be available on all OpenGL implementations. OpenGL will not arbitrarily limit the number of bits per color to 8. Clearing the contents of buffers

no longer uses the current color, but a special "clearing" color for each buffer (color, depth, stencil, and accumulation).

The transformation matrix has changed. In OpenGL, there is no single matrix mode. Matrices are now column-major and are post-multiplied, although that does NOT change the calling order of these routines from IRIS GL to OpenGL. OpenGL's glRotate*() now allows for a rotation around an arbitrary axis, not just the x, y, and z axes. lookat() of IRIS GL is now gluLookAt(), which takes an up vector value, not merely a twist. There is no polarview() in OpenGL, but a series of glRotate*()s and glTranslate*()s can do the same thing.

There are no separate depth cueing routines in OpenGL. Use linear fog.

Feedback and selection (picking) return values, which are different from those found on any IRIS GL implementation. For selection and picking, depth values will be returned for each hit. In OpenGL, feedback and selection will now be standardized on all hardware platforms.

Subject: Q13: How does a university or research institution acquire access to OpenGL source code?

There is a university/research institution licensing program. A university license entitles the institution to generate binaries and copy them anywhere, so long as nothing leaves the institution. The OpenGL source and derived binaries can only be used for non-commercial purposes on-campus. A university license costs $500 and can be obtained by contacting woo@sgi.com.

Subject: Q14: Who needs to license OpenGL? Who doesn't? How is a commercial license acquired?

Companies which will be creating or selling binaries of the OpenGL library will need to license OpenGL. Typical examples of licensees include hardware vendors, such as Digital Equipment, IBM, and Silicon Graphics who would distribute OpenGL with the system software on their workstations or PCs. Also, some software vendors, such as Du Pont Pixel, MediaVision, and Portable Graphics, have a business in creating and distributing versions of OpenGL, and they need to license OpenGL.

Applications developers do NOT need to license OpenGL. If a developer wants to use OpenGL, that developer needs to obtain copies of a linkable OpenGL library for a particular machine. Those OpenGL libraries may be bundled in with the development and/or run-time options or may be purchased from a third-party software vendor, without licensing the source code or use of the OpenGL(TM) trademark.

Since many implementations will be a shared library on a hardware platform, the royalty sometimes will be charged for each hardware platform. In those cases, it would not be charged for each application which used OpenGL.

In general, licensing a source code implementation of OpenGL would not be useful for an application developer, because the binary created from that implementation would not be accelerated and optimized to run on the graphics hardware of a machine.

If you need a license or would like more information, call Mason Woo at (415) 390-4205 or e-mail him at woo@sgi.com. There are licenses available restricted to site (local) usage, or permitting redistribution of binary code. The limited source license provides a sample implementation of OpenGL for $50,000. The license for commerical redistribution of OpenGL binaries has two most commonly chosen levels. Level 1 costs $25,000. Level 2 costs $100,000, and includes the sample implementation of OpenGL.

Both levels require a $5 royalty for every copy of the OpenGL binary, which is redistributed.

Subject: Q15: How is the OpenGL governed? Who decides what changes can be made?

OpenGL is controlled by an independent board, the Architectural Review Board (ARB). Each member of the ARB has one vote. The founding members of the ARB are DEC, IBM, Intel, Microsoft, and SGI. Additional members will be added over time. The ARB governs the future of OpenGL, proposing and approving changes to the specification, new releases, and conformance testing.

Subject: Q16: Who are the current ARB members?

In alphabetical order: Digital Equipment, IBM, Intel, Microsoft, and Silicon Graphics.

Subject: Q17: What is the philosophy behind the structure of the ARB?

The ARB is intended to be able to respond quickly and flexibly to evolutionary changes in computer graphics technology. The ARB is currently "lean and mean" to encourage speedy communication and decision-making. Its members are highly motivated in ensuring the success of OpenGL.

Subject: Q18: How does the OpenGL ARB operate logistically? When does the ARB have meetings?

ARB meetings are held about once a quarter. The meetings rotate among sites hosted by the ARB members. To learn the date and place of the next OpenGL ARB meeting, watch the news group comp.graphics.opengl for posting announcing the next "OpenGL Advisory Forum/OpenGL ARB meetings" or e-mail opengl-secretary@sgi.com and ask for the information.

Meetings are run by a set of by-laws, which are currently being approved. When they are approved, the by-laws will be publicly available for inspection.

Minutes to the ARB meeting are posted to comp.graphics.opengl.

Subject: Q19: How do additional members join the OpenGL ARB?

The intention is that additional members may be added on a permanent basis or for a one-year term. The one-year term members would be voting members, added on a rotating basis, so that different viewpoints (such as ISV's) could be incorporated into new releases. Under the currently proposed (but not yet ratified) by-laws, SGI formally nominates new members.

Subject: Q20: So if I'm not a member of the ARB, am I shut out of the decision making process?

There are many methods by which you can influence the evolution of OpenGL.

1) Contribute to the comp.graphics.opengl news group. Most members of the ARB read the news group religiously.

2) Contact any member of the ARB and convince that member that your proposal is worth their advocacy. Any ARB member may present a proposal, and all ARB members have equal say.

3) Come to OpenGL Advisory Forum and speak directly to ARB in person.

Subject: Q21: What is the OpenGL Advisory Forum?

Preceding every ARB meeting will be a meeting of the OpenGL Advisory Forum.

Members of the ARB will attend this meeting to listen to proposals and discussions. If you want to attend, you want to notify the Secretary of the ARB (e-mail opengl-secretary@sgi.com).

The Advisory Forum is intended to start as an informal, self-governing group of people, highly interested in OpenGL. The ARB will invite a representative of the Advisory Forum to attend the ARB meetings, in a non-voting capacity. Advisory Forum members can be candidates for both short-term and permanent membership in the ARB. The Advisory Forum is encouraged to formulate its own structure and rules and move into any direction it wants.

Minutes to the Advisory Forum meeting are posted to comp.graphics.opengl.

Corporations, universities, or individuals can be members of the Advisory Forum.

OpenGL Advisory Forum meetings are quarterly and are usually held a day before each OpenGL ARB meeting (see Question 18).

Subject: Q22: Are ARB meetings open to observers?

The ARB meeting will be open to observers, but we want to keep the meeting small. For the next meeting, the ARB will allow the Advisory Forum to nominate five representatives of the Advisory Forum to attend the ARB meeting, in a non-voting capacity. At any time, the ARB reserves the right to change the number of observers.

Aktivitetskalender

Hva skjer når og hvor?

December 1994

13	**User Interface Strategies '95: The Information Superhighway**, Live satellite broadcast, December 13, 1994 - 11 AM - 5 PM Eastern, Presented by Ben Shneiderman, University of Maryland, Frank Stein, IBM, H. Rex Hartson & Deborah Hix, Virginia Tech Kent Norman, University of Maryland, USA.
14-16	**Beyond Visualization**, Workshop on Information Science and Medical Images, Singapore.

February 1995

8-9	**VIRTUAL REALITY '95 - Applications and Trends**, The Fraunhofer Institutes IPA and IAO.
14.- 18	**The Third International Conference in Central Europe on Computer Graphics and Visualization,** 95 WSCG '95 (Winter School of Computer Graphics 95) to be held in Czech Republic.
23-24	**The first International Computer Animation Conference**, Snowbird, Utah, USA.

April 1995

19-21	**Computer Animation '95**, The seventh conference on Computer Animation, organized jointly by the University of Geneva, the Swiss Federal Institute of Technology and CGS, Geneva, Switzerland.

May 1995

15-19	**Graphics Interface '95**, Quebec City, Quebec, Canada.
24-27	**Intermedia Asia '95**, The definitive convention for the multimedia industry, Singapore.

June 1995

12-14 **6th Eurographics Workshop on Rendering**, Dublin, Ireland.

18-21 **ED - MEDIA 95**, World Conference on Educational Multimedia and Hypermedia, Graz, Austria.

July 1995

5-8 **Tenth International Conference on Mathematical and Computer Modelling and Scientific Computing**, Boston Marriott Hotel, Copley Place Boston Massachusetts, USA.

August 1995

14-18 **EHCI'95**, Working Conference on Engineering for Human-Computer Interaction, Grand Targhee Resort, Wyoming, USA.

October 1995

22-25 **CAD/Graphics'95**, The Fourth International Conference on CAD & CG, Wuhan, China.

November 1995

14-17 **The International Conference on Multi-Media Modelling**, Singapore.

Hva er NORSIGD?

NORSIGD - Norsk samarbeid innen grafisk databehandling - ble stiftet 10. januar 1974. NORSIGD er en ikke-kommersiell forening med formål *å fremme bruken av, øke interessen for, og øke kunnskapen om grafisk databehandling i Norge.*

Foreningen er åpen for alle enkeltpersoner, bedrifter og institusjoner som har interesse for grafisk databehandling. NORSIGD har, per juni 1994, 64 institusjonsmedlemmer og 37 personlige medlemmer. Medlemskontingenten er 1.000 kr per år for institusjoner. Institusjonsmedlemmene er stemmeberettiget på foreningens årsmøte, og kan derigjennom påvirke bruken av foreningens midler.

Personlig medlemskap koster 250 kr per år. Personlige medlemmer får tilsendt medlemsbladet *NORSIGD Info*, og er berettiget til redusert kontingent ved medlemskap i vår europeiske samarbeidsorganisasjon *Eurographics*.

Alle medlemmer får tilsendt medlemsbladet NORSIGD Info 4 ganger per år.

Interesseområder

NORSIGD er et forum for alle som er opptatt av grafiske brukergrensesnitt, uavhengig av om basisen er *The X Window System, Microsoft Windows* eller andre systemer.

NORSIGD arrangerer møter og seminarer, formidler informasjon fra internasjonale fora og distribuerer fritt tilgjengelig programvare. I tillegg formidles kontakt mellom brukere og kommersielle programvareleverandører.

NORSIGD har lang tradisjon for å støtte opp om bruk av datagrafikk. Foreningen bidrar til spredning av informasjon ved å arrangere møter, seminarer og kurs for brukere og systemutviklere.

GPGS-F

GPGS-F er en 2D- og 3D grafisk subrutinepakke. GPGS-F er maskin- og utstyrsuavhengig. Det vil si at et program utviklet for én maskin med f. eks. bruk av plotter, kan flyttes til en annen maskin hvor plotteren er erstattet av en grafisk skjerm uten endringer i de grafiske rutinekallene. Det er definert grensesnitt for bruk av GPGS-F fra FORTRAN og C.

Det finnes versjoner av GPGS-F for en rekke forskjellige datamaskiner, fra IBM stormaskiner til Unix arbeidsstasjoner og PC. GPGS-F har drivere for over 50 forskjellige typer utstyr (plottere, skjermer o. l.). Data kan utveksles mot CGM metafil.

GPGS-F eies av NORSIGD, og leies ut til foreningens medlemmer.

Eurographics

Eurographics ble grunnlagt i 1981 og har over 1.000 medlemmer over hele verden. Organisasjonen utgir et av verdens fremste fagtidsskrifter innen grafisk databehandling, Computer Graphics Forum. Forum sendes medlemmene annen hver måned. Eurographics arrangerer årlig en konferanse som inneholder seminarer, utstilling, kurs og arbeidsgrupper.

NORSIGD samarbeider med Eurographics. Personlige medlemmer i NORSIGD får 20% rabatt på medlemskap i Eurographics, og vi formidler informasjon om aktuelle aktiviteter og arrangementer som avholdes i Eurographics regi.

Kombinert personlig medlemskap i Eurographics og NORSIGD koster kr 375 per år, inkludert NORSIGD-rabatt. Det beregnes en registreringsavgift ved innmelding. Institusjonsmedlemskap i Eurographics er priset avhengig av antall kontaktpersoner i institusjonen.
Institusjonsmedlemmer får flere eksemplarer av Forum, og de får rabatt ved deltagelse på EG-konferansene. Eurographics kan bistå ved å holde kurs og seminarer hos institusjonsmedlemmer. Institusjonsmedlemmer kan også presentere seg i Forum, og de får tilgang til Eurographics' adresseliste. Det er en egen stand for institusjonsmedlemmene på EG-konferansene.

Hver kontaktperson har ellers de samme rettigheter som personlige Eurographics medlemmer.

NORSIGD v/Marianne Wallin
ViaNova AS
Postboks 53
1312 SLEPENDEN

Returadresse:

NORSIGD v/Nils Thune
Metronor AS
Postboks 238
1360 Nesbru

Styret i NORSIGD 1994

Funksjon	Adresse	Telefon	email
Leder	Nils Thune Metronor AS Postboks 238 1360 NESBRU	66 98 28 90 (sentralbord) 66 98 28 95 (fax)	thune@oslonett.no
Kasserer:	Marianne Wallin ViaNova AS Postboks 53 1312 SLEPENDEN	67 56 46 10 +45 (direkte) 67 56 46 00 (sentralbord) 67 56 46 20 (fax)	
Sekretær:	Jens Holwech Digital Equipment Corporation A/S Postboks 6401 Etterstad 0604 OSLO	22 76 85 43 (direkte) 22 76 85 00 (sentralbord) 22 76 86 22 (fax)	Jens.Holwech@nwo.mts.dec.com X.400: C=no A=telemax P=digital S=holwech G=jens
Styremedlem:	Knut Hasund SINTEF Oslo Postboks 124 Blindern 0314 OSLO	22 06 76 76 (direkte) 22 06 73 00 (sentralbord) 22 06 73 50 (fax)	Knut.Hasund@si.sintef.no
Varamedlem:	Reidar Rekdal Det Norske Veritas Sesam a.s Postboks 300 1322 HØVIK	67 57 73 18 (direkte) 67 57 72 50 (sentralbord) 67 57 72 72 (fax)	rre@sesam.dnv.no
Varamedlem:	Ketil Aamnes SINTEF-SIMA 7034 TRONDHEIM	73 59 70 54 (direkte) 73 59 29 71 (fax)	ketil@sima.sintef.no

Svarkupong

☐ Innmelding - Institusjonsmedlem ☐ Innmelding - Personlig medlem ☐ Ny kontaktperson ☐ Eurographics innmelding - Personlig (forutsetter personlig medlemskap i NORSIGD) ☐ Eurographics innmelding - Institusjon ☐ Adresseforandring	Navn:..... Firma:..... Gateadresse:..... Postadresse: ..... Postnummer/sted:..... ..... Telefon .....