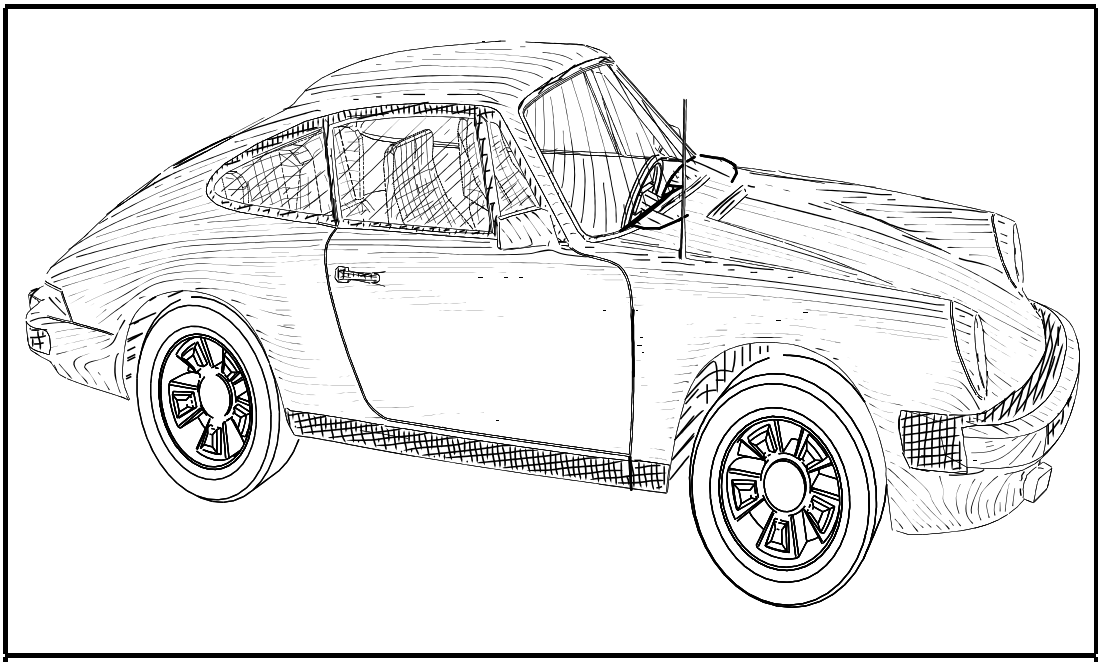
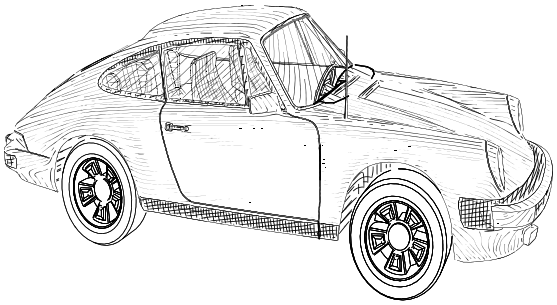
NORSIGD INFO

Nummer 2 1998



NORSK SAMARBEID INNEN GRAFISK DATABEHANDLING

ISSN 0803-8317



Om forsiden
Bildet på forsiden viser en datagenerert linjetegning av et kjent bilmerke. Den illustrerer hvordan linjetegninger som har transparente flater kan lages automatisk. Bildet er Figur 6b. til artikkelen *Hamel et.al.* i denne utgaven.

Hilsen fra styret

Kjære medlemmer,

Vårt 25-års jubileum nærmer seg, og det er på tide å begynne å skrive festtalene. Men hva skal vi skrive om? Skal vi sole oss i gamle dagers glans, da GPGS har vært en av de første grafikkpakkene som dekket brukernes behov? Eller skal vi se på det faktum at interessen daler fordi alle operativsystemer har grafikkfunksjonalitet innebygd?

Studentene har idag problemer å finne veiledere innen datagrafikk, fordi ingen av universitetene satser helhjertet på området. Så er vel alle interessante emner utforsket? Eller skjer det spennende ting som peker inn i fremtiden?

Den årlige SIGGRAPH konferansen er en slags pulsmåler innen datagrafikk. To som har besøkt SIGGRAPH rapporterer om sine inntrykk. På GPGS seminaret i fjor ble det presentert hvordan GPGS og Visual Basic integreres. En utvidet beskrivelse på hvordan dette gjøres i praksis har vært et ønske fra mange GPGS brukere. Vi har også en forskningsartikkel om grafisk abstraksjon. Nyheter innen GPGS og Grafikkhjørnet avrunder denne utgaven.

Hilsen,

Wolfgang Leister

NORSIGD Info
– medlemsblad for NORSIGD

Utgitt av:	NORSIGD
Ansvarlig:	Wolfgang Leister Norsk Regnesentral Postboks 114 Blindern 0314 OSLO
ISSN:	0803-8317
Utgivelser:	1998: 20/5 20/9 20/12
Annonsepriser:	Helside kr 5 000 Halvside kr 2 500
Oversettelser:	Wolfgang Leister
Layout:	Wolfgang Leister LaTeX2ε

Innhold

Reisebrev fra SIGGRAPH'98

Svein Løvmoen Ellingsen og Fredrik Viken, ViewTech ASA

Siggraph feiret i år 25 års-jubileum, og konferansen ble denne gang arrangert i Orlando, Florida. Konferansen inneholdt kurs, paneldebatter, paperpresentasjoner og en stor utstilling hvor hardware-leverandører og software-leverandører stiller ut på amerikansk manér. Som debutanter på SIGGRAPH var hovedinntrykket fra konferansen meget positivt. Vi var spesielt godt fornøyd med utbyttet fra noen av de kursene vi deltok på. Nedenfor følger en beskrivelse av våre inntrykk fra konferansen.

SIGGRAPH Course No. 2: Exploring Gigabyte Data Sets in Real Time: Algorithms, Data Management and Time-Critical Design. Kurset tok for seg visualisering av store mengder av vitenskapelige data, særlig CFD-simuleringer. Streamlines, isoflater og kuttplan var sentralt i presentasjonene. Steve Bryson, NASA Ames, ga en innføring i teknikker for CFD-visualisering. Så fikk vi en mer teoretisk innføring til en del spesialtilfeller (feature detection) innen CFD fra Bob Haimes (MIT): Deteksjon av vortex cores, sjokk, og grensesjikt. Her ble det mye ligninger og matematisk teori, ikke så mye praktisk kjøtt på beinet. Etter det kom Michael Cox (NASA Ames) med en del betraktninger rundt Data Management av store (100+ GB) datamodeller. Her fikk vi lære om pageing, segmentation og indexing, samt memory hierarchy, osv. I følge Cox er det ikke lønnsomt å benytte kommersielle databaser til å lagre resultater fra vitenskapelige beregninger for visualisering i dag. Det må lages properitære løsninger for å få ønsket ytelse. Til slutt i kurset var det en seksjon med Time-Critical Design. Her ga Bryson oss en innføring i hvordan man bør lage systemer med krav til responstid, både frame-rate og beregningstid (latency).

SIGGRAPH Course No. 17: Advanced Graphics Programming Techniques Using OpenGL. Vi siktet oss inn på seksjonen om visualisering av vitenskapelige data. Denne seksjonen omhandlet i første rekke volumvisualisering, og noen nyttige poenger kom frem: For det første bruk av et aksekors som 2D tekstur for å legge lengde/breddegrader på en geometri. Vi fikk en beskrivelse av hvordan en kan generere konturlinjer ved hjelp av tekstur. Resten av kurset var viet special effects og var interessant, men kanskje ikke så nyttige for oss i dag.

SIGGRAPH Course No. 20: Real-time Graphics for Visual Simulation: Advanced Techiques from the Top Down. Som tittelen antyder dreide det seg her mye om simulatormiljøer. Den delen av kurset vi overvar, dreide seg i hovedsak om oppbygging av terrengdatabaser for slike simulatorer. Dan Brockway fra Paradigm Simulation, Inc., snakket om terrenggenerering for flysimulatorer, og de spesielle hensyn en må ta her. Spesielt krevende er det faktum at man må bygge opp data for tre forskjellige sensor-systemer; nemlig visuelt, infrarødt og radar. Og her er det forskjellige krav til både presisjon og datagrunnlag. Etter Brockway, kom en Mr. Lewis fra Evans and Sutherland, som steppet inn for Ronald G. Moore. Han snakket om terrengdatabaser for bakkebaserte simulatorer, for f.eks. tank-krigføring. Her har man helt andre kriterier å forholde seg til, blant annet det faktum at når man befinner seg på bakken, så blir terrenget mye mer enn bare en kulisse, som den ofte kan betraktes som fra fly. Deretter snakket Stephen Gersuk fra Paradigm om perspektivkorreksjoner for ikke-plane displayer.

SIGGRAPH Course No. 29: Developing High-Performance Graphics Aplication for the PC Platform. Stort sett et meget interessant kurs. Tok for seg visualisering på Consumer PCs og på Workstations. Gary Tarolli (3DFx) snakket først om optimalisering på Consumer PCs. Hovedkonklusjonen var at man bør minimalisere state-changes og samtidig kjenne platformen man utvikler på, slik at man kan utnytte den. Holde ting innen L1/L2 cache, finne bottlenecks. Så fortalte Brian Hook (id Software) om utvikling av høyytelse software (Doom, Quake etc.) på Consumer PCs. Ikke så mye nyttig her, men interessant. Etter pausen kom Rich Ehlers (Evans and Sutherland Comp. Corp.)

med er redgjørelse om akselleraasjon på Workstation PCs. Dette var lite engasjerende, men lærte at båndbredde mellom grafikk, prosessor og minne er viktig for optimalisering på Workstation PCs. Også minimalisering av statechanges er viktig, ofte viktigere en på low-end utstyr. Etter dette snakket Bill Lorensen (GE) om vitenskapelig visualisering på PC. De brukte Visualization Toolkit. Ikke noe revolusjonerende, men interessant å høre at de er på vei over til PC, og er glade for at spillindustrien driver utviklingen på PC markedet. Mest eksempler fra medisin, der de kombinerer volumrendering med normal polygonvisualisering. Tilslutt var det en seksjon om optimalisering på PC, og hovedkonklusjonen er bruk triangler og tri-strips, minimaliser state changes og ikke clear mer enn du må. Alt i alt en interessant dag, med en god del nyttige småtips, samt oppdatering på hvor utviklingen går videre.

SIGGRAPH Course No. 35: Interactive Visualization and Web-Based Exploration in the Physical and Natural Sciences. Fokus i dette kurset lå på de spesielle hensyn en må ta når en skal visualisere data som kan komme fra flere, gjerne ulike kilder. Dette er en problemstilling som blir mer og mer aktuell etter som det blir vanligere å kunne tilby innsyn i forskjellig datamateriale f.eks. gjennom webtjenester. Et eksempel på dette er det kursorganisator Theresa-Marie Rhyne (Lockheed Martin/U.S. Environmental Protection Agency Visualization Center) viste i et case-study. Ved EPA har man utviklet et system som kan hente ut geografiske data fra U.S. Geographical Surveys ARC/Info database, og kople dette med miljødata, alt via et web-grensesnitt.

Kurset begynte med Bill Hibbard (University of Wisconsin, Madison), som snakket om datamodellering i forbindelse med systemer som skal kunne behandle og visualisere data på en slik måte som nevnt over. Det ble påpekt i hele kurset at det er viktig å ha atskilte data- og displaymodeller i slike tilfeller, slik at fremvisningen av dataene ikke avhenger av kilden, og at kilden kan være nettopp heterogen.

Lloyd Treinish (IBM T.J. Watson Research Center), snakket om data management, og problemstillingen hvordan bringe dataene fra databasen og fram til forståelige presentasjoner. Målet hans ligger i det han kaller regelbasert visualisering, hvor det skal være mulig for en bruker å spesifisere hva han ønsker å se på et svært høyt nivå, slik at metadata kan brukes i et regelsystem til å lage den presentasjonen som passer. De har begynt utviklingen av et system de kaller PRAVDA (Perceptual Rule-based Architecture for Visualizing Data Accurately og han innrømmet greit at dette var et tilfelle av å finne akronymet først og deretter finne på ord som passet). Her har de implementert bl.a. en fargeskalavelger for presentasjon av fringeplots, som baserer seg på å velge skalaer ut fra distribusjonen i dataene slik at presentasjonen gir rett forståelse av dataene

Mike Botts (University of Alabama, Huntsville) snakket om utviklingen av visualiseringsverktøy, særlig for store, multi-kilde datasett. Ghosts of visualization past henger fremdeles med oss mener han, og ber om at vi må komme nærmere virkeligheten i presentasjonene. Dvs. at man f. eks. presenterer en tidsanimasjon i real-time, slik at hendelser ikke barekommer riktig i sekvens og samtidighet, men at tiden faktisk også kan kreves å gå med en fastsatt hastighet. Han stresset også viktigheten av å separere display-modellen fra data-modellen, og ett av våre egne hjertebarn, nemlig det å kunne opprettholde interaktivitet, selv når datamengdene blir store

SIGGRAPH Exhibition

På utstillingsdagene brukte vi naturlig nok all den tid vi kunne i den heller overveldende utstillingssalen. Vi deltok i første omgang på de strømlinjeformede presentasjonene til de mest iøynefallende utstillerne. Intel, Sun og SGI utmerket seg spesielt, med store boots og proffe show.

Intel profilerte seg som den nye store workstation leverandøren med sin nye Xeon chip. Sun kjørte hardt på at de var (det seriøse, og eneste) alternativet til NT-baserte løsninger (Would you have the air traffic control system running on NT? I think not...), og var spesielt interessert i å kjøre fram Java3D.

SGI knallet til i kjent stil med Onyx2 show, hvor vi fikk en 10-minutters 3D-reise med realtime grafikk. Det går framover på den fronten også. Dessuten hadde de en stasjon med OpenGL Optimizer demo, hvor vi fikk se noen forskjellige store bilmodeller (1M polys) som var helt greie å manipulere. De hadde også en Onyx2 som kjørte en svær database med kart og satelittbilder som gjorde at en kunne zoome inn på (ideelt sett) et hvilket som helst sted i verden ned til det detaljnivået som fantes i basen. Siden de ikke hadde med seg mer enn 300

GB disk, var det dessverre bare USA som var tilgjengelig i detalj.

Microsoft presenterte sitt nye addon til Win98, Chromeffects, som er et system for å lage interaktiv grafikk o.l. på websider. Det kan virke som et forsøk på å haie markedsandeler på den delen fra Java3D, så får vi se senere hva de kommer med på den mer tekniske siden av distribuert/open 3D-grafikk arkitektur.

Ellers gikk vi rundt og samlet litt info om grafikkort, og fikk med oss en del brosjyrer. 3Dlabs hadde brosjyrer på Permedia3, uten at det virket som om det var klart for release enda. Matrox hadde heller ingenting av G200 å vise fram annet enn brosjyrer. HP hadde bare en søyle med 3 maskiner rundt hvor det sto og spant noen modeller på FX-kortene. Litt skuffa over liten profilering på rå 3D-ytelse. Vi fikk til slutt en OK prat med IBM om deres nye satsing på OpenGL-grafikk på RS/6000, og en fyr der skulle sende oss litt informasjon om deres representant i Sverige. Han mente det måtte være mulig å få til en deal om å teste en maskin.

SIGGRAPH Papers: Surfaces

Non-Uniform Recursive Subdivision Surfaces (Thomas W. Sederberg et.al.) En spesiell type subdivision surfaces ble demonstrert og kontinuitetsegenskaper for disse ble diskutert. Oppdelingen ble gjort på en litt annen måte enn den tradisjonelle todelingen, i det hjørnene i hver iterasjon ble delt i to nye kanter. I stedet for å lagre og oppgi flatene via de tradisjonelle knots, lagret de istedet parameter-verdiene for hver kant i utgangsflaten.

Fast Evaluation of Catmull-Clark Subdivision Surfaces at Arbitrary Parameter Values (Jos Stam) Det ble bevist at slike flater faktisk kan betraktes som en generalisering av NURBS, hvor Mr. Stam presenterte en 18-dimensjonal basis for dette parameterrommet i tilfellet med 5-valens noder. Hele teknikken generaliseres lett til vilkårlig valens. Mr. Stam er nok en velutdannet (les matematisk utdannet) mann, siden han lett satte på plass naive datamennesker som lurte på slike trivialiteter som hvordan han kunne være sikker på at hans 18 basis-vektorer var en komplett mengde. Meget imponerende framførelse.

A New Voronoi-Based Surface Reconstruction Algorithm. (Nina Amenta et.al. (University of Texas) Amenta presenterte deres algoritme for å konstruere en flate fra et sett med punkter. Det er egentlig ikke noe nytt å benytte Voronoi-diagrammer til dette, men poenget her var at de utnyttet heuristisk kjennskap til Voronoi-diagrammets geometriske egenskaper, som gjør algoritmen mindre kompleks enn tidligere kjente metoder. Typisk tidsforbruk for å rekonstruere en flate fra 30000 punkter var ca. 15 minutter., så dette er ikke så interessant i real-time grafikk.

Visual Basic og GPGS

Morten Ladstein, Millennium Software

Dette temaet ble tatt opp på GPGS seminaret høsten 1997. Det var stor interesse for å få dokumentert integrasjon av GPGS i Visual Basic. Forfatteren jobber ved Millennium Software AS i Trondheim.

GPGS-rutinene eksisterer i dag i form av et kompilert bibliotek som siden lenkes inn i hovedprogrammets kildekode. Da Visual Basic ikke støtter denne form for lenking, er en direkte oppkopling ikke mulig. Hadde imidlertid GPGS-rutinene eksistert i form av en selvstendig Active-X komponent hadde en direkte kopling ikke bare vært mulig, men en veldig enkel affære. Siden en direkte angrepsvinkel ikke er mulig, må man løse problemet indirekte, noe som betyr utvikling ved hjelp av flere programmeringsspråk. Ved å benytte et ytterligere programmeringsspråk som støtter den nevnte form for lenking, samt støtter kommunikasjon med andre programmeringsspråk, kan man lage små rutiner som kaller de ønskede GPGS rutinene. I dette tilfelle benyttes Digital FORTRAN som middel for kommunikasjon med GPGS-rutinene.

VISUAL BASIC → FORTRAN. For at Visual Basic skal ha mulighet til å kommunisere med et FORTRAN miljø, må koden gjøres tilgjengelig på FORTRAN-siden. Dette blir gjort ved en enkel setning for hver rutine man ønsker offentliggjort.

```
!DEC$ ATTRIBUTES DLLEXPORT::VB_GPGS_INITIALIZE

Subroutine VB_GPGS_INITIALIZE(Inst, MainhWnd)
Integer  MainhWnd, Inst
....
Return
End
```

Denne setningen er kun gyldig i Digital Fortran og fungerer derfor ikke i f.eks. Watcom Fortran. Denne setningen er imidlertid ikke nok. Visual Basic har ikke mulighet til å kalle eksterne rutiner med mindre de er deklarert for bruk, dvs. man må fortelle VB at man har en ekstern rutine. Dette blir gjort som følger:

```
Syntaks 1:
[Public | Private] Declare Sub name
   Lib "libname" [Alias "aliasname"]
   [([arglist])]

Syntaks 2:
[Public | Private] Declare Function
   name Lib "libname" [Alias "aliasname"]
   [([arglist])] [As type]

Eksempel:
Private Declare Sub GPGS_Initialize
   Lib "ShipShap.dll"
   Alias "VB_GPGS_INITIALIZE" _
   (Instance as Long, ParenthWnd as Long)
```

Etter denne deklarasjonen kan rutinen brukes som om den faktisk var skrevet i Visual Basic. Det er imidlertid en stor forskjell mellom eksterne og interne rutiner. Interne rutiner er "type safe", dvs. Visual Basic merker ved kompilering om argumentlisten er korrekt implementert eller ikke. Eksterne rutiner regnes derimot som ikke "type safe", med mindre et eksternt type bibliotek er implementert. Siden da ikke Visual Basic har mulighet til å verifisere de eksterne rutinene, er det veldig viktig at rutinene er deklarert hundre prosent korrekt.

Hvis du har en ugyldig deklarasjon vil man på et eller annet tidspunkt overskrive et ugyldig minneområde, noe som medfører en "windows exception error". Dette vil, under Windows NT medføre havari av Visual Basic, mens man under Windows 95 sannsynligvis står overfor et totalhavari. Dette er en feil som kan være vanskelig å oppdage, da den er avhengig av ingangsparametrenes binære størrelse. Man kan derfor oppleve havari i noen tilfeller, mens noen ikke. Dette kan enklest forklares ved hjelp av et eksempel. VB_GPGS_INITIALIZE rutinen har to inngangsparametre, begge av typen Integer*4. Dette betyr at man har en totalmengde inn på 8 bytes. En "windows exception error" inntreffer hvis de faktiske inngangsparametrene overstiger 8 bytes. Sender man inn f.eks. en integer*4 og en real*8 får man en feil. Sender man inn en integer*4 og en integer*2 vil man ikke oppleve noen form for havari, men siden man mister verdien i øvre del av andre parameter er det ikke sikkert rutinen fungerer korrekt. Dette er en form for feil som kan være veldig vanskelig å oppdage.

Bruk av GPGS. Initialisering av GPGS-hovedvindu For å bruke GPGS-biblioteket må man initialisere GPGS for bruk. Hvis man bruker flere uavhengige GPGS vinduer er det praktisk å gjøre dette i oppstarten av hovedprogrammet. For å avslutte GPGS på en korrekt måte trenger man også en avslutningsrutine, denne bør kalles ved avslutning av hovedprogrammet. Hovedtrekket i initialiseringsrutinen er som følger:

```
Subroutine VB_GPGS_INITIALIZE(Inst, MainhWnd)
....
  Id    = 72
  Ilti   = 17
  Call GPGS
  xiopt(1)    = 1         ! Invisible window
  xiopt(14)  = MainhWnd   ! Windows handle
  xiopt(15)  = Inst       ! Program Instance
  Call DevOpt( xiopt, ilti, 0, 0, Carr, 0)
  Call NitDev(Id)
  Call DatDev (0, 0, DevDat, 4)
....
Return
End
```

Som man kan se er man kun avhengige av to inngangsparametre; instansen på hovedprogrammet og dets vindu-"handle". Videre er det viktig å legge merke til at det initialiseringsrutinen egentlig gjør er å skape et usynlig GPGS hovedvindu, et vindu som senere ikke skal brukes, men er nødvendig for å kunne skape andre "child"-vinduer. For å avslutte GPGS trenger man kun følgende rutine:

```
Subroutine VB_GPGS_TERMINATE()
....
  Id = 72
  Call RlsDev(Id)
Return
End
```

Initialisering av GPGS-"child"-vindu I likhet med hovedvinduet må også "child"-vindu initialiseres for å kunne benyttes.

```
Integer Function VB_GPGS_CHILD_START (hWnd, ID)
....
xiopt(3)=0      !Left Border
xiopt(4)=130    !Right Border
xiopt(5)=0      !Bottom Border
xiopt(6)=100    !Top Border
xiopt(7)=0      !Size in Percentage

Call DevOpt(xiopt,ilti,0,0,Carr,0)
Call NitDwi(ID,3, hWnd)
Call VisDwi(ID,1)

Select Case (ID)
... Kall ønskede oppstartrutiner
... for forskjellig GPGS-vindu
End Select
Return
End
```

Man er også her avhengig av to inngangsparametre, hWnd, som er "child"-vinduets "handle", samt et eksklusivt ID-nummer. ID-nummeret blir videre diskutert i kapittelet som omhandler brukerinteraksjon med GPGS-vinduer. Grunnen til at NitDwi parameter nr 2 er satt til 3 og ikke 4 var komplikasjoner ved GPGS-vinduets oppførsel overfor andre vinduer. Kort oppsummert hadde GPGS-vinduet problemer med sin "z-order" posisjon. For å terminere et "child"-vindu kalles følgende rutine:

```
Subroutine VB_GPGS_CHILD_TERMINATE(ID)
....
        Call RlsDwi(lD)
Return
End
```

ID er samme eksklusive nummer som er benytter i initialiseringsrutinen. Dette er alt som trengs av kode for å vise et GPGS-vindu i et Visual Basic-miljø. Ønsker man brukerinteraksjon med GPGS-vinduet må man imidlertid være forberedt på litt mer "hard core" programmering.

Brukerinteraksjon med GPGS. Windows operativ system er basert på beskjeder ("messages"), og ved å avskjære disse beskjedene har man mulighet til å endre et vindus standard oppførsel. Dette er kalt "subclassing" og er gjort mulig fra og med Visual Basic 5.0 i form av AddressOf operatoren. For å avskjære den normale beskjedstrømmen må man erstatte den originale windows rutinen med en egen windows funksjon. Dette er en teknikk som krever inngående kunnskap om windowsprogrammering, der den minste feil vil forårsake totalhavari. Dette fordi man ved å benytte AddressOf operatoren setter til side alle Visual Basic sine innebygde sikkerhetsrutiner, som er lagt inn nettopp for å skjerme utvikler fra å begå kritiske lav-nivå feil. Et lite utdrag fra Visual Basic sin brukermanual forklarer dette på en grei og konsis måte:

Warning Using AddressOf may cause unpredictable results if you don't completely understand the concept of function callbacks. You must understand how the Basic portion of the callback works, and also the code of the DLL into which you are passing your function address. Debugging such interactions is difficult since the program runs in the same process as the development environment. In some cases, systematic debugging may not be possible.

Det er derfor å anbefale at en tredjeparts Active-X kontroll benyttes for å gjøre denne prosessen helt sikker og feilfri. Eksempel på en slik kontroll er Millennium Hook32 Active-X kontroll som er hundre prosent feilfri og tilgjengelig fra Millennium Software AS.

Nå som begrepet "subclassing" er forklart, er spørsmålet hvordan man kan benytte denne teknikken for å manipulere GPGS-vinduene slik vi ønsker. Siden kommunikasjon i Windows operativsystem baserer seg på beskjeder, og alle beskjeder går via et vindus "handle", synes dette som stedet å starte. Man trenger derfor GPGS-vinduets "handle". Siden dette ikke er den samme "handle" som blir sendt inn

i VB_GPGS_CHILD_START rutinen, blir man nødt å finne den på en annen måte. Det er her ID-nummeret i samme rutine kommer inn i bildet. For å kunne operere med mange GPGS-vinduer i samme applikasjon er man absolutt avhengig av at dette ID-nummeret er unikt for hvert GPGS-vindu.

Når man skaper et GPGS-"child"-vindu skaper man et vindu av klassetype `gpgs_windows`. Dette vindus overskrift ("caption") blir da "GPGS-F XX", der XX er vinduets ID-nummer. Det er denne overskriften som gjør det mulig å identifisere de forskjellige GPGS-vindu. Prøver man å benytte samme ID-nummer for ulike GPGS-vindu som er tilstede på samme tidspunkt, vil man følgelig aldri ha kontroll over hvilket av dem man arbeider opp mot, samt at man etter all sannsynlighet vil få interne GPGS feilmeldinger. Ved å deklarere tre eksterne rutiner som befinner seg i Windows egen user32.dll kan man ved hjelp av et vindus overskrift og dets klassenavn hente ut dets "handle" gjennom en iterasjonsprosess.

```
Private Declare Function GetWindow
   Lib "user32" _
   (ByVal hwnd As Long, ByVal wCmd As Long)
   As Long

Private Declare Function GetWindowText
   Lib "user32" Alias "GetWindowTextA" _
   (ByVal hwnd As Long, ByVal lpString
   As String, _ ByVal cch As Long) As Long

Private Declare Function GetClassName
   Lib "user32" Alias "GetClassNameA"  _
   (ByVal hwnd As Long,
   ByVal lpClassName As String,  _
   ByVal nMaxCount As Long) As Long
```

Funksjonen GetWindow kan, avhengig av parameteren wCmd, returnere sine barns ("children") vindu "handles". Ved å iterere seg gjennom disse, samtidig som man sjekker deres overskrift ("caption") og klassenavn, kan man hente ut den korrekte GPGS-vindu-"handle". GPGS-vinduets rettmessige far ("parent") har man allerede, det er første parameter i VB_GPGS_CHILD_START rutinen.

Når man har den ønskede "handle" kan det virkelige arbeid begynne. Millennium Hook32 Active-X kontroll har følgende interface:

```
Private Declare Function GetWindow
  Lib "user32" _
  (ByVal hwnd As Long, ByVal wCmd As Long)
  As Long

Private Declare Function GetWindowText
  Lib "user32" Alias "GetWindowTextA" _
  (ByVal hwnd As Long, ByVal lpString
  As String, _ ByVal cch As Long) As Long

Private Declare Function GetClassName
  Lib "user32" Alias "GetClassNameA"  _
  (ByVal hwnd As Long, ByVal lpClassName
  As String, _ ByVal nMaxCount As Long) As Long
```

Alt man altså trenger å gjøre er å sende inn vindus "handle" på det vindu man ønsker å avskjære beskjedstrømmen på, samt den beskjed man ønsker å avskjære. Da vil følgende rutine overta rollen til den ordinære windows rutinen for de ønskede beskjeder.

```
Private Sub CtlHook_WndProc
   (Msg As M_CtlHook32.WM, wParam As Long,
   _ lParam As Long, Result As Long)
....
   Select Case Msg
      Case WM_MOUSEMOVE:
           Utfør ønsket operasjon
      Case WM_ SIZE:
           Utfør ønsket operasjon
      End Select
....
End Sub
```

Ved hjelp av CtlHook_WndProc rutinen kan dermed utvikler manipulere brukers musbevegelser, vinduets størrelse o.l. Dette er nyttig hvis man f.eks. ønsker zooming, tilbakemelding om musens posisjon, ikke ønsker at vinduets størrelse skal overskride en viss størrelse etc.

GPGS og visual basics IDE-Miljø. Utviklerne av Visual Basic har lagt stor vekt på at språket skal være så trygt å bruke som overhodet mulig. Dette betyr at alle ressurser som blir knyttet opp ved hjelp av standard Visual Basic rutiner, vil automatisk bli frigjort på en korrekt måte. Når man jobber med GPGS i et Visual Basic IDE-miljø er man imidlertid avhengig av at rutinene i GPGS-biblioteket selv frigjør alle ressurser de måtte binde opp. I dette tilfelle betyr det at når man initialiserer et GPGS-vindu, må man også sørge for at det blir terminert på en ordentlig måte. Dette betyr at man til enhver tid må la applikasjonen terminere på en korrekt måte, også under utviklingen. Stopp knappen som finnes i Visual Basic miljøet må derfor aldri benyttes hvis GPGS er aktiv. Det vil nemlig forårsake at interne "callback" rutiner i GPGS-biblioteket som kommuniserer ved hjelp av GPGS-vindu-"handelen" ikke får beskjed om at vinduet er terminert, og følgelig vil prøve å sende beskjeder til et ugyldig minneområde. Totalhavari er ikke til å unngå.

Eventuelle spørsmål omkring innholdet rettes til morten@millennium.no

An Approach to Visualizing Transparency in Computer-Generated Line Drawings

Jörg Hamel, Stefan Schlechtweg, Thomas Strothotte
Otto-von-Guericke Universität, Magdeburg

Forfatterne jobber innen grafiske abstrakjsonsteknikker. Målet er å lage grafiske presentasjoner som retter oppmerksomheten mot bestemte detaljer i en illustrasjon. Kontaktadressen er hhv. hamel, stefans, tstr@isg.cs.uni-magdeburg.de

This paper builds on principles of depicting transparency in hand-made line drawings, and develops a method to generate similar, but computer-generated, line drawings in a two-step process. In the first step a brightness image is generated which is used as a basis for the line drawing created in the second step. Three different methods derived from traditional drawing techniques are shown.

As opposed to photographs and photorealistically rendered pictures, illustrations have the advantage that they can be more selective about their content and detail. Important details can be emphasized while extraneous can be omitted (see Dooley, Cohen [2]). Shapes can be simplified and hidden structures revealed. It has been shown that line drawings can be used to adjust finely the message conveyed by an image [14, 11, 15]. On the other hand, illustrations included in text books often use no colors, or at least very few colors to keep down the cost of the print. Therefore, line drawings have held their place in scientific illustration, despite of the advances of photography and computer graphics in the last century.

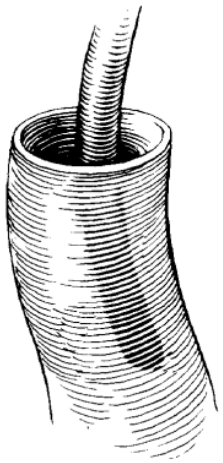


Figure 1a.

The generation of illustrations on the computer has attracted attention in the research community in recent years. In particular, Saito and Takahashi [6] proposed a method for enhancing rendered images by computing hatching lines based on G-buffers containing information of u-v coordinates of the parametric surface and lines along edges. Computer generated copper plates were the result of a raytracing system by Leister [4]. These copper plates can incorporate specular reflection and transparency. However, no special attention was payed to transparency, it was merely something inherited from the raytracing basis of the system. Winkenbach and Salesin [7] presented a rendering system for pen-and-ink illustrations based on parametric surfaces. Classical rendering techniques are applied to the lines, so shadows, texture, environment and bump mapping result in richer images. Schofield developed *Piranesi*, a rendering system for 'non-photorealistic' images, which handles like a paint program [10]. It works on a rendered picture augmented by G-Buffers containing material and depth information. The sketch-renderer described by Strothotte et.al. [13] provided interaction facilities to design lines and hatchings in a flexible way.

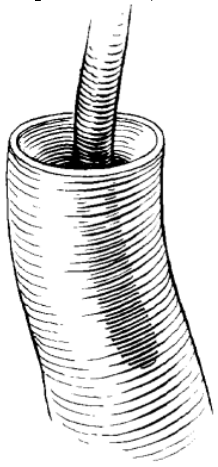


Figure 1b.

This paper treats a specific problem in computer-generated line drawings: how to visualize transparency. While transparency is well-established in illustrations based on photorealistic rendering (see e.g. Seligman, Feiner [12], who describe the Intent-Based-Illustration System IBIS) it has hardly been used in computer-generated line drawings so far—the exception being the raytracer by Leister. Probably, the rare visualization of transparent objects in manual line-drawings is responsible for this. Transparency, however, is essential to communicate the location of occluded objects.

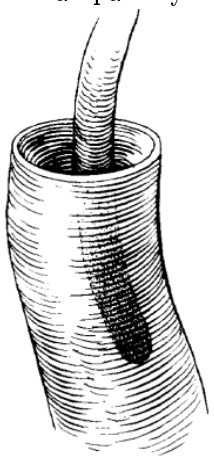


Figure 1c.

The paper is organized as follows. The section right after this introduction presents the reference line drawings and discusses the techniques which were used to create them. In the third section, two methods for rendering special pixel images are developed. These images are used in Section 4 to render the actual lines. Sec-

tion 5 describes the rendering system that was used as the basis for this article. The last section concludes the article and gives a preview to future work.

Drawing Techniques

This paper concentrates on line drawing techniques depicting transparency as presented in "The Guild Handbook of Scientific Illustration" [3]. In this book, techniques and methods are discussed which illustrators use to create their drawings. Characterizing for the techniques dealing with transparency is that they do not use colors. For colored illustrations, Hodges just recommends to lighten the color of the occluded object.

Examples which inspired our work are shown in Figure 1 a-c. The pictures show a stick-like object pointing downward into a semi-transparent tube, using increased thickness of lines (1a), using additional lines (1b), and using different line styles (1c).These three different ways for expressing transparency in a line drawing are drawn by hand by a scientific illustrator (taken from [3, page 101].
The illustrator used three different ways to show the 'shining through' stick on the surface of the tube. In the first picture, he drew extra thick lines. Notice that the lines at least on the front face of the tube are nowhere as thick as on the 'stick-part'. In the second picture, the stick is shown by additional lines interlacing with lines shading the surface of the tube. Comparing with the other two pictures it is obvious that the overall spacing of the lines is greater here. In the third picture, a different style altogether is used. Here, the part shining through is depicted with another shading technique: stippling. Characterizing for all three pictures is that the way the lower part of the 'stick' is drawn is independent from the upper part and that the lower part fades out the closer it comes to the opening of the 'tube'. The three drawings are appealing and intuitively understandable. For someone who wants to render similar pictures on a computer, however, it would be interesting to know if they are based on a common physical model, as that would keep things simple.

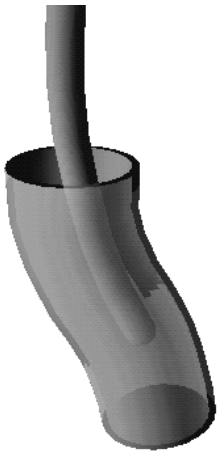


Figure 2a.

Rendering Techniques

To investigate this question—is a physical model used to generate these pictures?—the two objects were modeled using a common 3D-modeler and rendered with a simple, OpenGL-based renderer. The result can be seen in Figure 2a.
There is more information in this picture than we need. For example we do not need to see the backside of the tube and certainly not the thickness of the wall. Therefore, a simpler blending is applied as a postprocessing step as can be seen in Figure 2b.

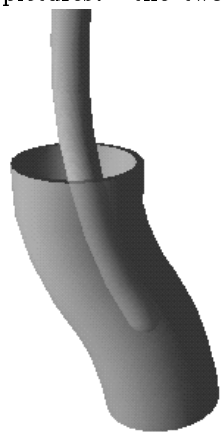


Figure 2b.

But still this picture is not suitable as basis for line drawings as the above examples. The lower part of the stick is visible for all its length in this picture, but not in the line drawings. Why is this so? Certainly, our picture is more realistic—even though we simplified the transparency calculation and used a quite simple lighting model—the stick does shine through all the way.

The answer is that line drawings seldomly follow physical models but adhere to certain empirical rules. Looking at the reference pictures, you can see for instance that the light source should be somewhere front-left taking into account the dark lines in the inner side of the tube, however, this does not explain why there is so much dark area left on the outside. This can only be explained by the assumption that the illustrator wanted to draw lines on the left as well so that the tube does not lose its spatiality. He also wanted to keep a certain minimal line length. For the transparency of the stick, we do not have to use assumptions as Hodges provides us with the rule:

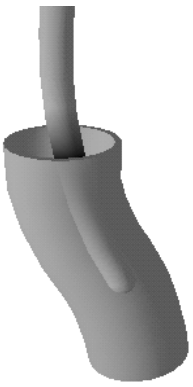


Figure 3.

- strengthen the shade where an object enters another object;

- set its intensity to zero at the edge and slowly increase the intensity again.

This is clearly a rather arbitrary way, and should therefore be modeled similarly. The first decision we have to make is, do we want to work in 2D—the image—or in 3D—the scene. Each way has positive and negative aspects, therefore, we will look at both in turn.

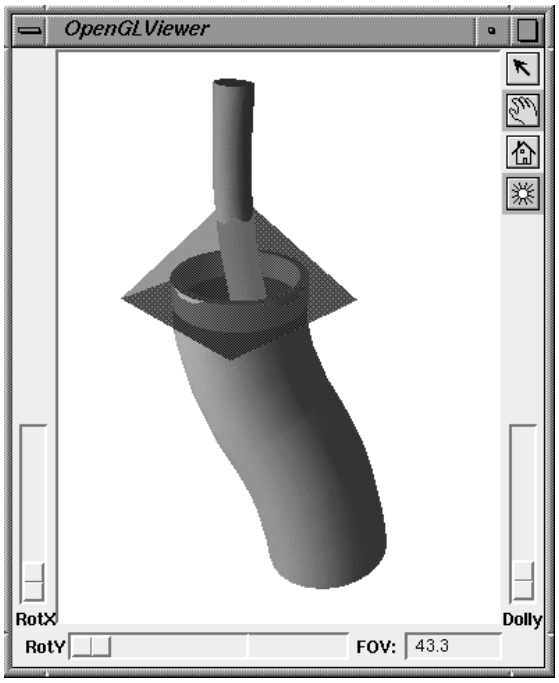


Figure 4a.

Working just on the brightness image is the most straightforward way: an edge detection in the objectID-buffer (another output of the OpenGLRenderer, see Section 5) between the outer and the inner part followed by a vectorization gives a number of lines. These can be used for all pixels of the occluded object—in this case the stick—to determine the smallest distance d_s. This distance is set in relation to the parameters d_{out} or d_{in}—depending on whether the pixel is in- or outside—which define the length of the respective fading areas.

Equation 1 shows the simple blending (a linear interpolation) of the brightness values from the picture without (b_i) and with (b_i^T) transparency applied to the pixels on the inside. In Equation 2, a linear scaling operation is sufficient for the outside part.

$$b_i' = \begin{cases} b_i \cdot \left(1 - \frac{d_s}{d_{in}}\right) + b_i^T \cdot \frac{d_s}{d_{in}} & \text{for } d_s < d_{in} \\ b_i^T & \text{otherwise} \end{cases} \quad (1)$$

$$b_i' = \begin{cases} b_i \cdot \frac{d_s}{d_{out}} & \text{for } d_s < d_{out} \\ b_i & \text{otherwise} \end{cases} \quad (2)$$

A look on the result in Figure 3 shows that the desired effect was achieved, however, the shape of the fading follows the edge between the inner and outer part. Thus in an example where this edge is formed differently, that is, where the curvature of the edge is more different to the curvature of the object, the result will look less satisfying.

The logical consequence from the latter thoughts is to define the fading analogously to the above 2D solution in 3D, thus implicitly taking perspective and curvature into account. For each area, a start point, a normal vector indicating the direction, and a length has to be defined.

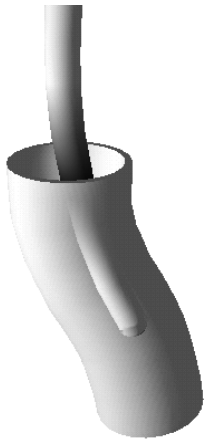


Figure 4b.

These parameters can be set using the OpenGLView with the help of a pyramid-formed widget (cf. Figure 4a). This works quite well as can be seen in Figure 4b.

Which of the two methods is to be preferred depends on the scene, certainly, in the scene used here, the two-dimensional method produced about the same result as the three-dimensional and since it requires less user interaction it might be preferred in this case.

For the final image, b_i^T in Equation 1 is set to zero so that the stick is just black and not showing any shading. Some arbitrary added shading supports the plasticity, and lessens the cold touch of the gouraud shading.

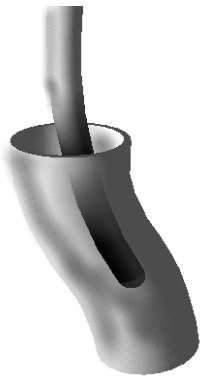


Figure 4c.

The resulting image shown in Figure 4c approximates the brightness distribution of our reference pictures quite well and gives us a working base for the creation of a line drawing.

Drawing Lines

To generate a line drawing that includes transparency in the way proposed by Hodges, we need

three different images:

- the 'opaque' image, where all objects are rendered opaque and the outer parts brightness is faded to zero towards the edge between inner and outer part;
- the 'transparent' image, as described in the previous section;
- a mask, which masks out all pixels not concealed in the opaque image (in this case—the lower part of the stick).

The methods that use these images are discussed in the following paragraphs.

Thick Lines

In the first reference image, the stick is shown by increasing the thickness of the lines. To achieve this effect, we intersect our lines with the edge of the mask and draw the part inside with a thickness based on the brightness of the transparent picture. The thickness of the line parts outside the masked area and all other lines is based on the opaque image.

More Lines

The second reference image shows the occluded object by interlacing the lines rendering the transparent object with additional lines. This time we draw only every second line on the tube, based on the opaque picture. The other lines we intersect with the mask edge and draw them based on the transparent image.

Different Style

The third reference image uses a different style altogether, here it is stippling. This case is quite similar to the last one. All we have to do is drawing the lines based on the opaque image, then add the lines—or dots, whatever our different style uses as primitives—drawn based on the transparent image and masked out.

Figure 5a-c The reader is encouraged to compare them with the reference pictures. The three different techniques—thicker, more lines, and different style—do deliver distinguishably different results.

A Rendering System

To understand why the process bringing transparency into line drawings developed in this article is actually two-stepped, it is useful to have a look at the rendering system our work is based on.

Architecture

The interface of the rendering system consists of four basic elements:

- In the OpenGL-View the scene as rendered with OpenGL is shown. Since OpenGL is hardwaresupported on a variety of platforms, rendering is quite fast, enabling interactive rotation and zooming. Attributes of objects that have a three-dimensional representation can be rotated, scaled and translated here as well.
- The Hierarchy-View is a view onto the structure of the scene and provides access to attributes of the objects.
- The Render-View is the front-end for the actual Hybrid Renderer. Results of the rendering process are shown here. A result, that is, a set of lines can be selected and removed from the image or dialogs for rendering and line style can be opened.

Functionality

There are currently five different renderers providing input to the hybrid renderer:

- An OpenGL-Renderer.
- A pixel-oriented renderer developed by Deussen [1], which intersects objects explicitly with planes of arbitrary orientation and performs edge detections on the resulting images. The results are copperplate-like lines, which can be used for hatching.
- A pixel-oriented renderer based on the one by Deussen, speed optimized for parallel planes only. Using the z-Buffer—an output from the OpenGL-Renderer—the space between the planes is filled with a color encoding the number of the next plane. Because perspective and curvature can get planes in the picture closer together or farther apart as intended, more than one of these pictures is necessary, four usually suffice though. Thus, edge detection has to be performed on only four pictures. The results are then interlaced.
- An analytic renderer working on polygonal meshes by Raab [5]. Edges between polygons with angles above a certain threshold are collected and returned. This results in lines describing the silhouette of the object.

- A simple stipple renderer using dithering of a grayscale pixel image.

The hybrid renderer draws the lines that are the output of the three line-renderers using the *linestyles* developed by Schumann and Schlechtweg [9]. These linestyles allow the encoding of surface related properties (as for example the brightness) in a line's attributes. So lines can change their thickness according to the brightness distribution on an object. The output of the OpenGL-Renderer can be used as a source for the values to be encoded. Other or additional sources are quite useful and yield a rich set of parameters to vary the appearance of the created line drawings, a few examples were shown in Figure 1, for further reference, see [9].
This article has shown methods to render line drawings containing transparency. The rendering process is divided into two steps: the creation of a brightness image with transparency; and the actual drawing of the lines. For each step, different methods were presented. The efforts were guided by analysis of the reference pictures and rules given by Hodges in [3].

An aspect not covered here so far is how to visualize different levels of transparency. Figure 6 demonstrates two levels of transparency. The windows in the figure 6b are not very transparent, the seats inside are only visible as shadows—like through a shower curtain. Notice that this is a logical result of the rendering process, when hatching is based on the lines of the transparent object and the transparency just results in a darkening of the transparent surface. Figure 6a was rendered with the methods presented in this paper, a 3D-fading in view direction delivers the intensity of the 'shadows'. In Figure 6b, in contrast, the seats are clearly visible, with contours and perspectively correct hatching drawn on the seats themselves. For this picture, the window was rendered separately and overlayed later, so that there is no transparency used except for this separation.

These two pictures can be envisaged to stand at opposite ends of the spectrum of transparency values. To investigate in between is one area of future work, another is the addition of colors. Last not least the resulting new ways for expressing transparency will have to be systemised and integrated into the rendering system and an appropriate user interface.

References

[1] O. Deussen, "Pixel-Oriented Rendering of Line Drawings" in: [15].

[2] D. Dooley and M. F. Cohen, "Automatic Illustration of 3D Geometric Models: Lines", *Computer Graphics 24*, Vol. 2, pp. 77–82, 1990.

[3] E. R. S. Hodges (Ed.), *The Guild Handbook of Scientific Illustration*, Van Nostrand Reinhold, New York, 1989.

[4] W. Leister, "Computer Generated Copper Plates", *Computer Graphics Forum*, Vol. 13, no. 1, pp. 69–77, 1994.

[5] A. Raab, *Techniken zur Exploration und Visualisierung geometrischer Modelle*, PhD Thesis at the Otto-von-Guericke University Magdeburg, Germany, 1998 (to appear).

[6] T. Saito and T. Takahashi, "Comprehensible Rendering of 3D Shapes", *Computer Graphics* (Proc. Siggraph), Vol. 24, No. 4, pp. 197–206, 1990.

[7] G. Winkenbach and D. H. Salesin, "Rendering Parametric Surfaces in Pen and Ink", *Computer Graphics* (Proc. Siggraph), Vol. 30, pp. 469–476, 1996.

[8] G. Winkenbach and D.H. Salesin, "Computer-Generated Pen-and-Ink Illustration", *Computer Graphics* (Proc. Siggraph), Vol. 28, No. 4, pp. 91–108, 1994.

[9] S. Schlechtweg, B. Schönwälder, L. Schumann, T. Strothotte, "Surfaces to Lines: Rendering Rich Line Drawings", *Conference Proceedings WSCG'98*, Vol. 2, pp. 354–361, 1998.

[10] J. Lansdown and S. Schofield, "Expressive Rendering: A Review of Nonphotorealistic Techniques" , *IEEE Computer Graphics and Applications*, pp. 29–37, May 1995.

[11] J. Schumann, T. Strothotte, A. Raab, S. Laser, "Assessing the Effect of Non-Photorealistic Rendered Images in CAD", *Proceedings of CHI '96*, pp. 35–42, 1996.

[12] D. D. Seligmann and S. K. Feiner, "Automated generation of intent-based 3D-illustrations", *Computer Graphics* (Proc. Siggraph), Vol. 25, pp. 123-132, 1991.

[13] T. Strothotte, B. Preim, A. Raab, J. Schumann, D. R. Forsey, "How to Render Frames and Influence People," *Computer Graphics Forum*, Proc. of Eurographics, Vol. 13, pp. 455–466, 1994.

[14] C. Strothotte, T. Strothotte, *Seeing Between the Pixels: Pictures in Interactive Systems,* Springer-Verlag, Berlin, 1997.

[15] T. Strothotte (ed.), *Abstraction in Interactive Computer Visualization: Exploring Complex Information Spaces,* Springer-Verlag, Berlin-Heidelberg-New York, 1998 (in print).

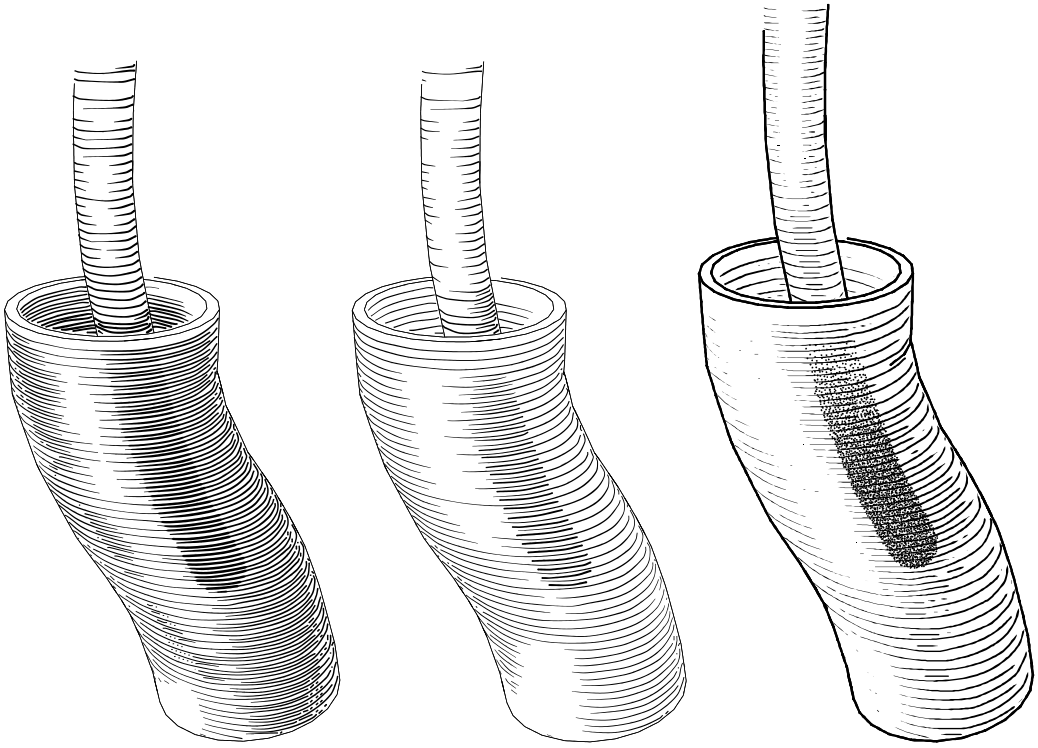


Figure 5 (a,b,c).

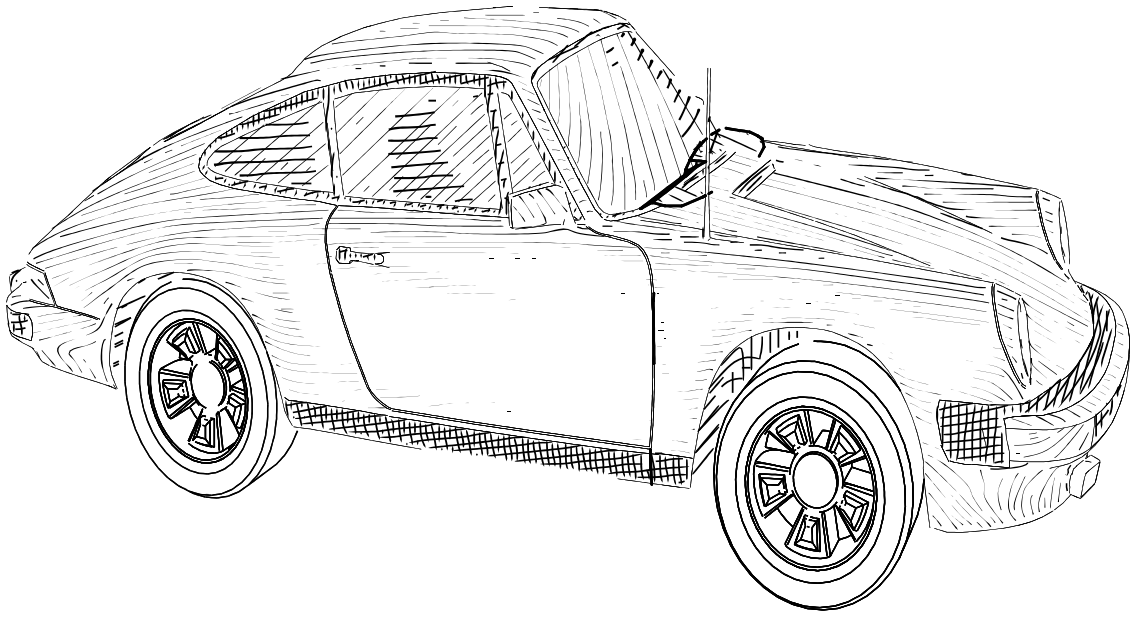


Figure 6a. (For Figure 6b see the cover page of this journal.)

GPGS hjørnet

Reidar Rekdal, DNV Software

Linux versjon av GPGS: En *privat* Linux versjon er i daglig bruk hos en GPGS bruker. Magnar Granhaug har vært involvert, men det er ikke utført noen fullskala test ennå. Vil du vite mere, kan du kontakte Magnar Granhaug, eller styrets sekretær.

CGM import i MS Word: Vi har tidligere beskrevet hvordan man kan importere CGM filer generert av GPGS i Word. I Word 97 på Windows NT kreves det imidlertid at det benyttes Administrator privilegier når man installerer CGM filteret, eller at man gjør endringer i registeret etter at filteret er installert. Her er et utdrag av artikkel Q172921 i Microsofts kunnskapsdatabase, som bekriver hvordan man endrer registeret hvis man ikke hadde Administrator rettigheter ved installasjonen:

1. Log into Windows NT as Administrator.
2. Click Start and then click run.
3. In the Open box, type "regedt32" (without the quotation marks), and then click OK.
   This will start the 32-bit version of the Registry Editor. If you run Regedit.exe instead of Regedt32.exe, you cannot change the registry security permissions.
4. In the Registry Editor, click the window titled HKEY_LOCAL_MACHINE.
5. Select the following registry key:

   ```
   HKEY_LOCAL_MACHINE\Software\
   Microsoft\Shared Tools\Graphic\
   Filters\Import\CGM
   ```

6. On the Security menu, click Permissions.
7. In the Registry Key Permissions dialog box, click the Replace Permission on Existing Subkeyscheck box.
8. In the Name list, double-click Everyone.
9. In the Special Access dialog box, click Full Control (All).
   This will change the permissions from Special Access to Full Control for the key and all its subkeys.
10. Click OK twice.
11. Close the Registry Editor.

Etter at dette er gjennomført, vil alle som logger seg inn på den aktuelle maskinen kunne importere CGM filer.

HPGL import i MS Word: Vi har også tidligere beskrevet hvordan vi kan importere HPGL filer i Word. I Word 97 er ikke dette mulig, ettersom HPGL er ett av de formatene som Microsoft ikke lenger *supporterer*.

GPGS relaterte 98 prosjekter: Det er usikkert om prosjektene *Implementering av Z-buffer i bitmapdriveren* og *Beskrivelse og uttesting av GPGS grafikk i Windows applikasjonsvindu* kan startes opp iår. De som var påtenkt disse prosjektene, tviler på at de har tid inneværende år. Skulle det være noen av våre medlemsbedrifter som mener de har kunnskap og ressurser til å ta på seg ett av disse prosjektene, vennligst ta kontakt med NORSIGD's sekretær.

Prosjekter for neste år: Tiden fram til nyttår går fort. Har du ideer til prosjekter for neste år, bør du allerede nå begynne å tenke på dette. Og husk, prosjektene trenger ikke være GPGS relatert.

Grafikkhjørnet — Datagrafikk på PDA

Wolfgang Leister, Norsk Regnesentral

Denne gangen skal vi se nærmere på bruk av datagrafikk på PDAer. En PDA er et akronym for "Personal Digital Assistant" og betegner en liten håndholdt datamaskin.

Mobilitet er et slagord som får stadig større betydning. Dermed har vi fått et marked for stadig mer portable dataløsninger, bl.a. datamaskiner som man bærer med seg. En laptop datamaskin har de fleste egenskapene til en vanlig PC, bortsett fra at den er beregnet for transport og drift i et mobilt miljø. Programvaren er også den samme som på en stasjonær maskin.

I mange sammenhenger er en laptop for stor, for tung eller upraktisk. Både skjermen og tastatur bidrar vesentlig til de ytre målene til en laptop.

En PDA er vanligvis på størrelse med en håndflate og veier noen få hundre gram. Som skjerm finnes det mest svart-hvit LCD skjermer med en oppløsning på 320×240. Skjermen brukes også til å gi inn data med en penn (touch screen). Det finnes mange varianter av maskiner: noen har tastatur, noen er integrert med modem eller mobiltelefon, noen har infrarød-kommunikasjon. Iternminne er på noen få MByte, mens det ikke finnes eksterne lagringsmedia.

De forskjellige leverandørene har gjort ulike valg når det gjelder systemprogramvare: noen bruker operativsystemet Windows CE, mens andre har egne løsninger (f.eks. PalmOS, Psion). Dette bestemmer også utvikling av programvare for PDA, inkludert hvilket API som kan benyttes for grafiske anvendelser.

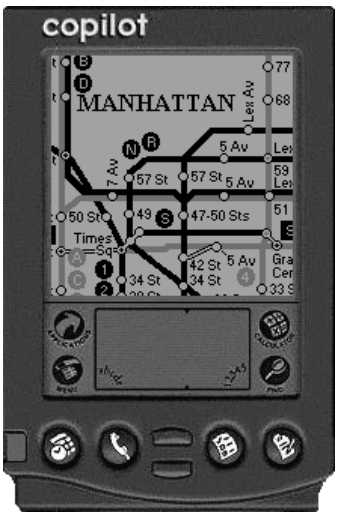
De fleste av PDAene har ikke tastatur. For å gi inn tekst skrives bokstaver med en spesialpenn på en berøringssensitiv skjerm. På noen av modellene må man lære seg et eget alfabet, mens andre har en modul som gjenkjenner håndskrift.

Grafikkmulighetene er begrenset med skjermstørrelse, svart-hvit skjerm og manglende grafikkprosessor. Dette gir utfordringer for programvareutviklere, som må tilrettelegge grafisk materiale som skal presenteres. Effektive metoder for fargereduksjon, dithering, og grafisk abstraksjon hører altså ikke en svunnen tid til, men er i høyeste grad aktuell!

Programmering skjer ofte off-line på en vanlig PC med egne utviklingspakker og biblioteker (cross-kompilering). For å teste ut programvaren brukes det simulatorer som kjøres på utviklingsmaskinen.

Web-browsing er mulig på en PDA når den tilkobles et nettverk, der grafisk materiale konverteres on-line. Delvis konverteres bilder før overføring på en spesiell proxy-server til et format som er egnet for rask fremvisning og som reduserer behovet for overføringskapasitet.

Noen av applikasjonene er basert på tekst, som terminkalender, epost og notatblokk. Spekteret av grafikk-applikasjoner rekker fra tegneprogrammer og grafiske notatblokker, presentasjon av tegninger, bilder og kart, navigasjon, oppslagsverk, innstruksjoner for supportavdelinger til spill og underholdning.



Et kartutsnitt vist på en PalmPilot.

På internett finnes det mye informasjon om temaet. Leverandørene av PDAer har sider og søkemotorer med programvare som kan hentes fra nettet for nærmest alle anvendelser. *http://www.handheldinterfaces.com* er informative Web-sider til en tilfeldig valgt forhandler. Informasjon om *Windows CE* finnes på Microsofts Web sider, mens *Palm Pilot* finnes under *http://www.palmpilot.com* og *http://www.pilotgear.com*. Ellers anbefales bruk av søkemotorer og indekssider for å finne mer informasjon.

Aktivitetskalender

Hva skjer når og hvor?

Oktober 1998	
11–13	**MICCAI 98** , (First International Conference on Medical Image Computing and Computer Assisted Interventions), Boston MA, USA. `http://www.ai.mit.edu/miccai98.html`.
12–14	**IEEE International Conference on Systems, Man and Cybernetics**, Rutgers University, USA. `mailto:jafari@gandalf.rutgers.edu`.
12–15	**MMM '98** The International Coinference on Multimedia Modeling, Lausanne, Sveits. `http://ligwww.epfl.ch/~thalmann/mmm98.html`.
November 1998	
1–4	**Eleventh Annual Symposium on User Interface Software and Technology**, San Francisco CA, USA. `mailto:mynatt@parc.xerox.com`.
2–5	**ACM Symposium on Virtual Reality Software and Technology 1998**, Taipei, Taiwan. `mailto:snyang@cs.nthu.edu.tw`.
8–12	**ICCAD** (IEEE International Conference on Computer Aided Design), Kyushu, Japan. `mailto:yasuura@c.cse.kyushu-u.ac.jp`.
Februar 1999	
8–12	**WSCG'99**, The Seventh International Conference in Central Europe on Computer Graphics and Visualization 99, Plzen, Tsjekkia. `http://wscg.zcu.cz`, klikk på WSCG'99.
23–26	**VRML99**, the Fourth International Conference on the Virtual Reality Modeling Language and Web 3D Technologies, Paderborn, Tyskland. `http://www.c-lab.de/vrml99/`
Mars 1999	
5–6	**Simulation und Visualisierung'99**, Magdeburg, Tyskland. `http://www.simvis.org/tagung99`.
August 1999	
8–13	**SIGGRAPH'99**, Los Angeles, USA. `http://www.siggraph.org/`.
September 1999	
7–11	**Eurographics'99**, Milano, Italia. `http://eg99.dsi.unimi.it/`.

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 datbehandling. NORSIGD har per januar 1998 44 institusjons- og 12 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*. Kontingenten er redusert til 150 kr ved samtidig medlemskap i vår europeiske samarbeidsorganisasjon *Eurographics*.

Alle medlemmer får tilsendt medlemsbladet *NORSIGD Info* 3–4 ganger per år.

Interesseområder

NORSIGD er et forum for alle som er opptatt av grafiske brukergrensesnitt og grafisk presentasjon, 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

GPGS er en 2D- og 3D grafisk subrutinepakke. GPGS er maskin- og utstyrsuavhengig. Det vil si at et program utviklet for et operativsystem 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 fra FORTRAN og C.

Det finnes versjoner av GPGS for en rekke forskjellige maskinplattformer, fra stormaskiner til Unix arbeidsstasjoner og PC. GPGS har drivere for over femti forskjellige typer utsyr (plottere, skjermer o.l.). GPGS støtter mange grafikkstandarder slik som Postscript, HPGL/2 og CGM. GPGS er fortsatt under utvikling og støtter stadig nye standarder.

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

Eurographics

NORSIGD samarbeider med Eurographics. Personlige medlemmer i NORSIGD får 20 SFr rabatt på medlemskap i Eurographics, og vi formidler informasjon om aktuelle aktiviteter og arrangementer som avholdes i Eurographics-regi. Tilsvarende får Eurographics medlemmer kr 100 i rabatt på medlemskap i NORSIGD.

Eurographics ble grunnlagt i 1981 og har medlemmer over hele verden. Organisasjonen utgir et av verdens fremste fagtidsskrifter innen grafisk databehandling, *Computer Graphics Forum*. *Forum* sendes medlemmene annen hver måned. Eurographics konferansen arrangeres årlig med seminarer, utstilling, kurs og arbeidgrupper.

NORSIGD
v/ Reidar Rekdal
Dnv Software
Postboks 300
1322 HØVIK

Returadresse:
NORSIGD v/ Reidar Rekdal
DNV Software
Postboks 300
1322 HØVIK

Styret i NORSIGD 1998

Funksjon	Adresse	Telefon	email
Leder	Ketil Aamnes ViewTech AS PB 47 Pirsenteret 7005 TRONDHEIM	73 54 61 23 (direkte) 73 54 61 44 (fax)	Ketil.Aamnes @viewtech.no
Fagansvarlig	Wolfgang Leister Norsk Regnesentral Postboks 114 Blindern 0314 OSLO	22 85 25 78 (direkte) 22 85 25 00 (sentralbord) 22 69 76 60 (fax)	leister@online.no
Sekretær	Reidar Rekdal Det Norske Veritas Software Postboks 300 1322 HØVIK	67 57 73 18 (direkte) 67 57 72 50 (sentralbord) 67 57 72 72 (fax)	reidar.rekdal @dnv.com
Styremedlem	Gisle Fiksdal MARINTEK A.S Postboks 4125, Valentinlyst 7002 TRONDHEIM	73 59 59 07 (direkte) 73 59 57 76 (fax)	Gisle.Fiksdal @marintek.sintef.no
Varamedlem	Svein Taksdal Norges Vassdrags- og Energiselskap Hydrologisk Avdeling, Seksjon data Postboks 5091, Majorstua 0301 OSLO	22 95 92 86 (direkte) 22 95 92 01 (fax)	svein.taksdal @nve.no
Varamedlem	Rune Torkildsen Autograph Broadcast Systems AS Postboks 2 5002 BERGEN	55 90 81 40 55 90 80 70 (sentralbord) 55 90 80 90 (fax)	Rune.Torkildsen @tv2.no

Svarkupong

- ◯ Innmelding – institusjonsmedlem
- ◯ Innmelding – personlig medlem
- ◯ Innmelding – Eurographics medlem
- ◯ Ny kontaktperson
- ◯ Adresseforandring

Navn: .....
Firma: .....
Gateadresse: .....
.....
Postadresse: .....
.....
Postnummer/sted: .....
.....
Telefon: .....
Telefaks: .....
email: .....