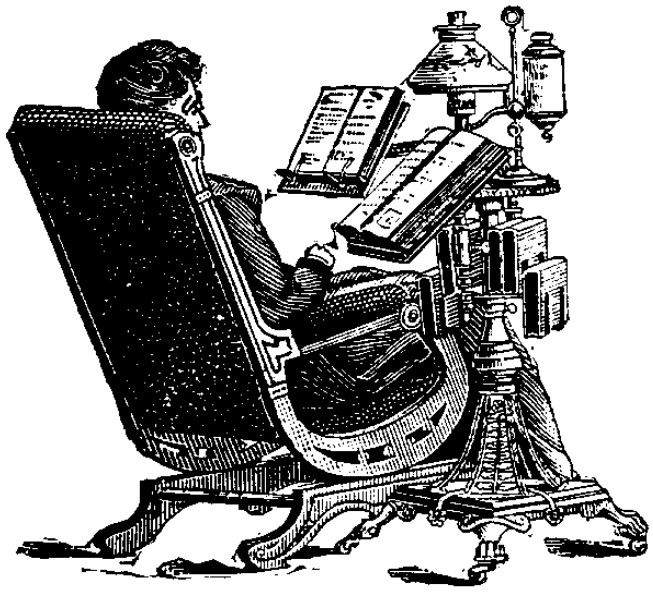
Grafikk på Web
— en oversikt

Norsk Regnesentral
ANVENDT DATAFORSKNING
Norwegian Computing Center/Applied Research and Development

NOTAT/NOTE



IMEDIA/01/99

Wolfgang Leister
Arve Larsen

Oslo
Januar 1999

ANVENDT DATAFORSKNING

Notat / Notat

Tittel/Title:
Grafikk på Web — en oversikt

Dato/Date: Januar
År/Year: 1999
Notat nr/:
Note no: IMEDIA/01/99

Forfatter/Author:
Wolfgang Leister, Arve Larsen

Sammendrag/Abstract:

Web teknologien ser ut til å bli stadig mer brukt som plattform for interaksjon med brukeren. Prosjektet undersøker mulighetene for datagrafikk og grafisk presentasjon på web. Billedformater, CGI-teknikker, Java og CORBA, VRML, plugin-teknologier, mm. beskrives i en oversikt. Samtidig undersøkes hvordan eldre grafikkapplikasjoner kan forsynes med et web-grensesnitt, også med henblikk på grafikkpakken GPGS.

Emneord/Keywords: Datagrafikk, WWW, Web, Client-Server, Java, Corba

Målgruppe/Target group:

Tilgjengelighet/Availability: Open

Prosjektdata/Project data: NORSIGD

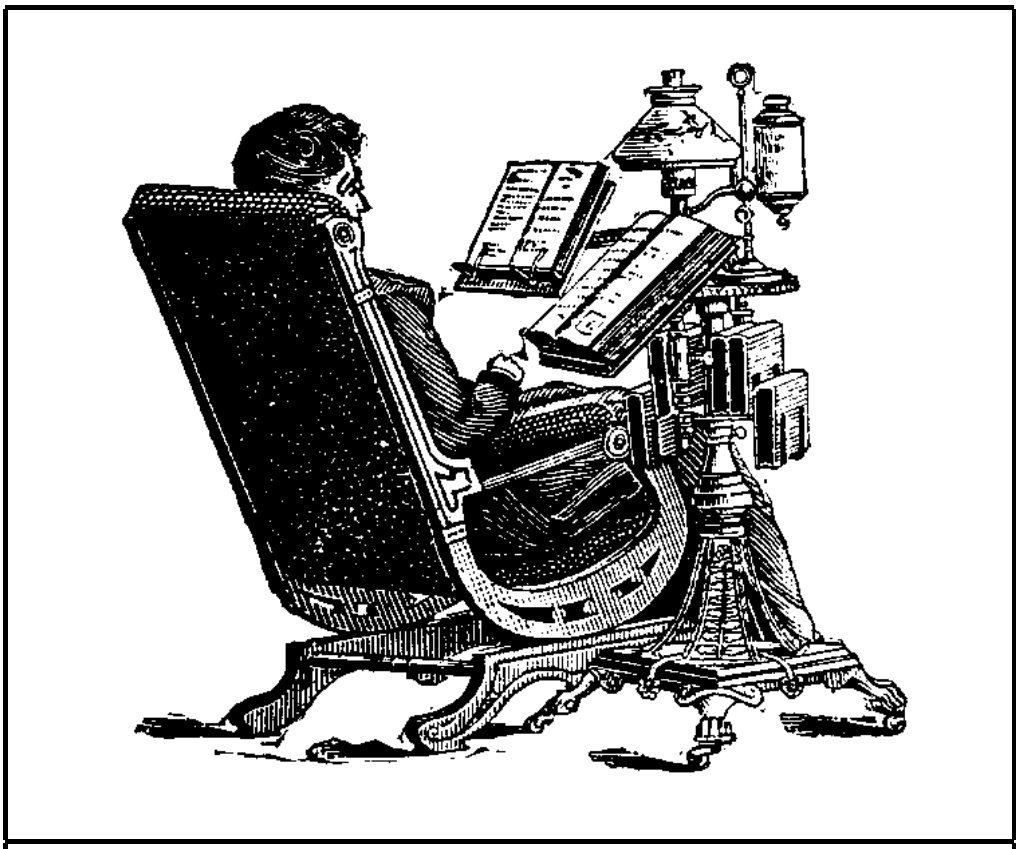
Prosjektnr/Project no: 618000

Antall sider/No of pages: 14

Norsk Regnesentral/Norwegian Computing Center
Gaustadalléen 23, Postboks 114, N-0314 Oslo, Norway
Telefon (+47) 22 85 25 00, telefax (+47) 22 69 76 60

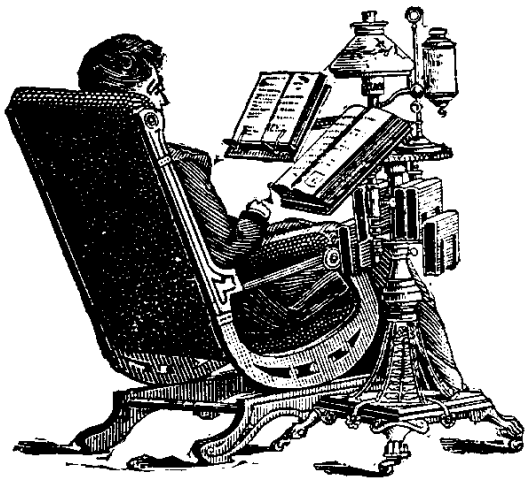
NORSIGD INFO

Nummer 1 1999



NORSK SAMARBEID INNEN GRAFISK DATABEHANDLING

ISSN 0803-8317



Om forsiden
Bildet på forsiden viser en teknisk innretning som kan betraktes som forgjengeren til en nettleser. Bildet av *Holloway book-rest and dictionary holder* stammer fra en reklame i år 1888.

Hilsen fra styret

Kjære medlemmer,

For 25-år siden var det stort behov for å lage et maskin- og skjermuavhengig grafikkbibliotek. GPGS har vært et av de første svarene på denne utfordringen. Tiden har ikke stått stille, og internett og WWW har igjen forandret de tekniske forutsetningene.

Mens Web-browseren utvikler seg stadig mer til et standardisert brukergrensesnitt, som alle brukere etterhvert blir vant til å bruke, finnes det ingen enhetlige løsninger for å få datagrafikk over på det nye mediet.

Vi skal se på datagrafikk-teknikker som brukes sammen med WWW. Ved siden av en introduksjon i WWW-teknikker tar vi grafikkbiblioteket GPGS som eksempel hvordan applikasjoner kan bygges opp for å gjøre dem egnet for bruk på WWW.

NORSIGDs årsforsamling har i 1998 vedtatt gjennomføring av et prosjekt i NORSIGDs regi, som tar for seg dette temaet. Denne spesialutgaven av NORSIGD Info er en prosjektrapport som skal komme alle NORSIGDs medlemmer til gode.

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:	1999: 10/2 20/5 20/9 20/12
Annonsepriser:	Helside kr 5 000 Halvside kr 2 500
Oversettelser:	Wolfgang Leister
Layout:	Wolfgang Leister LaTeX2ε

Innhold

Hilsen fra styret 3
Grafikk på Web – en oversikt 4
Hvorfor denne oversikten 4
Oppbygning av WWW 4
Grafikkformater 8
Teknikker i klienten 9
Tjenersiden 11
Integrasjon av WWW og grafikkbiblioteker . 12
Konklusjon 16

Grafikk på Web — en oversikt

Wolfgang Leister, Arve Larsen

Norsk Regnesentral
ANVENDT DATAFORSKNING

Web-teknologien ser ut til å bli stadig mer brukt som plattform for interaksjon med brukeren. Prosjektet undersøker mulighetene for datagrafikk og grafisk presentasjon på web. Billedformater, CGI-teknikker, Java og CORBA, VRML, plugin-teknologier, mm. beskrives i en oversikt. Samtidig undersøkes hvordan eldre grafikkapplikasjoner kan forsynes med et web-grensesnitt, også med henblikk på grafikkpakken GPGS.

1 Hvorfor denne oversikten?

Web-teknologi blir brukt til mer enn ren informasjonstilgang. Mange applikasjoner, også grafiske, aksesseres ved hjelp av Web-teknologi og ikke minst Web-grensesnitt. Web-grensesnittet utvikler seg stadig mer til et felles brukergrensesnitt. En skille mellom applikasjon og presentasjon er et kjennetegn ved bruk av Web-teknologi. Dermed kan ressursene være fordelt, dvs. plassert på geografisk forskjellige steder, mens brukeren kan aksessere alle viktige data fra arbeidsstasjonen sin.

Mange eksisterende applikasjoner er ikke tilpasset det nye mediet. Spesielt applikasjoner med mye grafikk krever ved en tilpasning mye omtanke iht. systemarkitektur, ressursbruk, og mulighetene som WWW tilbyr. Slike applikasjoner kan være beregnings-, simulerings- eller visualiseringsapplikasjoner, delvis basert på eldre grafikkbiblioteker, som kan ha vært i bruk i årevis uten store forandringer. Ved overgangen til et nytt medium er det hensiktsmessig å ikke endre alt radikalt, men å bruke så mye som mulig om igjen.

NORSIGD har utviklet det grafiske biblioteket GPGS siden midten av 70-tallet. Denne undersøkelsen skal også gi et svar på hvilke muligheter som finnes for å porte applikasjoner som bruker dette biblioteket til bruk på WWW. Undersøkelsen gjennomføres med sikte på å overføre erfaringene til å gjelde andre biblioteker og nyutvikling av applikasjoner.

Etter en gjennomgang av de vanlige teknikker, blir forskjellige metoder, som har en nær tilknytning til datagrafikk, belyst. Spesielt blir det lagt vekt på applikasjoner med et grafisk innhold. Vi har ikke som intensjon å lage en manual eller å dekke alle kjente muligheter, men å vise hvordan implementeringer kan gjennomføres.

Problemstillinger forbundet med multimedia eller digital TV blir ikke behandlet i denne rapporten, selv om også disse teknikkene har aspekter innen datagrafikk og WWW. Innen Multimedieteknikk står synkronisering og integrasjon av forskjellige datatyper i forgrunnen, mens digital TV i hovedsak beskjeftiger seg med streaming-teknikker, og innholdsleveranse i en broadcast-setting. Denne rapporten er heller ikke ment til å dekke Internett som sådan (f.eks. protokoller i transportlaget, QoS-teknikker eller nettverk).

2 Oppbygning av WWW

World Wide Web, WWW, eller W3 ble foreslått av Tim Berners-Lee under et forskningsopphold hos CERN. I høsten 1980 utviklet han et program "Enquire-Within-Upon-Everything", som gjorde det mulig å legge inn pekere i et dokument til tilfeldig valgte noder. ENQUIRE ble kjørt på Norsk Data maskiner under operativsystemet SINTRAN-III.

I 1989 foreslo Berners-Lee en verdensomspennende hypertekst-prosjekt, som etterhvert skulle bli til World Wide Web. WWW ble utviklet under forutsetningen av at mange personer skulle kunne jobbe sammen i felleskap ved å legge ut informasjon på et vev av hypertekstdokumenter. WWW dokumenter skulle legges på vertsmaskiner, mens en tjener-software, som ble kalt **browser** (nettleser), skulle gi tilgang til informasjonen som var lagt ut på denne serveren. Nettleseren skulle hente inn informasjon ved å søke etter URLen til en peker. Deretter bruker den HTTP-protokollen for å hente dokumentet som er kodet i HTML.

Aktørene på WWW har forskjellige roller på internett. Aktørene kan være personer, grupper eller organisasjoner:

Brukere bruker en nettleser for å dra nytte av innholdet (konsument).

Innholdsleverandører legger innhold i tjenester tilrette i form av tekst, grafikk, og annet innhold. Dette kan være universiteter, firmaer, forlag eller privatpersoner, som har en interesse av å spre et budskap.

Aksessleverandør (IAP, Internet Access Provider) leverer den tekniske tilgangen for brukeren for å kunne bruke tjenesten fra innholdsleverandør.

Tjenesteleverandør (ISP, Internet Service Provider) er ansvarlig for å tilby og vedlikeholde den tekniske siden av tjenesten for innholdsleverandøren (web hosting).

WWW er en tjeneste som opprinnelig ble utviklet for å gi brukerne tilgang til dokumenter og tjenester. Dermed er WWW en del av middleware. WWW gjør dokumenter tilgjengelig på internett og har utviklet seg til et globalt informasjonssystem. Med WWW kan det oppnås en standardisering av brukergrensesnittet. Sluttbrukerne kan bruke det samme grensesnittet for alle applikasjoner uansett firma-, industri- eller landegrenser.

WWW baseres på følgende teknologier:

- Web-tjener (Web server)
- Nettlesere (Web browsere)
- Uniform Resource Locator (URL)
- Hypertext Transfer Protocol (HTTP)
- Hypertext Markup Language (HTML)
- Web-navigasjon og søkemotorer
- Overganger (Gatewayer) for å aksessere ikke-Web-ressurser.

2.1 Nettleseren

En Web-browser eller nettleser er brukerens grensesnitt til WWW. Den skal være enkel i bruk. Ved hjelp av pull-down menuer, pop-up menuer, knapper og kommandoer gis det mulighet for å få tilgang til remote server, å lese dokumenter, å skrive ut resultater, laste ned kode, lagre dokumenter og navigere i WWW. Nettlesere bruker HTTP (HyperText Transfer protocol) for transfer av innholdet og tillater å hente informasjon til nettleseren. Det finnes mange forskjellige nettlesere, bl.a. Lynx, Mosaic, Netscape Navigator/Communicator, Internet Explorer, Arena, Amaya og Opera.

Vanligvis blir data bare overført ved oppfordring fra brukeren eller gjennom et skript i nettleseren. I motsetning til dette er **Push technology** en software-komponent som enten leverer informasjon direkt til en datamaskin uten at informasjonen bestilles på web eller søkes fram, eller som informerer når data er tilgjengelig eller har blitt oppdatert.

2.2 HTTP

HTTP (HyperText Transfer Protocol) er en protokoll på applikasjonsnivå som er utviklet for bruk på WWW. HTTP bygger på en enkel request-response (forespørsel-svar) modell. Det opprettes en forbindelse til en web-server og et dokument lastes ned til nettleseren. Dokumentet presenteres deretter for brukeren. Protokollen gir ikke adgang til å lagre statusinformasjon. Alle forespørsler blir gjennomført uavhengig fra den forrige.

Dokumentet som lastes ned kan også inneholde referanser til andre objekter som skal lastes inn. Hvert slikt objekt hentes ned over en egen HTTP-forbindelse.

Protokollen er utviklet for å gi brukeren muligheten for å arbeide i en hypermedie-kontekst. Modellen som brukes har fire steg:

Connection: Klient-applikasjonen (nettleseren) etablerer en forbindelse med serveren.

Request: Klienten sender en request-melding til serveren, bl.a. navnet til ressursen (URI), request-metoden, protokollversjon, request modifiers, client information og evtl. innhold. Motoder er bl.a. GET, HEAD, POST, PUT eller DELETE. Et eksempel på en request er:

```
GET norsigd.html HTTP/1.0
```

Response: Serveren svarer med å sende en header med resultatkode, MIME-type og dokumentet.

Close: Klienten eller Serveren lukker forbindelsen.

2.3 HTML

HTML (Hyper Text Markup Language) er et enkelt og plattformuavhengig språk for å beskrive og kode Web-dokumenter. HTML tillater bl.a. å legge hypertekst linker (pekere) inn i et dokument som er spesifisert som URL. Dokumenter kan også inkludere annet innhold i et dokument, som grafikk, multimedia, audio, video, etc.

HTML er basert på SGML (Standard Generalized Markup Language), som ble utviklet av IBM for å gjøre dokumenter tilgjengelig på tvers av plattformgrenser. HTML-dokumenter blir lagret i en vanlig tekstfil med koding av endel spesialtegn. HTML-markup-tags blir brukt for å beskrive formatteringen. Med dette kan overskrifter, kursiv- eller fet skrift, lister og tabeller beskrives, og grafikk, lyd og plassering av annet materiale beskrives. Mulighetene for å angi eksakt layout av en side eller bruk av grafiske elementer er ytterst begrenset. Ønskes det en eksakt beskrivelse av et dokument blir style-sheets eller plug-ins for spesielle formater (f.eks. Acrobat Reader) brukt.

Vi gir et kort eksempel på et HTML-kodet dokument:

```
<HEAD>
<TITLE>NORSIGD Hjemmeside</TITLE>
</HEAD>
<BODY bgcolor="#cfcfcf">
<H1>NORSIGD Hjemmeside</H1>
<P> Velkommen til NORSIGDs hjemmeside.
Her f&aring;r du vite mer om
<UL>
<LI><A "HREF=foren.html">Foreningen</A>
<LI><A "HREF=gpgs.html">GPGS</A>
</UL>
</P>
<IMG ALIGN=TOP ALT="[foto]"
     SRC=norsigd.gif>
</BODY>
```

HTML har inneholder også muligheter for å bygge opp strukturerte skjemaer (FORMS). Disse integreres i en webside slik at brukeren kan taste inn eller velge ulike verdier for de forskjellige feltene. Disse verdiene kan så sendes via HTTP til tjenersiden for behandling.

Dynamic HTML defineres av WWW-konsortiet (W3C) som en kombinasjon av HTML, style sheets, og skripts som tillater at dokumenter kan endres dynamisk. Dynamisk HTML gir web-forfattere muligheten til mer fleksibilitet og kontroll ved at spesielle WWW elementer og HTML objekter endres dynamisk. Dette gjøres gjennom et skript-språk som gir muligheten til å forandre atributter av grafikk og tekst uavhengig av hverandre.

2.4 Grafikk på WWW

Rastergrafikk og bilder blir vanligvis lagt inn som tag med en IMG-tag i html, f.eks.

```
<IMG ALIGN=TOP ALT="[foto]"
     SRC=bilde.gif>
```

Hvilke billedformater som kan brukes bestemmes av nettleseren. Vanligvis er formatene GIF og JPEG implementert, mens PNG-formatet kan bli en mye brukt standard i fremtiden. Formater som ikke er implementert i nettleseren kan implementeres i en egen browser-plugin. Mer om disse grafikkformater beskrives i avsnitt 3.

Noen av formatene (f.eks. GIF) tillater å kode animasjoner i en fil som en sekvens av enkeltbilder. Dermed er det også mulig å presentere korte forhåndsberegnete animasjoner. Ved bruk av "MIME-multipart"-taggen er det i en viss grad mulig å vise en mediestrøm som kan inneholde en animasjon, dog uten QoS-kontroll. (Med QoS kontroll menes muligheten å styre kvalitetsparametre for en strøm.)

Et **Image Map** er en definisjon av geometriske 2D-områder for et bilde. Disse områdene forbindes med en URL som blir aktivert når det klikkes på det tilsvarende området. Med dette som utgangspunkt utvikles det brukergrensesnitt hvor et image-map legges bak et bilde. Når et område aktiveres skjer behandlingen enten i klienten (f. eks. i JavaScript), eller på tjeneren (f.eks. ved hjelp av CGI-grensesnittet).

2.5 Eksterne applikasjoner og Browser-Plugin

Funksjonaliteten for en nettleser kan utvides på flere måter for å støtte datatyper for lyd, video, animasjon eller grafikk:

- bruk av klientbaserte språk så som JavaScript, Java, ActiveX
- eksterne applikasjoner (Helper Applications)
- Browser Plug-Ins

For å kunne starte eksterne applikasjoner for datatyper som nettleseren ikke håndterer, legges det inn en kobling mellom MIME type og navn og parametre til en ekstern applikasjon. Denne må være installert på forhånd på maskinen, og all kontroll foregår utenfor ansvaret til nettleseren. Det finnes dog muligheter for en løs kommunikasjon mellom applikasjon og nettleseren (f.eks. ved å starte applikasjonen eller nettleseren med spesielle parametre).

Plug-ins er applikasjonsmoduler som kjøres innenfor nettleseren. Selv om disse er en del av nettleseren iht. minneområde, tråder, presentasjon og andre ressurser, er plug-ins programmert som separate moduler som utvider funksjonaliteten til nettleseren, ved å støtte flere MIME

typer for avspilling av f.eks. lyd, video, animasjoner, grafikk, mm.

Plug-ins implementeres ofte som et dynamisk bibliotek (f.eks. .dll (Windows) eller .so (Unix) filer) med et definert grensesnitt (API)). I noen implementasjoner blir nettleser-funksjonene aktivert via JavaScript, Java-klasser eller gjennom et C-grensesnitt. Når en nettleser blir startet opp, finner den alle plug-ins som er registrert og installert på maskinen. Når det kommer en for nettleseren ukjent MIME type blir header-informasjonen eller filnavnet brukt for å identifisere riktig plug-in. Denne blir lastet inn i minneområdet til nettleseren, instansiert og initialisert. I tillegg blir datastrømmen, parametre og handles (pekere) til vinduer på skjermen gitt videre til selve plug-in'en. Det finnes også spesielle tags som styrer bruken av plug-ins (f.eks EMBED, SRC). Implementeringsdetaljer for plug-ins finnes på utviklersidene til firmaene som lager nettleser software [?, ?].

En ulempe ved plug-ins er at de er implementasjonsspesifikke. Dermed trengs det forskjellige implementasjoner for forskjellige plattformer.

I dagens nettleserne brukes plug-ins for f.eks å vise PNG-kodete bilder og div. video applikasjoner. Men det er også aktuelt for spesielle applikasjoner og det kunne tenkes at grafikkbiblioteket GPGS implementeres som plugin.

2.6 Web Server og Web Gateway

En **Web server** er en datamaskin og programvare som har tre hovedfunksjoner:

- lagre dokumenter kodet i HTML og annet materiale,
- et program som mottar forespørsler fra nettlesere, og som sender data tilbake til nettleseren,
- Gatewayer som kan generere Web innhold (dvs. HTML dokumenter) og dermed gi tilgang til innhold som ikke foreligger som Web-dokument.

En **Web Gateway** er en tjeneste som gir tilgang til ressurser som ellers ikke ville vært tilgjengelig gjennom en nettleser. I tillegg til rene konverteringsfunksjoner må en slik gateway bl. a. håndtere statusinformasjon o.l. som skjuler seg i tradisjonelle brukergrensesnitt og anvendelser.

Generelt finnes det følgende muligheter for en Web gateway:

- Common Gateway Interface (CGI)
- Server-Side Includes (SSI)
- Stand-alone server
- Ulike teknikker på klienten

Gatewayer brukes for firma-interne applikasjoner, databaser eller eldre programware. De gir tilgang til data fra forskjellige kilder, bl.a. relasjonsdatabaser, og kan brukes for å håndtere nærmest all informasjon som ikke er designet med et HTML-grensesnitt. Disse gatewayene aksesserer ikke-HTML informasjon, konverterer det til HTML format (eller andre formater som en Web browser kan vise) og overfører informasjonen med HTTP protokollen. En slik gateway kan implementeres på klientsiden, på tjenersiden eller som en kombinasjon. Vi har beskrevet dette i egne kapitler nedenfor.

2.7 Uniform Resource Locator (URL)

URL er grunnlaget for å adressere ressurser på WWW. En URL angir protokoll (f.eks. http), server, portnummer, filnavn (path) eller tjeneste, samt mulige parametre. For noen av parametre finnes det default-verdier, dersom det angis URL delvis (partial URL). Filnavn og filtype er ofte koblet. Det generelle formatet er

```
protocol://host:port/path?parameter
```

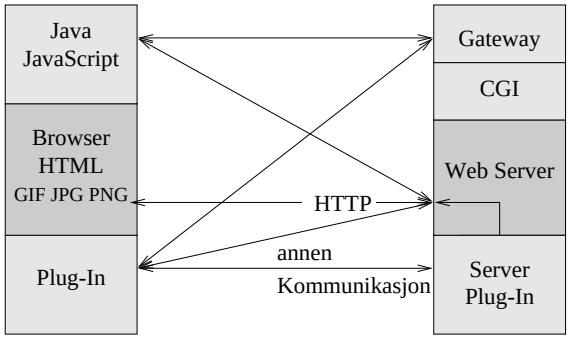
URL brukes både i HTML-dokumenter og i nettleseren for å angi ressurser. URL blir definert av IETF (Internet Engineering Task Force, http://www.ietf.org) i RFC 1738 [?].

2.8 MIME

Ifølge internett standarden RFC 822 kan emailmeldinger bare bestå av ren tekst. Med MIME (Multipurpose Internet Mail Extensions, RFC 2045-2049) kan også data overføres som bruker andre tegnsett, foreligger i et binært format eller består av flere deler.

MIME blir brukt utenfor det egentlige definisjonsområdet med stor suksess, bl.a. for å definere medietyper på WWW. MIME definerer bl.a. en versjonsnummer for å finne riktig programversjoner, innholdstype (content-type, se RFC 1049), og overføringsformat (Content-Transfer-Encoding).

Med MIME-typen *multipart* kan flere dokumenter samles i en datastrøm. Mens det opprinnelig ble brukt for å legge ved flere dokumenter i en epost-melding, kan det utnyttes i nettlesere for å overføre billedsekvenser. Bildene blir



Figur 1: Komponenter i WWW

sendt i riktig rekkefølge separert med multipart-headeren, og vises direkte i nettleseren. Dette gir visse muligheter for å overføre animasjoner på WWW.

3 Grafikkformater

I dette avsnittet beskriver vi grafikkformatene som brukes i Web-sammenheng. GIF og jpeg er tradisjonelle filformater, som har vært i bruk lenge, mens PNG er en etterfølger til GIF-formatet. En introduksjon om grafikkformater på WWW finnes på W3C sine Web sider [**?**].

3.1 PNG

PNG (Portable Network Graphics) er et utvidbart filformat for tapsfri komprimering og portabel lagring av rasterbilder, som ble implementert med sikte på bruk på nettlesere. PNG ble utviklet som erstatning for GIF formatet, som er patent-beskyttet. PNG er på mange måter også en erstatning for det ofte brukte TIFF formatet. Indekserte farger, gråverdibilder, true color bilder, og alfa-kanal er støttet. Fargeoppløsningen strekker seg fra 1 til 16 bits per piksel (og kanal). PNG kan brukes i en pikselstrøm med en progressiv display-opsjon. PNG er robust og gir mulighet for å sjekke fil-integriteten, og enkel deteksjon av vanlige overføringsfeil. I tillegg kan PNG lagre gamma- og kromatisitets-data for en forbedret fargepresentasjon på forskjellige plattformer.

3.2 JPEG

JPEG (uttales "dsjay-peg") er en standard billedkomprimerings-mekanisme. JPEG er et akronym for *Joint Photographic Experts Group*, som er navnet på kommiteen som utviklet standarden. JPEG-formatet er utviklet for komprimering av farge- eller gråtonebilder som stammer fra naturlige scener fra den virkelige verdenen. Den er egnet for fotografier, fotorealistiske bilder og lignende, men egner seg i mindre grad for tekst, tegneserier eller strektegninger. JPEG håndterer bare enkeltbilder. For komprimering av filmsekvenser brukes MPEG (Motion Pictures Expert Group) standarden. (Det finnes også en M-JPEG standard for filmsekvenser).

JPEG bruker en ikke-reversibel komprimeringsmetode, noe som betyr at man mister informasjon under komprimeringen. Ved dekomprimering er bildet ikke identisk med utgangsbildet, men en tilnærming. JPEG er utviklet for å utnytte de begrensningene det menneskelige øye har. F.eks. blir små fargenyanser fjernet fordi disse oppfattes i mindre grad enn nyanser i lyssyrken. Dersom bildene skal analyseres med datamaskiner, kan disse små forandringene som introduseres av JPEG formatet gi problemer for algoritmene, selv om forskjellene knapt er synlige for det menneskelige øyet.

I JPEG kan komprimeringsgraden styres, f.eks. ved å angi hvor mye informasjonstap som tillates. Dermed kan den som genererer et bilde bestemme om filstørrelse eller kvalitet skal være avgjørende. Med lav kvalitet kan man få til små filer, noe som kan brukes for å indeksere og katalogisere bilder. I tillegg kan det også avgjøres under dekodingen av et bilde om man prioriterer hastighet mot kvalitet. For å øke hastigheten kan det brukes raske, men unøyaktige tilnærminger isteden. Denne metoden kan også brukes ved å kode et bilde, men her spiller hastighetsgevinsten ingen stor rolle.

3.3 GIF

Billedformatet GIF er et akronym for "Graphics Interchange Format", og er utviklet av Compuserve. GIF er utviklet for overføring og utveksling av rasterbilder slik at disse overføres og lagres uavhengig av hardware, display eller applikasjons-software. GIF deler bildet opp i blokker og underblokker, som inneholder relevante parametere og data som brukes for rekonstruksjonen av et bilde. En GIF datastrøm er en strøm av enheter og underenheter som representerer grafikk. Disse enhetene er vanligvis avhengige av hverandre og har samme styringsinformasjon. Derfor anbefales at en enkoder prøver å gruppere lignende data for å redusere overhead. Formatet antar at datatransporten er feilfritt, da formatet ikke har tiltak for å gjenkjenne eller korrigere feil. Selv om formatet ikke er beregnet for det, er det til en viss grad mulig å lagre animasjoner med det.

4 Teknikker i klienten

Vi har tidligere beskrevet plug-ins og helper applications som brukes til å utvide funksjonaliteten til en nettleser. Her vil vi beskrive teknikker som er spesielt vinklet mot å implementere interaksjon og logikk i websider som lastes ned.

4.1 Java og Java-Applets

Java er et objektorientert programmeringsspråk som spiller en spesiell rolle i WWW. Opprinnelig ble språket definert og introdusert av Sun Microsystems under navnet **Oak** for å utvikle programmer for elektroniske apparater.

Ved hjelp av Java kan interaksjoner i Websider programmeres. Java-programmer kompileres til systemuavhengig kode som kjøres i et Java-miljø. Slike miljø finnes tilgjengelig for et stort antall plattformer og operativsystem. Java-applikasjoner blir i stor grad utviklet implementasjonsuavhengig, og kan kjøres på alle omgivelser som inneholder et slikt Java miljø, noe de fleste web-klienter gjør. Det gjør at Java applets kan integreres i websider, slik at disse lastes ned fra en web server og kjøres i Webklienten.

Viktige egenskaper er:

- Java applets er et eksempel på mobil kode, som er utviklet og lagret på en maskin, og som så flyttes til en annen maskin på oppfordring.
- Java applets gjør Web-applikasjoner til ekte client/server applikasjoner, fordi Java-kode kan kjøre forretningslogikk på Web-klienten.
- Databasetilgang kan skje direkte fra browseren uten å involvere et gateway-program som kjøres på serveren. Et Java-program kan sende en request fra brukeren direkte til en remote database ved å bruke f.eks. JDBC.
- Animasjoner, spreadsheets, business transaksjoner, grafikk, etc. kan kjøres direkte i nettleseren.

For å utvikle distribuerte applikasjoner med Java finnes det flere muligheter. Eller: dersom det finnes et behov for å skrive en Java applikasjon, hvor en Java applet på en nettleser kaller et annet program på en annen maskin så har man følgende valg:

- å skrive egen kode for å håndtere kommunikasjonen.
- å bruke et distribuert mellomlag som f.eks. RMI, CORBA eller DCOM.

Den første muligheten er ikke veldig attraktiv, men for enkel kommunikasjon er den et reelt alternativ. Den andre muligheten blir mer og mer utbredt fordi implementasjonene stadig blir bedre og utviklingsmiljøene kraftigere og lettere å bruke.

Nedlasting og Kjøring av Java Applets

Nettleserens nedlastingsprosess består av:

- Brukeren velger en HTML side.
- Nettleseren lokaliserer siden og starter nedlastingen.
- Mens siden lastes ned starter tekstformatteringen.
- Grafikk lastes ned, dersom det finnes IMG eller FIG tags i HTML koden.
- Java Applets er markert med en APPLET tag. Eskempel: En Java applet ved navn `Norsigd.class` med størrelsen 120×100:

```
<APPLET CODE=Norsigd.class
        WIDTH=120 HEIGHT=100>
</APPLET>
```

- Basert på applet-tagen setter Web-leseren et eget område i websiden. Dette området bruker appleten senere til input og output.
- Applet-koden ligger som regel på samme maskin hvor HTML siden ligger.

- Nettleseren laster ned den angitte klassen og andre klasser denne bruker.
- Java/nettlesere har også lokale klasser som brukes i applets.
- Etter at appleten har blitt lastet ned, kjører nettleseren spesielle metoder, init() og start() som henholdsvis klargjør appleten og starter appleten.

Det første Java miljøet (Java 1.0) var sterkt begrenset i forhold til hva appleter kunne og ikke kunne gjøre. I nyere versjoner av Java (dvs. Java 1.1, 2 osv) er det lagt mye jobb i å gjøre Java-miljøet mer komplett og stabilt, slik at en Java applet kan få tilgang til større deler av klientplattformen dersom det er ønskelig.

Grafiske objekter og GUI kan programmeres i Java. Egne plattformuavhengige biblioteker implementerer basisfunksjonaliteten. Ved siden av 2D-biblioteker finnes det også biblioteker som støtter 3D-funksjonalitet.

4.2 Javascript

Javascript er et standardisert scriptspråk som integreres i en web-side og tolkes og kjøres i en nettleser. Språket har spesielle funksjoner for å interagere med nettlesere og web-sider, noe som gjør det egnet til å legge inn funksjonalitet i web-baserte applikasjoner. Javascript er standardisert og støttes av de fleste nettlesere. Det finnes også andre scriptspråk som kan legges ved på samme måte, f. eks. Microsoft sitt VBScript. Den praktiske forskjellen ligger i hva de ulike nettleserne støtter. Bruksområdet for Javascript er noe mer begrenset enn for Java. Javascript brukes hovedsakelig til ren interaksjon (dvs. brukergrensesnitt) og i liten grad til forretningslogikk.

De grafiske mulighetene for JavaScript er veldig begrenset. Stort sett gir JavaScript muligheten å animere HTML-objekter.

4.3 ActiveX

ActiveX-kontroller er Microsoft sin teknologi for å inkludere programmer i en nettleser. På samme måte som med Java kan en ActiveX-kontroll kjøre i andre miljøer enn en nettleser. En ActiveX-kontroll er egentlig et COM-objekt som følger bestemte regler for interaksjon med sine omgivelser. Siden COM-objekter også er sentrale i utviklingen applikasjoner på en Microsofts plattformer (Windows) vil det si at det å skrive en ActiveX-kontroll ikke er vesentlig forskjellig fra å skrive andre windowsprogrammer.

En ActiveX-kontroll kan få tilgang til maskinen og operativsystemet på samme måte som et vanlig program. Dette gir stor fleksibilitet med hensyn på hva som er mulig. Samtidig blir det lettere for en uvennlig kontroll å gjøre skade på den maskinen den kjøres på. Microsoft addresserer dette ved å tilby signering av ActiveX-kontroller. Det vil si at brukeren må godkjenne kontrollen basert på dens signatur før den får tilgang til systemet.

Den tette integrasjonen med Microsofts plattformer er også en ulempe. ActiveX-kontroller er i praksis ikke tilgjengelig på andre web- plattformer. Likevel gjør den store utbredelsen av Microsoft sine plattformer at ActiveX-kontroller er mye brukt. Bruksområdet for en ActiveX-kontroll i en webside er utgangspunktet det samme som for en Java applet. I praksis vil det kunne være forskjeller fordi knyttningen til klientplattformen er forskjellig.

4.4 Broadway

I versjonen X11R6.3 av **X Window System**, som har fått kodenavn **Broadway** er det mulig å bruke en plug-in i en nettleser for å vise X-applikasjoner i nettleseren. Med Broadway support i nettleseren kan websider starte X-applikasjoner, som enten kommer opp i separate vinduer på desktoppen, eller innenfor en webside. Applikasjonen har de samme kjennetegnene som en vanlig X applikasjon. Dermed kan applikasjoner ha sitt display på en nettleser uten at disse må portes til nye programmeringsspråk som f.eks. Java.

Da kommunikasjon mellom server og klient kan være temmelig båndbreddekrevende, brukes Low-Bandwidth X (LBX) eller X.fast for å komprimere datamengden. Det finnes lignende teknologier som også egner seg for å mobile applikasjoner. Som eksempel nevner vi Tarantella, som bruker en utvidelse av X-protokollen med sterk komprimering). Tilsvarende produkter finnes også for windows-applikasjoner.

4.5 VRML

VRML (Virtual Reality Modelling Language) brukes for å visualisere tredimensjonale geometrimodeller. VRML i versjon 1.0 er et subset av OpenInventor 3D filformatet når det gjelder grafikk. I tillegg finnes det utvidelser for navigering (WWW inline og WWW anchor) og for å styre oppløsningsgraden for en 3D modell. Filene som

inneholder VRML objekter får .wrl som extension. I en nettleser blir fremvisning av innholdet oftest implementert som browser-plug-in.

En VRML fil består av en node i Inventor filformat. Denne noden kan bestå av et valgfritt antall undernoder. En 3D scene består bl.a. av punktlyskilder, spots, lys med gitt retning, kameraer og koordinattransformasjoner. Som geometriprimitiver finnes det kule, sylinder, kjegle og polygon-mesh. I tillegg kommer materialbeskrivelser og teksturer.

VRML 2.0 er en videreutvikling, som i tillegg er interaktivt styrbar. Denne standarden gir mulighet til dynamiske, skalerbare 3D omgivelser med en høy andel av objekter som er i bevegelse. Avstandsavhengig audio, behavioural models, animasjon og sensorer er blant de mulighetene som skal gjøre cyberspace mer realistisk. Proogrammeringsspråket Java of JavaScript kan også benyttes som utvidelser.

4.6 CandleWeb

En teknikk spesielt egnet for presentasjon av datagrafikk på WWW ble utviklet av det norske firmaet CandleWeb (http://www.candleweb.no). Utgangspunktet programmeringsspråket **Å**, som er et subset av programmeringsspråket C, uten strukturer og pekere. Derimot finnes det utvidelser for grafikk objekter, polygoner, plysplines, tekst, bitmaps, og input-objekter. Ved å bruke genereringsverktøyet *Awethor* kan Å-kode genereres interaktivt fra et grafisk verktøy.

Å skiller seg ut fra andre programmeringsspråk ved at grafikk og koden kobles sammen på en spesiell måte. Det finnes en direkte forbindelse mellom variabler i programmet og grafikkobjektene ved hjelp av **empowered variables**, slik at endringer i en verdi i programmet utgjør en endring i de grafiske objektene som har denne verdien som parameter. Ved å kompilere Å til Java kan disse kjøres direkte i en nettleser.

Konseptet bruker også spesielt kodet grafikk ved navn QDV (Quick and Dirty Vector graphics), som er raskere og bruker mindre filstørrelser enn tilsvarende GIF eller JPEG filer. Dermed redusereres nedlastings- og fremvisningstiden. Fremvisningen i browseren gjøres i en Java-applet. QDV har muligheten til å legge inn GIF og JPEG filer, linjer, polygoner, splines, etc. I tilleg finnes det muligheter for pekere, HTML-tekst, og animasjoner.

Det finnes lignende teknikker fra andre tilbydere, f.eks. Macromedia Shockwave, hvor grafiske presentasjoner vises frem ved hjelp av en Plug-In.

4.7 Multimedia

Standardene innen multimedia har stort sett oppgaver innen beskrivelse, koding og synkronisering av mediestrømmer, og har derfor liten tilknytning til vårt tema. Innen multimedia og digital TV finnes det flere standarder og forslag for å integrere disse med WWW. Standardene fra MPEG (Motion Picture Expert Group), spesielt MPEG-4 og MPEG-7, MHEG (Multimedia and Hypermedia information coding Expert Group) er omfattende normeringsforslag som dekker bruken av multimedia fra koding over sykronisering til presentasjon av innhold.

SMIL er et forslag som bygger på utvidelser av HTML, hvis hovedoppgave er å beskrive den logiske oppbyggingen av presentasjoner med flere medietyper og gi informasjon om synkronisering av datastrømmene. For bruk innen digital TV finnes det flere forslag som bygger på utvidelser av HTML, hvor hovedmålet er å integrere WWW med TV-teknikk, og å beskrive brukerinteraksjon (f.eks. BHTML eller ATVEF).

5 Tjenersiden

Applikasjoner som beveger seg litt utover det elementære (dvs. å presentere statiske filer) krever et eget apparat på tjenersiden. Egne tjenerprosesser har til oppgave å ta imot spesielle forespørsler og returnere et resultat på bakgrunn av innholdet i forespørselen. Vi har valgt å beskrive tre klasser av slike tjenerside-applikasjoner. Klassene er definert etter hvilke metoder som benyttes for å motta og sende forespørsler. Disse er vanlige metoder for å håndtere HTTP-forespørsler (så som CGI), standard mellomvareløsninger som CORBA og MPEG, samt mer proprietære løsninger så som RealVideo.

5.1 HTTP-håndtering

En HTTP-forespørsel håndteres som regel ved hjelp av CGI. Det vil si at en HTTP-forespørsel sendes til en web-tjener som fordeler ansvaret for behandlingen av forespørselen videre. Dette gjøres f. eks. ved at det startes en prosess som mottar data via CGI, prosesserer forespørselen og returnerer resultatet. Den nye prosessen som startes kan f. eks. skrives i Perl, Java eller Lisp. Hvordan selve prosesseringen foregår varierer fra installasjon til installasjon. Det enkleste er å la

web-tjeneren starte en helt egen prosess som gjør selve prosesseringen. Andre løsninger har et run-time system som f. eks. benytter en pool av tilgjengelige prosesser for å øke effektiviteten. Dette er f. eks. vanlig ved bruk av Java servlets. Microsofts ASP (Active Server Pages) er en annen måte å gjøre slik prosessering på. Her kjøres spesielle script i selve web-tjeneren.

En prosess eller et script som startes på denne måten får tilgang til en del variable. Disse kan grovt sett deles i tre: Klient-, server- og forespørsel-spesifikke variable, som gir nærmere opplysninger om programvare, brukeren, datatyper, navn på ressurser, o.l. Ved hjelp av postmetoden i HTTP og forms i HTML kan også applikasjonsorienterte variabler overføres. Variable kan også overføres som en del av en URL (bak et "?"-tegn). Mer om CGI og HTTP håndtering finnes i praktiske Håndbøker om emnet, f.eks. [**?**].

En annen, mindre fleksibel løsning er "server-side includes" (SSI), dvs. spesielle merker i en HTML-fil som byttes ut med aktuelt innhold før siden sendes til klienten. Mens forespørselen behandles byttes SSI-tags i HTML-dokumentet ut med sitt endelige innhold. Slike tags brukes vanligvis til å inkludere dato, filnavn o.l., men kan også brukes til å hente inn headinger, menyer eller andre grafiske elementer.

5.2 Mellomvare og standardiserte protokoller

Ulike mellomvareløsninger kan benyttes for å sende data til og fra web-tjenere. Disse er ofte implementert som egne applikasjoner, men kan også være en del av selve web-tjeneren. Det spesielle med en slik løsning er at kommunikasjonen til og fra den aktive delen av applikasjonen ikke går direkte via HTTP (selv om HTTP kan brukes som bærer). Det kan være inter-prosess-kall som i CORBA, RPC og DCOM eller streamede data som i MPEG. Fordelen med slike løsninger er at de har innebygde mekanismer for å skreddersy kommunikasjonen, noe som ikke finnes i HTTP.

5.3 Proprietære løsninger

Proprietære løsninger er som regel tilpasset spesielle bruksområder. Et eksempel er RealVideo som tilbyr egne applikasjoner for å levere video over nettet. Tilsvarende tilbyr en del web-utviklingsverktøy egne tjenere som tar imot og leverer data til spesielle klienter (f. eks. en plug-in).

6 Integrasjon av WWW og grafikkbiblioteker

I dette avsnittet undersøker vi hvordan grafikkbiblioteket GPGS kan integreres med WWW ved hjelp av Java og andre teknikker. GPGS brukes som eksempel for et grafikkbibliotek med generell funksjonalitet.

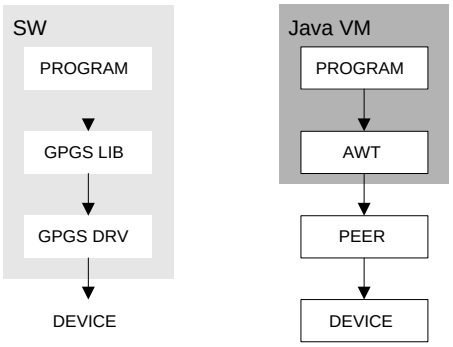
GPGS er et grafikkbibliotek utviklet av NORSIGD siden 1974. Målet var å utvikle et systemuavhengig grafikkbibliotek i forhold til operativsystem, vindusystem, output devices, mm. Biblioteket er fortsatt i bruk og det har blitt utviklet endel applikasjoner som bruker funksjonaliteten til biblioteket. Ved mer utstrakt bruk av WWW-teknologi, f.eks. i forbindelse med et intranett, må muligheter for en overgang til Web plattformen vurderes. Spesielt et samspill mellom GPGS og Java virker interessant. Begge har rutiner som kan brukes til å behandle grafikk. Disse rutinene er ordnet i et grafikksystem med en bestemt struktur.

GPGS har en overordnet struktur som kan deles i fire: selve programmet, GPGS-biblioteket, devicedriver og selve devicet (se figur 2 til venstre). Grafikksystemet i en applet kan også deles i fire hovedelementer: selve programmet, awt-komponenter med tilhørende Windows-komponenter (peers) og selve devicet (se figur 2 til høyre).

En viktig forskjell mellom GPGS og appletmodellen er tilgangen til skjermområdet. En generell GPGS-applikasjon har i prinsippet tilgang til hele skjermområdet. En Java-applet har bare direkte tilgang til bestemte områder på skjermen. En applet har derfor ikke mulighet til å påvirke annet enn det som er definert innen dens eget skjermområde. Et eksempel på dette er en Java-applet som henter opp en web-side, som kommer opp i et skjermområde som er kontrollert av nettleseren og som appleten ikke får tilgang til.

Dersom en Java-applet skal benytte seg av et GPGS-bibliotek er det nødvendig å se på integrasjon mellom det appleten har direkte tilgang til (i utgangspunktet AWT komponenter) og det GPGS genererer. Det finnes grovt sett to måter å integrere applet-modellen og GPGS-modellen på i forhold til dette:

1. Appleten sender et kall til GPGS-biblioteket. Dataene behandles og resul-



Figur 2: GPGS-modellen og applet-modellen

tatet sendes tilbake til appleten og føres tilbake til AWT.

2. Appleten sender et kall til GPGS-biblioteket. Dataene behandles og føres videre i GPGS-modellen.

Forskjellen mellom disse er i hvilken grad de gir vindusintegrasjon, dvs. i hvilken grad Java-programmet får tilgang til vinduskomponenter via AWT og ikke bare via GPGS. Den første varianten ovenfor gir ikke vindusintegrasjon, mens den andre gir vindusintegrasjon. En fordel med vindusintegrasjon er det at deler av programmet (f. eks. eksisterende klasser) kan bruke AWT mens deler av programmet som kan dra nytte av spesielle GPGS-funksjoner bruker det.

6.1 Ren Java

En mulighet for å integrere GPGS med Java er å implementere GPGS direkte i Java. Da må selve biblioteket portes til Java samtidig som det må lages en driver som kobler dette biblioteket til AWT (se figur 3)[1].

De største fordelen med en slik løsning er at den i prinsippet kan kjøres overalt hvor det finnes en Java-implementasjon, og at løsningen gir god vindusintegrasjon. De største ulempene er at hele biblioteket må skrives om, samt at hastigheten på eksisterende Java-implementasjoner gjør at de ikke egner seg for tyngre grafikkapplikasjoner.

6.2 Lokalt bibliotek

En annen mulighet er å definere et JNI (Java Native Interface) grensesnitt mellom en eksisterende GPGS implementasjon og Java. Selve programmet vil da skrives i Java mens alle kall til GPGS behandles utenfor den virtuelle maskinen og kjøres på klienten. En slik løsning kan designes både med og uten vindusintegrasjon.

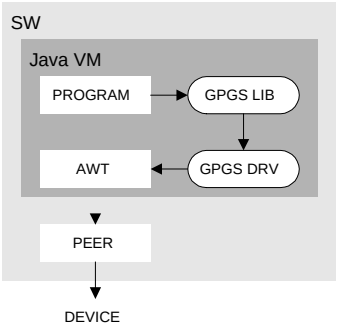
Løsningen uten vindusintegrasjon krever bare at det defineres et JNI grensesnitt som Java-programmet kan bruke til å kontakte GPGS-biblioteket. Løsningen med vindusintegrasjon krever i tillegg at det lages en egen GPGS-driver og en Java-stubb som sender resultatet videre i applet-modellen. Med vindusintegrasjon kan eksisterende biblioteker brukes som de er, mens de til gjengjeld må installeres separat på alle steder hvor klienten skal kjøres.

6.3 GPGS-tjener

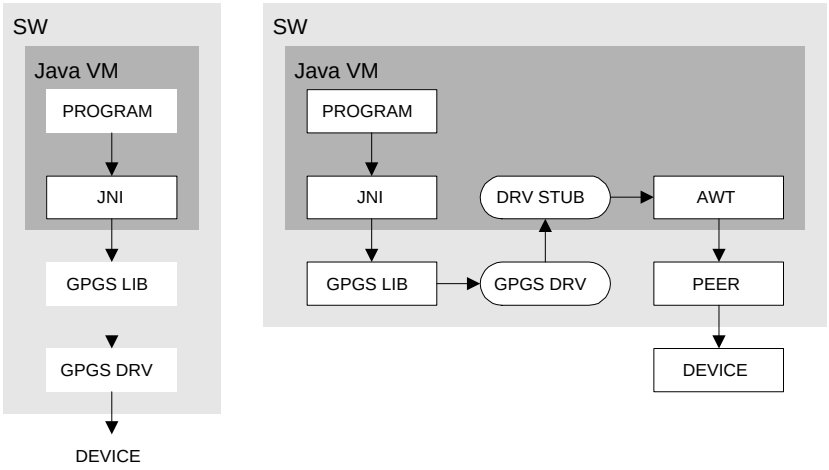
En tredje mulighet er å opprette en egen tjener som behandler GPGS-forespørsler og benytte en ORB (Object Request Broker, f.eks. en CORBA implementasjon) eller en annen form for kommunikasjon mellom Java-klienten og GPGS-tjeneren. Her må det lages en stubb i klienten som fanger opp alle kall til GPGS og så sender disse videre til GPGS-tjeneren. Der fanger en ny stubb opp forespørselen og behandler denne. Resultatet sendes til en driver-stubb som returnerer resultatet til klienten for videre prosessering, enten uten (figur 5) eller med (figur 6) vindusintegrasjon.

En fordel med en slik løsning er at den kan benytte seg av eksisterende GPGS-implementasjoner. En annen fordel er at GPGS-tjeneren kan kjøre på en maskin som er egnet for tunge grafikkapplikasjoner og at flere klienter kan dele denne maskinen. En stor ulempe er at ethvert kall til GPGS må sendes gjennom en

[1] Vi har valgt å å beholde den konseptuelle delingen mellom bibliotek og drivere selv om AWT i seg selv gjør noe av den samme jobben som GPGS-drivere.



Figur 3: Ren Java



Figur 4: Lokalt bibliotek uten (venstre) og med (høyre) vindusintegrasjon.

ORB (og dermed muligens over et nettverk). Hvor stor ulempe dette vil være er avhengig av hvordan de ulike stubbene kan optimaliseres (f. eks. ved å cache eller gruppere forespørsler).

Løsningen med vindusintegrasjon kan her være spesielt interessant fordi deler av grafikksystemet dermed kan kjøres lokalt i klienten (som i en vanlig applet). Dermed er bare kall spesielt til GPGS avhengig av en distribuert ORB.

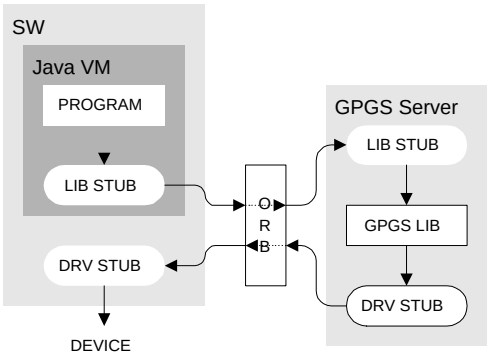
6.4 Alternativ vindusintegrasjon

I løsningene ovenfor er vindusintegrasjon realisert ved å tilbakeføre resultater fra GPGS til applet-modellen. Et annet alternativ kan være å erstatte AWT med tilsvarende GPGS-funksjoner (se figur 7). Et problem med en slik løsning er at det ikke er tillatt å laste ned klasser i pakker som begynner med "java." (f. eks. "java.awt"). Løsningen kan likevel realiseres, f. eks. ved å behandle alle klasser som leses inn (bytte ut java.awt, refereanser med tilsvarende AWTX referanser, eller ved å installere en spesiell java.awt på klienten).

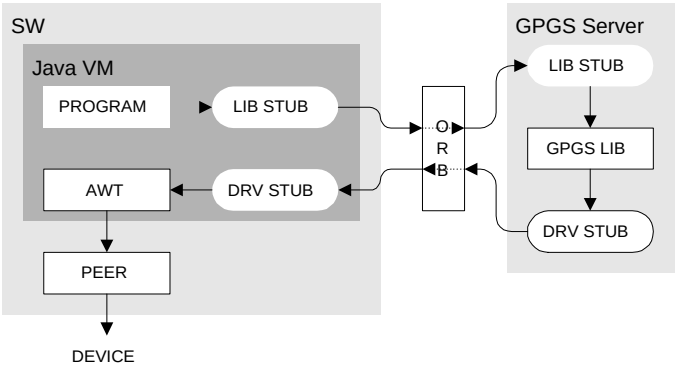
6.5 Eksisterende applikasjoner

For å gjøre eksisterende applikasjoner tilgjengelig via web er det i utgangspunktet to ting som må til. Først må input fra brukeren fanges opp i en nettleser og sendes til den maskinen hvor applikasjonen er. Etter prosessering må så resultatet fanges opp og sendes tilbake til nettleseren.

Først beskriver vi hvordan dette kan løses for en eksisterende applikasjon som har en enkel statisk struktur, mens vi deretter beskriver en situasjon hvor den eksisterende applikasjonen har en mer dynamisk struktur. Begge disse løsningene passer likevel best for batch-aktige prosesser hvor mesteparten av brukerinteraksjonen foregår først, så en kjøring på tjeneren som resulterer i et resultat som vises på klienten.



Figur 5: GPGS-tjener uten vindusintegrasjon



Figur 6: GPGS-tjener med vindusintegrasjon

Mer komplekse interaksjonsmekanismer fordrer generelt tilgang til den eksisterende applikasjonen gjennom en API.

Statisk struktur

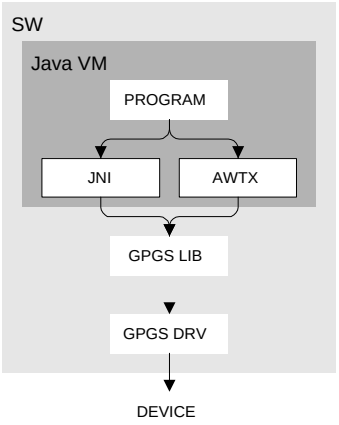
Med en statisk struktur menes at applikasjonen har et kjent antall inputparametre og genererer et bestemt resultat. Det vil si at parametrene kan fanges opp i en klient og sendes til applikasjonen. Når resultatet er klart, fanges dette opp og sendes tilbake til klienten.

I klienten kan enkle parametersett fanges opp som HTML-forms. For mer komplekse parametersett (f. eks. med mye avhengigheter eller valideringsregler) kan Java eller andre klienprogrammeringsteknikker benyttes. Parameterverdiene må så sendes til et mottaksapparat på en tjener. Her overføres alle parametre fra forespørselen til applikasjonen. Hvordan dette gjøres er helt avhengig av hvordan applikasjonen virker. Eksempler kan være konfigureringsparametre, prorammeringsgrensesnitt (API) eller ren "skjermavlesing" (screen-scraping). For applikasjoner som bruker grafikkbiblioteket GPGS vil det som oftest være de to første alternativene som er mest aktuelle.

Konfigureringsparametre brukes hovedsakelig i batch-orienterte programmer. Her overføres parametre ved starten av programmet. Det kan være rene oppstartsparametre (kommandolinje), konfigureringsparametre i en fil eller omgivelsesvariable, eller en del av en pipe (prosess-til-prosess kommunikasjon).

API kan med fordel benyttes dersom slikt finnes. Slike programmeringsgrensesnitt kan f. eks. være gitt som bestemte bibliotekskall, ORB-kall eller som en mer lineær protokoll (f. eks. over et nettverk).

Screen-scraping vil si å lese av skjermbufferet eller en annen representasjon av det som faktisk vises på skjermen. Tilsvarende puttes parametre inn via applikasjonens mekanismer for brukerinteraksjon. Denne metoden kan være nyttig dersom ingen



Figur 7: Alternativ vindusintegrasjon

andre alternativer finnes. Metoden krever høy teknisk integrasjon mot den infrastrukturen applikasjonen benytter (f.eks. 3270 eller telnet), men kan være effektiv dersom denne infrastrukturen er velkjent.

Etter dette fortsetter kjøringen av applikasjonen som før, bortsett fra at resultatet må fanges opp. Dette kan f. eks. gjøres ved hjelp av spesielle drivere, egne API eller screen-scraping. For applikasjoner som bruker GPGS vil den eksisterende jpeg-driveren kunne være en god løsning.

Spesielle drivere som har mulighet til å sende resultatet til klienten, enten direkte (til en plug-in e.l.) eller som et bilde i en webside.

API kan brukes til å lage noe som tilsvarer skjermdriver. Som ovenfor fanger den opp resultatet og formatterer det på en egnet måte og sender det til klienten.

Screen-scraping kan igjen brukes ved at resultatet leses av skjermen, formateres på en egnet måte (f. eks. som et bilde i en web-side) og sendes til klienten. Dette forutsetter tilgang til disse ressursene.

Felles for alle disse løsningene er at de i tillegg til å vise et statisk resultat kan bruke klientens interaksjonsmekanismer for å la brukeren operere på resultatet, f. eks. velge et utsnitt. F. eks. kan en i ren HTML bruke imagemaps e.l. og i Java egne klasser som tar imot brukerens nye parametre og sender disse tilbake til klienten.

Dynamisk struktur

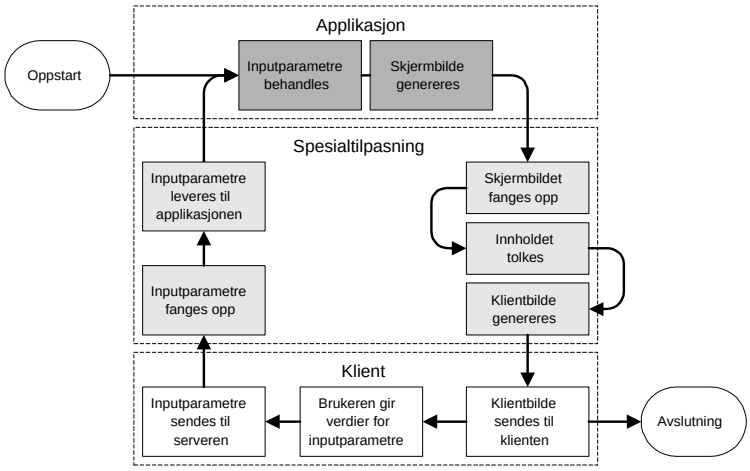
En applikasjon kan sies å ha en dynamisk struktur dersom det ikke er mulig eller ønskelig å beskrive alle mulige inpu-parametre på forhånd. Det kan være fordi det blir et uforholdsmessig stort sett hvor bare begrensede deler benyttes i en bestemt kjøring, eller fordi settet med mulige input-parametre endres ofte.

En slik applikasjon krever en mer komplisert løsning fordi det er nødvendig å tolke resultatet for å kunne definere brukerinteraksjon. Det vil si at det lages en løkke hvor applikasjonen først lager et skjermbilde for å sette inn parametre. Dette skjermbildet fanges opp som beskrevet ovenfor og tolkes. På bakgrunn av dette genereres det et klientbilde som sendes til klienten. Inputparametrene fanges så opp og returneres til programmet slik som beskrevet ovenfor. Dermed startes neste rundgang. Denne løkken er skissert i figur 8.

7 Konklusjon

Grafiske metoder brukes i de fleste applikasjoner som har et brukergrensensitt. Etterhvert har slike applikasjoner blitt gjort tilgjengelige på Web med forskjellige tekniske metoder. For endel eldre grafikkbiblioteker, som inntil nå var utelukket fra bruk på Web finnes det flere muligheter for å integreres med det nye mediet. Valget av metode er avhengig av applikasjonen og andre krav slik at det ikke er mulig å gi en generell anbefaling.

Utviklingen av WWW vil ikke stoppe med det første, og mange nye utfordringer vil komme. Vi vil få mer mobilt utstyr som PDA og mobiltelefoner, som vil brukes med grafisk bruker-



Figur 8: Prosesseringsgangen ved dynamisk struktur

grensesnitt. Derfor vil vi trenge proxy-løsninger og konvertere slik at både båndbredde og de spesielle egenskapene til apparatene untyttes optimalt.

Slikt utstyr er optimalisert for å være lite og bærbart fremfor å tilby høy kvalitet på grafikk. Å bruke grafiske teknikker fra WWW på slike bærbare løsninger vil gjøre det mulig å lage sømløse applikasjoner som kan anvendes på både kontor-datamaskiner og på mobilt utstyr. Fortsatt trengs det forskning og utprøving av metoder som gjør sømløs aksess på dokumenter og ressurser mulig.

Denne rapporten er et første steg for å presentere teknikker for sømløshet for spesielle applikasjonsklasser. Ved å bruke WWW som eneste brukergrensesnitt er det første integrasjonssteget nådd. I fremtiden vil en adapsjon til nye tekniske utviklinger begrense seg til å tilpasse disse til Web grensesnittet.

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: .....