Programutvikling med "VISUAL BASIC".

Av Steinar Thorvaldsen, Høgskolen i Tromsø. Copyright 24.11.94, sist rev. 28.11.2K
steinar@hitos.no

Del 1 Innledning.

1. Målsetting med kurset

- Gi en innføring i design av programvare til undervisning
- Gi en innføring i programutvikling for PC
- Gi en innføring av implementasjon (systemering, programmering m.m.) av moderne grafiske brukergrensesnitt

Kurset tilbys studenter som tar videreutdanningskurset IT-2 ved HiTø, AFL

2. Design av grafiske brukergrensesnitt

- Brukeren står sentralt og kontrollerer programmet (ikke motsatt)
- Brukeren skal kunne håndtere skjermbildene mest mulig intuitivt
- Klar og estetisk oppbygging av skjermbilder
- Rask tilbakemelding til brukeren
- Tilgivende skjermbilder, ikke straffende.

3. Windows programmering

- Vi lærer programutvikling innen Windows-standarden.
- Visual Basic har den laveste terskelen. Viktigste konkurrent er Borland Delphi
- Utviklingsverktøy som Visual Basic gjør Windows programmeringen mye enklere en tradisjonell programmering. Brukeren setter sammen skjermbildene med enkle operasjoner (peke, plassere, skalere, m.m.), mens koden består av korte sekvenser som velges ved å klikke med musa.
- Kalles ofte visuell objekt-orientert programmering (OOP)

4. Arbeidsmåte

- Litteratur: Se litteraturlisten under.
- Bruk *online help* som er innebygd i Visual Basic.
- Kompendium, øvinger og løsningsforslag ligger/kommer på nettet.
- Husk: Øving gjør mester!
- Gitte tidsfrister for alle øvingsoppgaver.

5. Litteratur

Støttebøker i Visual Basic:
Bernt Bertheusen: *Visual Basic 6.0 i praksis.*
EDB-kunnskap, 1999.
Se: http://www.edbkunnskap.no/visba60.html
Deler av pensum ligger på nettet: http://www.edbkunnskap.no/vbbonus/ Passord: Se side 13 i boka.

Boka dekker det egentlige Visual Basic (VB). Det finnes også en *miniversjon* av VB, Visual Basic for Applications (VBA), som følger med Officepakken. VBA er noe anonymt og bortgjemt i Office (i menysystemet må du aktivere *Vis-Verktøylinjer-Visual Basic*). Det kan være like greit å starte med VB for å bli kjent med selve systemet og hva det kan brukes til. De grunnleggende prinsippene i VB vil gjelde for alle Microsoft sine produkter.

Det finnes mange andre bøker som dekker Visual Basic, spesielt **på engelsk:**
Greg Perry m.fl.: Teach yourself Visual Basic 6 in 24 hours. Sams publishing, 1998. ISBN 0-672-31306-5 Denne boka koster et par hundre kroner og inneholder en **CD** med øvingsversjon av Visual Basic 5.0. Denne versjonen kan brukes til alt bortsett fra å lage ferdige (kompilerte) versjoner av programmene. Dette er en trolig den billigste måten å skaffe seg programmet på. Den kan kjøpes via Internett ved å klikke på lenken over.

Støttebok i Design:
Minken/ Stenseth: Brukerorientert programdesign. NLS 1998.

6. Nødvendig programvare

Presentasjonsprogram
Internett med hjemmeside og fjernundervisning for IT-2.

Utviklingsverktøy
Microsoft Visual Basic (VB) for Windows Ver. 6.0. Se punktet over om litteratur. VB inngår i Microsofts framtidige .NET-strategi som fra høsten 2001 planlegger å gjøre det mulig å programmere over web med fritt valg blant ca. 15 ulike programmeringsspråk. Disse vil kommunisere over nettet med den nye standarden XML (eXtensible Markup Language).

Online Help
Visual Basic Help inkl. Basic Help, Books Online...(følger med Visual Basic). Mye er også på Internett.

7. Lynkurs i VB

(Se: Bernt Bertheussen: Visual Basic 6.0 i praksis. EDB-kunnskap 1999. Lærebokas kap 1-3)

I startfasen vil visuell programmering for de fleste fortone seg som noe fremmed og rotete. Omtrent like kaotisk som førsteinntrykket av Windows kan være for den helt nye bruker. I visuell programmering er tankegangen nemlig noe annerledes enn i tradisjonell programmering som Pascal. I Pascal har man jo som oftest **ett program** med tilhørende prosedyrer å forholde seg til, mens i visuell programmering har man **mange samtidige objekter** hver med **sine** tilhørende prosedyrer å tenke på. Dette fører til at det er lett å miste oversikt alt fra starten av.

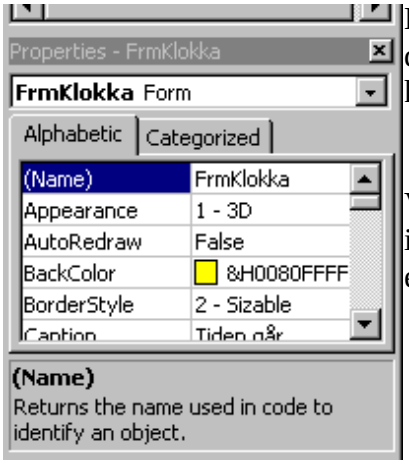
Men den høye inngangsterskelen tjener seg fort inn igjen ved at de programmene man lager er fleksible å bruke og lette å videreutvikle. Når vi bruker Visual Basic, vil programmene vi produserer ligge nær opp til et standard windows brukergrensesnitt.

Visual Basic (VB) har en forløper i de to norskutviklede produktene Mosaikk for Dos og Winix Toolkit for windows. Men VB er et mer generelt produkt enn disse, og dessuten er det mer "fullendt" enn de norske produktene noen gang ble. Fra og med versjon 2.0 av VB, stoppet videreutviklingen av de norske produktene opp. VB har i dag en stor brukermasse - også blant profesjonelle. Programmerere bruker det ofte som en snarvei i sin produktutvikling, mens spesielle finesser må legges til ved klassisk programmering (C-programmering eller Pascal for windows).

Under innlæringen av VB kan det kanskje være til hjelp å tenke på at i Visual Basic er det egentlig to hovedbegreper: **Objekt** og **Hendelse:**

A) Objekter.
Objekter er noe grafisk eller konkreter - noe som kan vises. Eksempler er bilder, tekster, menyer knapper osv.

Disse objektene kalles gjerne også "**kontrollere**". Vi har flere typer av dem i Visual Basic. De vises på skjermen i den såkalte "verktøykassa". Ved å dobbeltklikke på et objekt i verktøykassa, får du fram ett slikt objekt til bruk i ditt program.



Hvert objekt har for det første **egenskaper**, dvs. data knyttet til seg (engelsk: **properties**). Det kan være farge, navn, posisjon, størrelse o.l.

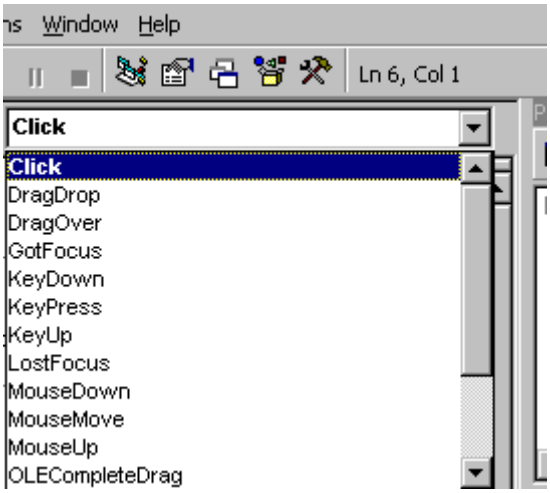
Ved å aktivere (klikke på) et av objektene du har i ditt program, kan du også få fram tilhørende egenskaper, slik det vises på bildet til venstre.

Et objekt har dessuten et sett tilhørende **metoder** (engelsk: "methods"), dvs. operasjoner som det kan gjøre. Det kan være å bevege seg, gjøres seg stor/liten eller sette igang noe annet.

Skulle vi definere hva vi mener med et objekt, kunne vi si de med følgende kortform: "Et objekt har en identitet, en tilstand og en oppførsel" (G. Booch, 1991) Objekter kan også skapes og forsvinne mens et program kjøres.

B) Hendelser
Hendelser (engelsk "event") er valg som brukeren gjør, ofte med musa.

Det finnes mange typer hendelser: Musa kan klikkes, dobbelklikkes, objektet kan dras eller slippes, osv.



Hva er så forskjellen på hendelse og metode? Jo, hendelser er noe **brukeren** gjør, mens metoder er noe mer internt **i programmet** - aksjoner som objektene selv kan gjøre.

Visuell programmering kan sammenliknes med set å sette opp et "**teaterstykke**" eller drama. Vi har en del skuespillere (=objekter) med sine kostymer (=egenskaper) og bevegelser og replikker (=metoder).

Men her blir det også en forskjell. For stykket skal kunne påvirkes av det som skjer i salen (=hendelser) fra publikum (=brukerne) sin side. På denne måten kan man lage fleksible og spennende programmer. Jeg har også sett noen eksperimentere med en slik type "interaktivt teater" der publikum skal kunne være med...

Visuell programmering kan også betegnes som "multi Pascal programmering". Det som tross alt gjør det hele litt lettere enn man har grunn til å frykte er at det hele er visuelt. Objektene tegnes opp konkret med sine egenskaper og roller, og legges så "i dvale" og lytter til de får en kommando/hendelse som vekker dem til aktivitet.

C) VIKTIG SKRIVEMÅTE.

Anta at vi har et objekt som heter **text1** . Da vil det som skrives etter objektnavn og punktum referere til en av objektets **egenskaper** eller **metoder**:

```
text1.Textcolor
text1.Move
```

Her er text1 = objektnavn, Textcolor= egenskap (properties) og Move en metode.

En annen ting som også er verd å merke seg er at alle objektene legges på et grunnobjekt som kalles **FORM**.

Formobjektene er "faderobjektet", dvs. eier andre objekter. Også kalt "Lerret".

Skal man referere til form, objekt og egenskap, skriver man:

```
form1!text1.Textcolor
```

For å holde god oversikt over de objekttypene man bruker i et program, har det blitt vanlig å la de **tre** første bokstavene i navnet referere til hva for type objekt det er. Se læreboka side 42 og 43.

D) ET EKSEMPEL

Vi kan nå starte opp Visual Basic...

Tips - andre progammer bør minimeres ved bruk av Visual Basic for å unngå for mye rot i bakgrunnen.

Helt til venstre i VB er verktøykassa med standard-objektene ("skuespillere"). En del av disse må vi bli kjent med.

Oppgave:
Lag et design som består av 2 knapper:

1. Knapp trykkes => dagens dato skal vises
2. Knapp trykkes => klokkas tid vises

Lag et kjørbart windowsprogram (exe-fil) til slutt.

Løsning:
Dobbelklikker først i verktøykassea på knapp-objektet.
En knapp lages på lerretet (formen). Velger så egenskap (properties), og endrer henholdsvis Name og Caption. Caption er en tekst som står på knappen (teksten som er "kapret").

Visk i redigeringsfeltet og skriv hva det skal stå på knappen.

Omdøp knappen ved å kalle den et nytt navn (let i properties) og skriv hva den skal hete (DatoKnapp) i stedet for Command1.

Egenskap: Endres fra -> Endres til:
Name Command1 -> DatoKnapp
Caption Command1 -> Vis dato

NB! Navnet og det som står på knappen trenger ikke være lik.
Det er lettere å huske navn enn å huske nummer på Command1, 2, 3, o.s.v.
Vi tager oss en knapp til på samme måte. Denne knappen skal brukes for å vise tid:

Egenskap Endres:
Name Command2 -> TidsKnapp
Caption Command2 -> Vis tid

Vi trenger også et textobjekt der resultatene skal vises, og dobbelklikker på A-objektet i verktøykassa. Objektet Text1 kommer fram.

Vi går så til properties for å gjøre en endring da vi ønsker at text'en på objektet i utgangspunktet skal være blank:

Egenskap: Endres fra ->Endres til
Name Text1 -> Text1
Caption Text1 -> (blank)

Nå er 3 "skuespillere" definert. (Vis dato, Vis tid, + tekst).

Hendelser.
Hvis vi klikker i Vis dato, skal datoen komme frem i textboksen.

Ved å dobbelklikke på Vis dato, henter den frem hendelsene knyttet til dette objektet.
Hendelsene ligger til høyre i det vinduet som kommer frem ved dobbelklikkingen.

Du ser at en rekke hendelser kan gis et innhold. Vi velge å programmere inn et innhold i hendelsen **Click**:

```
Private Sub DatoKnapp _ Click ( )
Text1.Text = "Dagens dato er " &Date$
'(henter dato fra systemklokka i windows).
End Sub
```

Det som skrives på linjen etter tøddeltegnet ' regnes som en kommentar eller forklaring fra programmereren. Ønsker vi å bruke variable, så opprettes disse ved hjelp av forstavelsen Dim, f.eks. Dim tall1 As Integer osv (se læreboka).

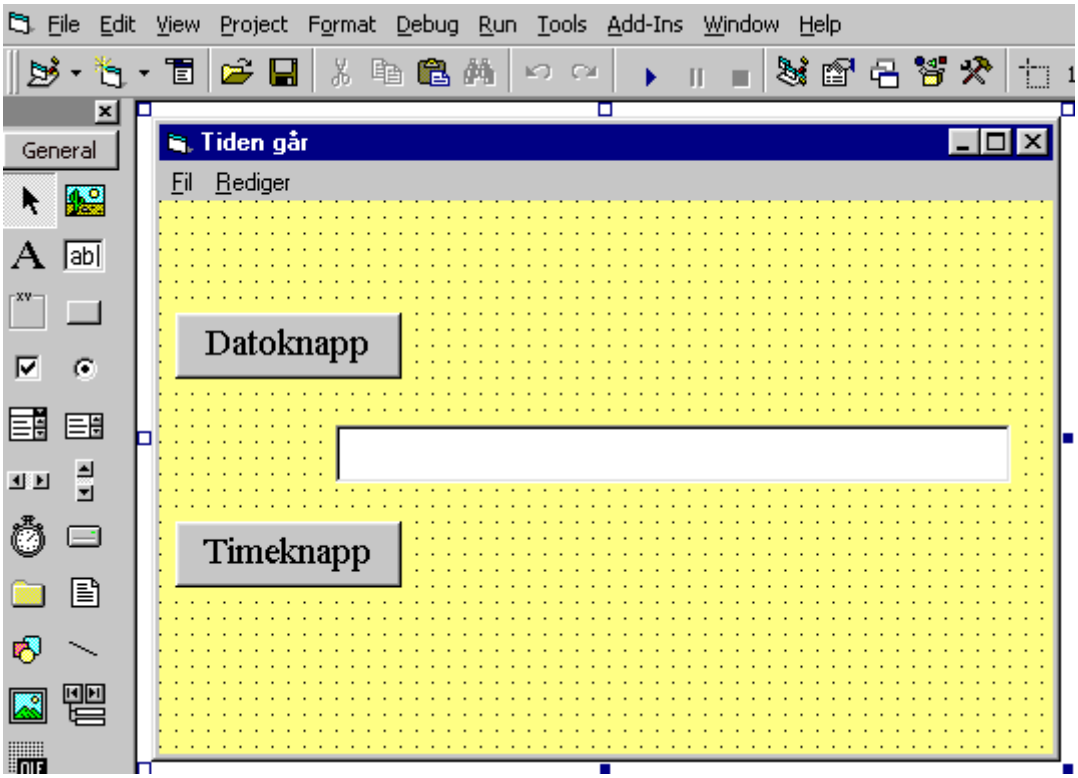
Tilsvarende kode lages for tidsknappen:

```
Private Sub Tidsknapp _ Click ( )
Text1.Text = "Klokka er " &Time$
End Sub
```

I tillegg kan vi også legge inn en respons i selve det bakenforliggende formobjektet:

```
Private Sub Form_Click ( )
Text1.Text = "Du bommet! Prøv på nytt!!!"
End Sub
```

Lukk så vinduet for hendelser, og gå tilbake til "lerretet" (form). Gå til Caption og omdøp form1 til: **Tiden går**. Skal du ha farge på bakgrunnsformen kan det også legges inn (gå til backcolour..).



Test nå programdesignet ved å gå opp på **Run** og Start.

Når du er fornøyd, kan du lage exe-fil av ditt program:

Skal programmet plasseres på en diskett, så sett inn diskett. Gå deretter til **File** + **Make exefile**
Kall det for f.eks.: klokka.exe.

I tillegg ønsker vi også å ta vare på kildeprogrammet. I dette tilfellet består det av to filer, ei prosjektfil (.vbp i versjon 4, 5 og 6, og .mak fil i VB versjon 3) og ei formfil (.frm). Hadde programmet bestått av flere former, ville det blitt tilsvarende antall former å lagre. Prosjekt-fila holder orden på de former som skal inngå i programmet.

Gå til File og trykk: **Save project AS.**
Filnavnet Project1 omdøpes til f.eks. Klokka.vbp.
Husk også å lagre formen (klokka1.frm).

Nå er Windowsprogrammet komplett for kjøring.
START fra filbehandling i windows og sjekk at du finner filnavnet: Klokka.exe.

MERK: Det kan være en liten detalj som gjør at programmet ikke starter i en del tilfeller der du ikke har installert Visual Basic på maskinen din. For f.eks. versjon 6.0 må Visual Basic's biblioteksfil **Msvbvm60.dll** (Ca 1-2 Mbyte) nemlig distribueres sammen med de programmene du lager. Kopier evt. denne sammen med det ferdiglagde programmet, slik at det alltid er der i de tilfeller du ikke har Visual Basic på maskina der programmet skal kjøres. Fila Msvbvm60.dll kan kopieres og distribueres fritt. Den legges ofte inn i Windows sin systemkatalog.
For versjon 4.0 må Visual Basic's biblioteksfil vb40032.dll (Ca 700 K) på tilsvarende måte distribueres sammen med de programmene du lager. For VB 4.0 finnes også fila vb40016.dll som kan brukes for å kjøre programmet med 16-bits versjoner av Windows. (Windows 3.x).

Del 2: Variable, valg og egne funksjoner

Les: Bernt Bertheussen: Visual Basic 6.0 i praksis. EDB-kunnskap 1999. Kap 4, 6, 7 og 9.

Neste del av læreboka omhandler if-setninger og datatyper. Jeg gir et kort sammendrag her. Kapittel 5 om løkker lar vi vente litt. Erfaringsmessig vil denne type klassisk programmering kreve modning, slik at dere som fra før har kjennskap til programmering vil ha en fordel, selv om skrivemåten (syntaksen) kan være noe uvant. Dere som ikke har programmert noe før må nok regne med en ekstra innsats for å komme opp denne motbakken.

1. Variabler.

En subrutine eller prosedyre er de steder i VB der man skriver inn kode. Subrutinen inneholder altså kode som angir hvilke handlinger vi vil at programmet skal utføre. I VB sine subrutiner, kan vi programmere omtrent på samme måte som det gjøres i andre programmeringsspråk. Vi definerer variable, regner, sammenlikner de variable og gjør våre valg. I programmering er det viktig å kjenne til at de variable kan være av forskjellige **datatyper**. Dessuten må vi vite at noen variable er kun lokale innen sin private subrutine, mens andre kan være mer globale for større deler av programmet.

LOKALE VARIABLE.
I VB defineres lokale variable inne i prosedyra (Subrutina) der den skal gjelde. Framfor den variable setter vi ordet **Dim**:

```
Dim Teller As ??????
```

Etter As kan vi sette den ønskede datatype, som kan være:

```
Integer -32768..+32767
Long større heltall
Single desimaltall
Currency store pengebeløp med opptil 4 desimaler
Double svært lange desimaltall
String svært mange tegn etter hverandre
Boolean True eller False
Object Enhver objektreferanse
Date fra 1. jan år 100 til 31.des år 9999
Variant generell datatype.
```

Hvis vi kutter ut As ????? når vi definerer den variable, setter VB den til å være av typen Variant (default datatype). Det finnes spesielle funksjoner som brukes til å konvertere data mellom de ulike datatyper.

Variable definert med Dim opprettes når prosedyra er i bruk, og slettes når den er ferdig med jobben.

I VB finnes også en mulighet for at verdien til den variable skal huskes til neste gang prosedyra kalles opp. Dette gjøres enkelt og greit ved å skrive **Static** i stedet for Dim.

Eks.:

```
Private Sub Picture1_Click ()
Static Tittel As String ' Deklarerer variable.
Dim I, Bredde As Integer, Start As Single
.......
End Sub
```

De variable tilordnes verdier litt ulikt for tallvariable og strenger (merk "gåseøynene"):

```
Tittel = "Puslespill"
Bredde = 120
```

VB gir også mulighet for å definere tabellvariable eller matriser. Matriser er variable med samme navn, men de skilles fra hverandre med en indeks som f.eks. kan gå fra 0 til 99 .

Eks.: Dim Summer(0 To 99) As Single . Man kan også bruke flerdimensjonale matriser. Grense oppad er 60 dimensjoner! Mer om matriser senere. (Kap 8 i læreboka.)

FORM-VARIABLE.

VB har også mulighet for å definere variable som er tilgjengelig ett nivå høyere enn de lokale, nemlig på form-nivå. Disse variable er tilgjengelig for alle objekter og prosedyrer som hører til på denne formen. Disse variabler defineres med Private-setningen i formens generelle del.Variable på formnivå bevares så lenge applikasjonen kjøres. De bevares også når formen lastes midlertidig ut.

GLOBALE VARIABLE.

Disse er tilgjenelig fra alle former, kodemoduler og prosedyrer i programmet. De globale variable må, som modulvariable, defineres i den generelle del av en form. Man skriver enkelt og greit `Public` framfor dem istedet for Dim. det kan være en god regel å definere alle globale variable i kun en av formene. Globale variable beholder sin verdi så lenge programmet kjører.

KONSTANTER.

Bruk av konstanter er god programmeringsskikk. Dette gjelder også VB, og er på denne måten:

```
Const Moms = 0.24, Versjon = "1.07f"
```

For konstanter gjelder de samme regler om nivåbruk som for variable (men bruk av Static er unødvendig). Globale konstanter må defineres i den generelle del av en form med ordet Public framfor:

```
Public Const Lisens = "Mortensnes Skole"
```

Ellers bør man benytte den såkalt "kontekstsensitive hjelp som finnes i VB. Det betyr at når man f.eks. har skrevet ordet CONST i en prosedyre, og så har glemt hvordan resten gjøres, så

kan man bare trykke på Hjelp-knappen F1, så kommer det hjelp på skjermen om det tema markøren står på.

2. If-setningen

En if-test kan skrives slik:

```
If Tall = 0 Then
......
ElseIf Tall = 2 Then
......
Else
......
End If
```

3. Select Case-setning:

I en del tilfeller blir Select Case- setningen mer oversiktlig:

```
Select Case Tall
Case 0
......
......
Case 1
......
Case 2 To 25
......
Case Else
.....
End Select
```

4. Egendefinerte funksjoner

Som nevt har alle objekter i VB sine forhåndsdefinerte subrutiner. Men i tillegg til disse, kan vi lage våre egne funksjoner og subrutiner. I kapittel 9 i VB-boka benyttes slike funksjoner for å regne om fra favner til meter, og omvendt. Dette gjøres ved å velge "Tools/Add Procedure" fra VB-menyen. Alternativt kan man enkelt og greit bare skrive inn inn et nytt funksjonshode mens man er inne i kode-vinduet:

```
Private Function Favner(M)
Favner = M / 1.883 ' Regner om fra Meter til Favner.
End Function
```

Funksjonen kan f.eks brukes slik:

```
Dim F,M As Integer
M = 1000
F = Favner (M)
```

Verdien M regnet i meter blir regnet om til verdien F i favner

Vi kan lage et annet eksempel ved å summere tall:

```
Private Function SumTall (A,B) [As ????]
....
....
SumTall = A + B
End Function
```

For Funksjoner må vi merke oss at de tilordnes en verdi og denne verdi må tilhøre en datatype. Man kan hoppe over [As????] i funksjonshodet, men da blir funksjonen forhåndsdefinert til å være av datatypen **Variant**.
Merk at funksjoner må kalles med parantes rundt sine parametre:

```
Sum = SumTall (Text1.Top, Text2.Top)
```

Del 3: Objekter, menyer og dialogbokser

Les: Bernt Bertheussen: Visual Basic 6.0 i teori og praksis. EDB-kunnskap 1999. Kap 12-14.
Vi begynner med litt rekapitulering:

1. Programmeringsprosessen i Visual Basic

Denne består som vi har sett av en **visuell** del og en **kodedel**:

- **Tegn vinduet/brukergrensesnittet**
- **Tilpass egenskapene til alle objektene**
- **Skriv kode til handlingene**

VB har innebygget en rekke objekter av ulik type. Det har derfor blitt vanlig å gi de navn som begynner med tre bokstaver som angir hvilken *type* objekt det er.

2. Noen objekter og viktige egenskaper:

Form-objekter:

- **Name:** frmXxxxx
- **Caption:** tekst i tittelfelt

Formene kan også kalles vinduer. På hvert vindu kan vi plassere andre kontroller-objekter.

Tekstboks (eng: textbox):

- **Name:** txtXxxxx
- **Text:** teksten som skal vises i boksen når programmet kjører
- **BackColor:** Bakgrunnsfarge
- **Font …** (tre prikker): en rekke egenskaper som bestemmer skriftutseendet.
- **Multiline:** Kan være True eller False og avgjør om det kan skrives flere linjer med tekst i boksen.

Tekstlapp (eng: label):

- **Name:**lblXxxxx
- **Caption:** teksten som vises.
- **Font … :** Se font for Tekstboks.

Kommandoknapp (eng: Command button):

- **Name:**cmdXxxxx
- **Caption:** Teksten som vises på knappen med eventuelt hurtigtast.

Rullefelt (eng:Scrollbar):

- **Name:** hsbXxxx eller vsbXxxx, som viser om det er snakk om en horisontal eller en vertikal scrollbar.
- **Value:** verdien som tilsvarer posisjonen på glideren i rullefeltet.
- **Max:** Verdien som tilsvarer posisjonene ytterst til høyre eller nederst i kontrollen.
- **Min :** Verdien som tilsvarer posisjonene ytterst til venstre eller øverst i kontrollen.
- **Smallchange:** hvor mye verdien skal endres ved klikk på endepiler
- **LargeChange:** hvor mye verdien skal endres ved klikk i rullefeltet mellom endepilene og glideren.

Picture:

Name:picXxxxx

Du kan plassere så mange bilder du vil i et vindu ved å legge inn flere objekter av typen Picture. Du kan bruke forskjellige filtyper, som GIF, JPG, BMP, ICO, og WMF-filer.

Kontrollen Picture mangler egenskapen *Stretch*, som innebærer at bildet kan skaleres i både høyden og bredden. Hvis du vil strekke et bilde, kan du bruke kontrollen **Image** i stedet.

Egenskapen *AutoSize* finnes i kontrollen Picture. Denne egenskapen sørger for at selve kontrollen retter seg etter bildets størrelse.

3. Menyer:

Menyer brukes for å gi et program mer funksjonalitet, eller for å gi brukeren et nytt nivå av kontroll. Menyene må designes, og de må tilknyttes programkode. Når du lager menyer, jobber du derfor som vanlig først med visuell programmering, og deretter med kodeprogrammering.

Menyens designvindu:

Verktøyknappen for menydesign  . finner du på menylinja (tool bar) og ikke i verktøykassa for objekter.

4. Dialogbokser

En dialogboks er et selvstendig vindu som sender og mottar informasjon mellom brukeren og det vinduet som startet dialogboksen. Vinduet inneholder nødvendige kontroller, og kan som regel beveges over hele dataskjermen, uten å være begrenset til programmet det tilhører. På den annen side er dialogboksen som regel laget slik at den hverken kan skaleres eller styres med systemkontrollene på hver side av tittellinja.

Når brukeren ikke kan komme videre i programmet før hun har lukket vinduet, kalles vinduet modalt.

Typer av dialogbokser

Vi kan skille mellom to hovedtyper av dialogbokser:

- **Felles (*common*) dialogbokser. Kalles *standard* dialogbokser i læreboka**
- **Forhåndsdefinerte (*predefined*) dialogbokser**

A. Felles/standard dialogbokser (kap 14)

Vi får tilgang til felles dialogbokser ved å velge det ikonet som heter CommonDialog fra verktøyboksen og plassere det i vinduet. Kontrollen vises ikke i kjørefasen, men bare i designfasen.

Merk at hvis denne kontrollen mangler i verktøyboksen på din PC, må du velge *Components…* fra *Project*-menyen og krysse av for Microsoft Common Dialog Control 6.0.

VB definerer seks forskjellige dialogbokser. De seks er :

1. **Velge farge** ***(ShowColor)***
2. **Velge font** ***(ShowFont)***
3. **Hjelp** ***(ShowHelp)***
4. **Åpne fil** ***(ShowOpen)***
5. **Skriv ut** ***(ShowPrinter)***
6. **Lagre fil** ***(ShowSave)***

Dette er eksempler på bruk av *metoder* i VB. Har vi f.eks.definert en dialogboks med navn *cdlBoksen* i vårt design, lar det seg gjøre å bruke denne til å endre en farge ved kommandoen *cdlBoksen.ShowColor* . I tillegg må Flags-egenskapen være definert i dialogboksen. Etter dette kan brukerens valgte farge hentes ut via egenskapen *Color* i *cdlBoksen*-objektet. Til dette bruker vi gjerne en variabel:
cdlBoksen.Flags = cdlCCRGBInit
cdlBoksen.ShowColor
Hvilkenfarge = *cdlBoksen.Color*
....
Slik lar det seg gjøre å åpne ulike typer dialoger ved å sette dialogverktøyets metode til en av valgene ovenfor. De ulike dialogboksene kan i VB tilpasses våre behov med de detaljene vi måtte ønske.

B. Forhåndsdefinerte dialogbokser (kap 12)

VB har i hovedsak to forhåndsdefinerte dialogbokser. Den ene, **MsgBox**, brukes for å formidle korte beskjeder, og ta i mot en kortfattet reaksjon fra brukeren. Den andre, **InputBox**, er en enkel redigeringsboks som brukes for å ta i mot skriftlige utsagn fra brukeren, begrenset til en linje med tekst. Disse boksene er ellers stort sett like, og er omtalt i kap. 12 i læreboka.

Disse to boksene finnes ikke i verktøykassa til VB, men de lages av programmet mens det kjøres. I progrmmer er det ofte behov for å bruke meldingsbokser der brukeren kan velge mellom en eller flere svaralternativer. Slike meldingsbokser er ferdig tilgjengelig i Visual Basic som **MsgBox**'er.

Dialogen MsgBox har opptil 5 argumenter:

MsgBox(prompt [, knapper][, tittel][, hjelpefil, context])

Hvis du vil hoppe over enkeltargumenter, må kommaene likevel være med. De siste argumentene har å gjøre med kontekst-følsom hjelp for denne kontrollen.

Hvis du vil bruke mer enn bare det første argumentet, må dialogen alltid stå i et funksjonsuttrykk. MsgBox kan altså brukes som en vanlig kommando, eller som en funksjon som returnerer brukerens respons som verdi av funksjonen. Denne funksjonsverdien kan så i sin tur brukes som utgangspunkt i en if-setning. Meldingsboksene viser sin melding i en dialogboks, og venter på at brukeren skal velge en trykknapp. MsgBox-funksjonen returnerer en verdi som indikerer hva for knapp brukeren har valgt, mens MsgBox statmentet brukt alene ikke gjør dette:

Brukt som funksjon:
Envariabel = MsgBox(prompt [, [knapper][, tittel]])
Merk parantesene rundt de aktuelle parametrene adskilt med komma.

Brukt alene:
MsgBox prompt[, [knapper][, tittel]]

prompt er en tekststreng som vises som melding i dialogboksen.
tittel er også en streng som vises i tittellinja på dialogboksen. Hvis dette utelates, brukes overskriften "Microsoft Visual Basic" som default tittel for applikasjoner når de kjører fra Visual Basic sitt utviklingsmiljø, og applikasjonens navn på den endelige exe-filea som kompileres av Visual Basic.

I tillegg til å bruke MsgBox for å formidle meldinger til og fra brukeren, kan du også bruke denne type dialog til debugging. Debugging er å finne og rette opp feil i programmet. Det kan ofte gjøre det lettere å finne feil, dersom du kan følge med på hva for forgreininger programmet gjør, uten å behøve å stoppe programmet.

Argument nummer to (**knapper**) spesfiserer antall knapper som skal vises, og evt. forklarende ikoner som skal være med, samt hva for knapp som er default m.m.

Følgende tabell viser noen av boksenes navn, verdi og betydning:

navn	verdi	betydning
vbOKOnly	0	OK knapp alene (default)
vbOKCancel	1	OK og Avbryt knapper.
vbAbortRetryIgnore	2	Avbryt, Prøv igjen, og Ignorer knapper
vbYesNoCancel	3	Ja, Nei, og Avbryt knapper.
vbYesNo	4	Ja og Nei knapper.
vbRetryCancel	5	Prøv igjen og Avbryt knapper.
vbCritical	16	Kritisk melding.
vbQuestion	32	Advarsel med spørsmål.
vbExclamation	48	Advarsel.
vbInformation	64	Informasjonsmelding.
vbDefaultButton1	0	Første knapp er default (default).
vbDefaultButton2	256	Andre knapp er default.
vbDefaultButton3	512	Tredje knapp er default.
vbApplicationModal	0	Application modal box (default).
vbSystemModal	4096	System modal message box.

Den første bruppe verdier (0-5) beskriver antallet og type på de knapper som skal vises i dialogboksen; gruppen nummer to (16, 32, 48, 64) beskriver ikontype; den tredje gruppen(0, 256, 512) bestemmer hva for knapp som er default; og den fjerde gruppen (0, 4096) bestemmer modus for meldingsboksen.

Når man ADDERER verdier for å danne en sluttverdi for parameteren **knapper**, så skal det kun tas med ett tall fra hver gruppe. Tips: Søk på Help i VB om tema MsgBox.

I applikasjonsmodus kan MsgBox vise maximum of 1024 tegn. Lengre meldinger brytes av. MsgBox brekker opp linja automatisk ved høyre kant av dialog boksen. His du selv vil velge linjeskift, kan ikke returtasten brukes direkte. Istedet plasses CHR(10) (dvs. ANSI character 10 som lager linjeskift), før første tegn i påfølgende linje. Hvis dialogboksen viser en Avbryt-knapp, vil det å trykke Esc ha samme effekt som å velge Avbryt.

MsgBox brukt som Funksjon

Verdien som **returneres** av MsgBox FUNKSJONEN, indikerer hva for knapp brukeren har valgt, som det framgår av denne tabell:

navn på konstant	verdi	beskrivelse
vbOK	1	OK valgt
vbCancel	2	Cancel valgt
vbAbort	3	Avbryt valgt
vbRetry	4	Prøv igjen valgt
vbIgnore	5	Ignorer valgt
vbYes	6	Ja valgt
vbNo	7	Nei valgt

Merk! Man kan bruke de symbolske konstantene for å slippe å huske alle disse tallverdiene. Se ellers eksempler på dette i VB HELP.

Eksempel: (Hentet fra VB HELP)
Eksemplet bruker MsgBox funssjonen til å vise en kritisk feilmelding i en dialogboks med en Ja knapp og en Nei knapp. Nei-knappen er predefinert som standardsvar. MsgBox-funksjonen returnerer en verdi basert på hva for knapp brukeren har valgt. MsgBox statementer brukes så for å vise en melding som verifiserer hva for knapp som tidligere ble valgt. Eksemplet gjør utstrakt bruk av variable for å presentere en så generell løsning som mulig.

```
Dim Msg, Style, Title, Help, Ctxt, Response, Meld
Msg = "Do you want to continue ?" ' Definerer melding.
Style = vbYesNo + vbCritical + vbDefaultButton2 ' Definerer knapp.
Title = "MsgBox Demonstration" ' Definer tittel.
Help = "DEMO.HLP" ' Definerer Help fil.
Ctxt = 1000 ' Definerer kontekst
Response = MsgBox(Msg, Style, Title, Help, Ctxt) ' Viser melding.
If Response = vbYes Then ' Bruker valgte Ja.
Meld = "Du valgte Ja."
Else Meld = "Du valgte Nei eller trykket Enter."
End If
MsgBox Meld ' Viser det valg som er gjort.
```

5. Øvingsoppgaver

Idéer til oppgavene er hentet fra diverse kurs og lærebøker.

OPPGAVE A: Bondesjakk.
Lag et enkelt spill for bondesjakk som ser slik ut:

Bruk label-objekter som ruter i midten. Ved klikk i hver av disse blir det skrevet ut en "X" i objektet, og ved dobbelklikk blir det skrevet ut en "0". For å få til et pent rutemønster, vil det lønne seg å endre egenskapene **Alignment** og **BorderStyle** slik at utskriften kommer midt i en ramme som legges på hvert av label-objektene.

Ved valg av knappen for Nytt spill, skal alle markeringene i label-objektene fjernes, og Avslutt-knappen skal avslutte programmet.

> Hent løsningen laget av Tom Erik Gundersen ved å klikke på fila Bondesjakk.frm med **høyre** museknapp. Denne løsningen inneholder også if-tester som viser når en av spillerne har fått tre på rad.

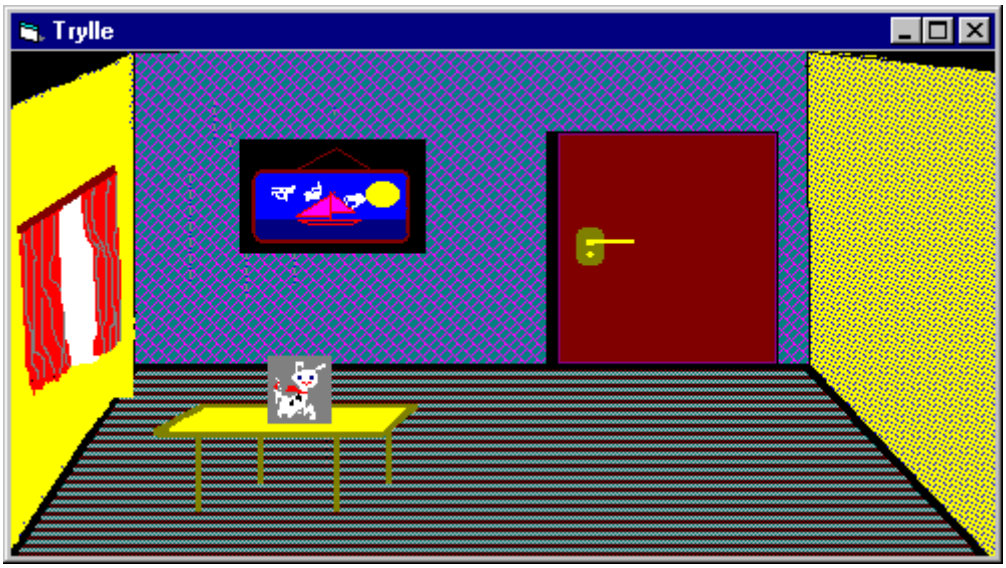
OPPGAVE B: Sum og produkt.
Lag et Visual Basic program som har følgende skjermbilde:

Når du klikker på knappen "Sum" skal det store feltet øverst vise summen av de to tallene i de to mindre feltene. Når du klikker på knappen "Produkt" skal det brede feltet øverst vise produktet av de to tallene i de to mindre feltene. Når du bruker en av de to horisontale scrollbar-kontrollene, skal tallverdiene i tekstfeltene rett ovenfor endre seg. Scrollbarene skal ha tallverdier i området -100 til +100.

Tips: For å gjøre en tekst gjøres til et tall kan du bruke `val(tekst)`. Tilsvarende kan et tall gjøres til en tekst ved å bruke `str(tallverdi)`.

En ferdig løsning finner du ved å klikke på fila Sumprod.frm med **høyre** museknapp.

OPPGAVE C: En liten tryllelek.
Vi tar utgangspunkt i et rom med et bord og en dørramme i veggen. Vi har to bilder av døra som passer i ramma: henholdsvis åpen og lukket dør. Av katta har vi også to bilder: En katt på gulvet og en på bordet.



Vi skal lage et program som muliggjør flytting av katta og åpning/lukking av døra. Hvis vi klikker med musa på den åpne døra, skal bildet erstattes av den lukkede døra og omvendt. Ved å klikke på katta på bordet, skal den hoppe ned på gulvet og motsatt. Men hvis døra står oppe, skal katta forsvinne ...

Kall programmet Trylle. Opprett billedobjektene du har bruk for, f.eks.:
Rom.gif, Lukket.gif, Open.gif, Katt.gif, Gulvkatt.gif (hver av disse bildene kan du hente fra nettet ved å bruke **høyre** museknapp).

Programmer de nødvendige if-testene til hvert bilde og prøv det ut.

Legg så inn bildet Gardin.gif som skal være slik at slik at den (gardina) blåser bort når døra åpnes, men kommer på plass igjen når døra lukkes.

Legg til slutt inn bildet Maleri.gif som skal henge på veggen og som det skal være mulig å flytte rundt til forskjellige posisjoner ved å dra og slippe med musa.

En ferdig løsning kan du hente ved å lagre de to filene: Trylle.frm og Trylle.frx ved å bruke **høyre** museknapp.

Del 4: Bilder og grafikk

Les: Bernt Bertheussen: Visual Basic 6.0 i teori og praksis. EDB-kunnskap 1999. Bonuskap. 5 og 6.

1. Grafiske objekter.

Det er mye spennende man kan få til med grafiske objekter. Visual Basic har fire grafiske objekter som du kan bygge inn i ditt design: **Picture box**, **Image**, **Line** og **Shape**:




I tillegg har programmereren rike muligheter til i sin kode å definere koordinatsystemer, bruke tegningsprimitiver (linjer, rektangler, sirkler, ellipser m.m.), bestemme linjetyper og fyllmønstre, kombinere farger m.m.

2. Picture box, Image: Egenskaper, metoder

Disse objektene har en del egenskaper og metoder felles. **Picture box** objektet er mer fleksibelt, og mer krevende (minne, prosesseringstid). **Image** objektet egner seg best for statisk bruk, mens **picture box** objektet er ideelt for dynamiske situasjoner (f.eks. animasjon).

Egenskaper
Picture box og Image objektet har bl.a. følgende viktige egenskaper:

Name, Enabled, Left, Top, Width, Height og Visible. Som hos alle andre objekter som du lager med verktøykassen er koordinatene til øverste venstre hjørne *(Top, Left)* samt til nederste høyre hjørne *(Width, Height)* relative til formen som eier dem. Øverste venstre hjørne i formens klientområde har koordinater (0, 0).

Følgende felles egenskap kan trenge en nærmere forklaring:

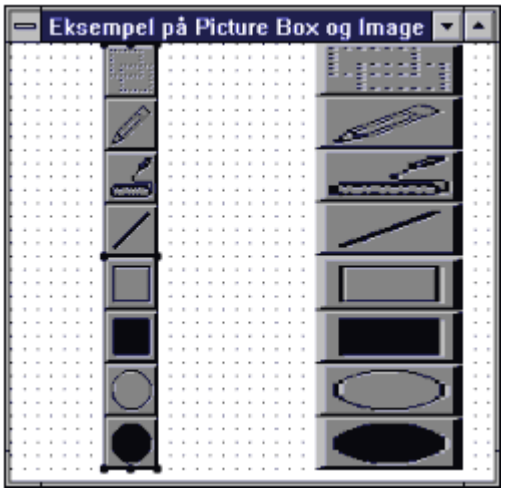
> *Picture:* Ved å klikke på ... i Properties vinduets *Picture* kan du laste inn et konkret bilde objekt eller et ikon når du designer. Du kan også bruke LoadPicture-funksjonen for å laste inn et slikt objekt under kjøring.

Følgende egenskaper gjelder kun for **Picture box** objekter:

> *AutoRedraw:* Hvis True gjennopprettes det grafiske objektet etter at det er blitt forandret (f.eks. ved at et vindu midlertidig har dekket over deler av objektet).

Følgende egenskap gjelder kun for **Image objekter**:

> *Stretch:* Hvis True blir det grafiske innholdet strukket til å fylle Image objektet. Hvis False skjer det motsatte: Image objektet tilpasser seg innholdet.



Til høyre ser du et eksempel med et Picture box objekt (venstre), og et Image objekt (høyre) med Stretch = True.

Hendelser
De viktigste hendelser er Click, DblClick og Paint.

Metoder/prosedyrer
Følgende funksjon er felles for Picture box og Image objekter:

> *LoadPicture:* Laster et bilde eller en ikon fil. Returverdien må tilordnes Picture-egenskapen til objektet: *Picture = LoadPicture(<filnavn>)*

3. Line, Shape: Egenskaper og metoder

Line og **Shape** er egentlig verktøy som du kan bruke for å tegne rette linjer eller diverse geometriske former. I utgangspunktet tegner Shape rektangler, men ved å gi objektets Shape egenskapen en passende verdi får du rektangler, rektangler med avrundede hjørner, sirkler og ovale former.

Egenskaper
Line og Shape objektet har bl.a. følgende viktige egenskaper som du kjenner fra før: *Name* og *Visible*. Shape-objekter har også *Left, Top, Width, Height* egenskaper. Line-objekter har som spesielle egenskaper koordinatene til endpunktene, henholdsvis *X1, Y1* og *X2, Y2*.

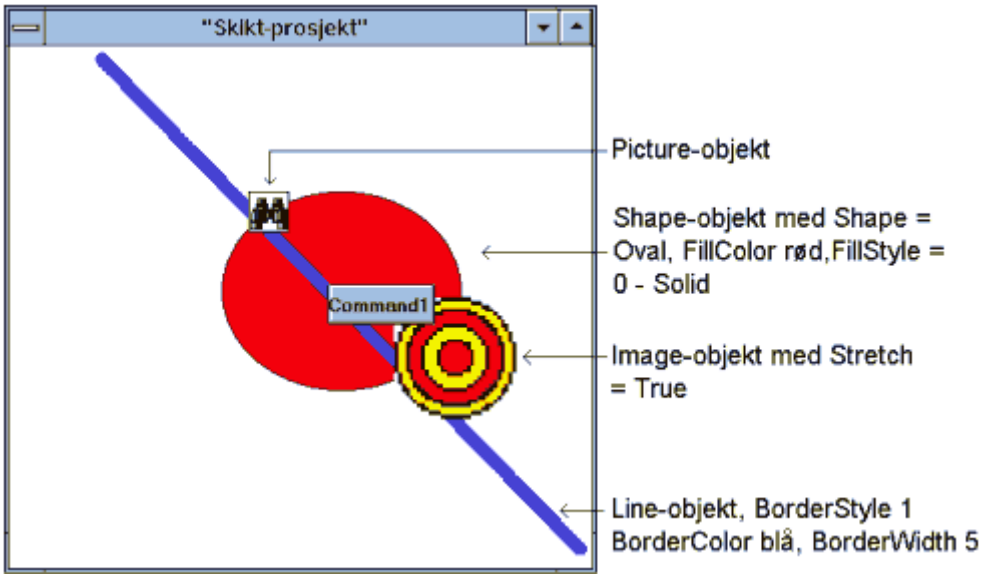
Følgende felles egenskaper kan trenge en nærmere forklaring:

Egenskap	Forklaring
BorderColor	Bestemmer fargen til Shape-objektets ramme eller til linjen (for Line-objektet).
BorderStyle	Bestemmer mønsteret til Shape-objektets ramme eller til linjen (for Line-objektet). De aktuelle verdiene er: 0 – Transparent, 1 – Solid, 2 – Dash, 3 – Dot, 4 – Dash - Dot, 5 – Dash - Dot - Dot, 6 – Inside Solid.
BorderWidth	Bestemmer linjebredden for Shape-objektets ramme eller til linjen (for Line-objektet).

Følgende egenskaper er spesielle for Shape-objekter:

Egenskap	Forklaring
FillColor	Bestemmer fargen til Shape-objektets indre.
FillStyle	Bestemmer mønsteret til Shape-objektets indre. De aktuelle verdiene er: 0 – Solid, 1 – Transparent, 2 – Horizontal Line, 3 – Vertical Line, 4 – Upward Diagonal, 5 – Downward Diagonal, 6 – Cross, 7 – Diagonal Cross.

Eksempel
Figuren under illustrerer plassering av objekter i skikt.



Legg spesielt merke til at objekter som befinner seg i det samme skiktet allikevel vil kunne overlappe hverandre. De objektene som er blitt laget sist vil i tilfelle ligge ovenpå de eldste objektene. Du kan endre dette – den såkalte Z-orden – ved å bruke *Edit.Bring to Front* henholdsvis *Edit.Send to Back*.-private subrutiner.

4. Grunnleggende tegneoperasjoner ("grafiske primitiver")

Visual Basic har "grafiske primitiver" for tegning av piksler, linjer, rektangler, sirkler/ellipser inkl. buer og sektorer. Du kan i tillegg fylle lukkede figurer med farge. Videre finnes funksjoner som skaffer informasjon om grafiske tilstander.

PSet – Piksel-tegning
PSet-metoden fyller et piksel med spesifiserte koordinater med farge (med default farge gitt ved det aktuelle objekts ForeColor egenskap, hvis ingen annen farge spesifiseres med opsjonen color nedenfor). Syntaksen er:

```
[object.]PSet [Step](x, y)[,color]
```

`Step` parameteren brukes dersom x, y skal tolkes relativ til *CurrentX*, *CurrentY*.

Som eksempel for *PSet*-metoden, kan du starte **Help**. Søk med PSet som stikkord, kopier eksemplet på "(general)"-delen i koden til et design med én form. Test eksemplet og så ser du tilfeldig plassering av punkter som vist på figuren under:

Point – Fargeinformasjon
Point-metoden returner fargen til et piksel med spesifiserte koordinater. Syntaksen er:

```
[object.]Point(x,y)
```

Line
a) Tegning av linjer
Line-metoden har følgende syntaks:

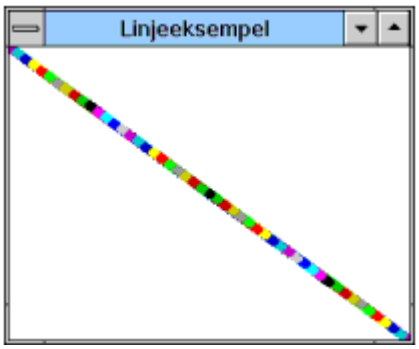
```
[object.]Line [[Step](x1, y1)] - [Step](x2, y2) [color]
```

Det første settet med koordinater, (x1, y1), er startpunktet til linjen. Hvis du sløyfer startpunktet, så brukes pennens aktuelle posisjon (gitt ved *CurrentX*, *CurrentY*).
Det andre settet med koordinater, (x2, y2), er sluttpunktet til linjen. Dette settet kan ikke

sløyfes.
Med color spesifiseres fargen til linjen. Step står for relative koordinater (se Help).

Følgende kode tegner en linje i varierende farger når du klikker på hovedvinduet:

```
Private Sub Form_Click ()
Dim CX, CY, F, F1, F2, I ' Declare var.
ScaleMode = 3 ' Set ScaleMode to pixels
CX = ScaleWidth / 2 ' Get horiz. center
CY = ScaleHeight / 2 ' Get vert. center
DrawWidth = 8 ' Set DrawWidth.
For I = 50 To 0 Step -2
F = I / 50 ' Perform interim
F1 = 1 - F: F2 = 1 + F ' calculations
Forecolor = QBColor(I Mod 15) ' Set
foreground color
Line (CX*F1, CY*F1)-(CX*F2, CY*F2)
Next I
End Sub
```



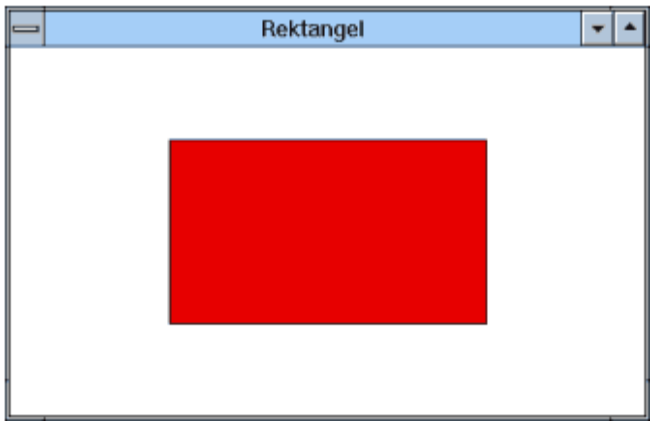
b) Tegning av rektangler
For å tegne rektangler med Line-metoden brukes følgende skrivemåte:

```
[object.]Line [[Step](x1, y1)] - [Step](x2, y2), [color], B[F]
```

Her står (x2, y2) for koordinatene til rektanglets nederste høyre hjørne.
Opsjonen F står for fill. Tar du denne opsjonen med, så blir rektanglet fylt med den fargen som er spesifisert for linjen (default blir ForeColor verdien). Sløyfer du F-parameteren, fylles rektanglet med den fargen som gjelder for FillColor-egenskapen og med det mønsteret som gjelder for FillStyle-egenskapen.

Til høyre ser du et rektangel som er tegnet og fylt med rød farge. Egenskapene er satt til:

```
FillColor = (rød)
FillStyle = 0 - Solid
```



Rektanglet tegnes med kommandoen

```
Line (-50, 50)-(50, -50), ,B
```

Hvis du i stedet bruker:

```
Line (-50, 50)-(50, -50), ,BF
```

så fylles rektangelet med ForeColor verdien.

For andre eksempler med tegning av rektangler, bruk Help.Search, bruk Line som stikkord, kopier eksemplet på “(general)”-delen i koden til et program med én form. Test eksemplene på ulike firkantede mønstre.

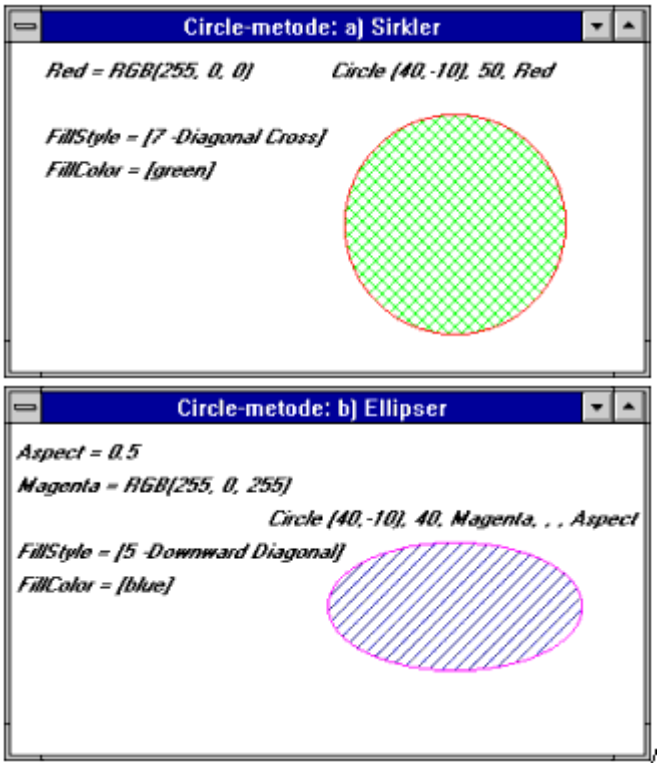
Circle

a) Tegning av sirkler

For å tegne sirkler med Circle-metoden brukes følgende skrivemåte:

```
[object.]Circle [Step](x, y), , radius, [color]
```

der (x,y) er sirkelens senter. Se eksemplet til høyre.



b) Tegning av ellipser

For å tegne ellipser med Circle-metoden brukes følgende skrivemåte:

```
[object.]Circle [Step](x, y), radius, [color], , aspect
```

der aspect bestemmer ellipsens form (mindre enn 1 for brede, større enn 1 for smale ellipser). Se eksemplet.

5. Koordinatssystemer

I utgangspunktet bruker Visual Basic et koordinatssystem basert på ScaleMode = 1 – Twips. Twips står for twentieths of a point, eller 1/1440 tomme. Origo er plassert i klientområdets øverste venstre hjørne, og aksene peker til høyre og nedover, .

ScaleMode

Formens *ScaleMode*-egenskap bestemmer enhetene i koordinatssystemet. Følgende muligheter finnes:

0 – User-defined: Brukeren kan velge enhet og akseretningenes fortegn
1 – Twips: 1/20 typografiske enheter = 1/1440 tomme
2 – Points: 1 typografisk enhet = 1/72 tomme
3 – Pixels: Piksler (hardware-avhengig)
4 – Characters: 1/6 tomme i vertikal og 1/12 tomme horisontal
5 – Inches: Tomme
6 – Millimeters: mm
7 – Centimeters: cm

Scale-metode

Med *Scale-metoden* kan brukeren definere vilkårlige enheter og akseretningenes fortegn.

6. MouseDown, MouseMove, MouseUp

MouseDown, *MouseMove* og *MouseUp* er meldinger som rapporterer at brukeren har trykket ned en musknapp, dratt med musen eller slippet opp en musknapp. De gir deg avanserte

muligheter for å prosessere mushendelser. Den aktuelle syntaksen avhenger av om det er en form eller en kontroller som mottar hendelsen.

For en form eller kontroller har meldingsprosesseringsfunksjonene *MouseDown* og *MouseUp* følgende syntaks:

```
Private Sub Form_MouseDown (Button As Integer, Shift As Integer, X As
Single, Y As Single)
Private Sub Form_MouseUp (Button As Integer, Shift As Integer, X As Single,
Y As Single)
```

Betydningen av de forskjellige parametre er:

1) `Button` er et såkalt biltfield parameter som forteller hva for museknapp som ble brukt. Se VB-Help eller læreboka side 153. Verdi 1 betyr venstre museknapp, verdi 2 høyre museknapp og verdi 3 betyr begge museknapper.

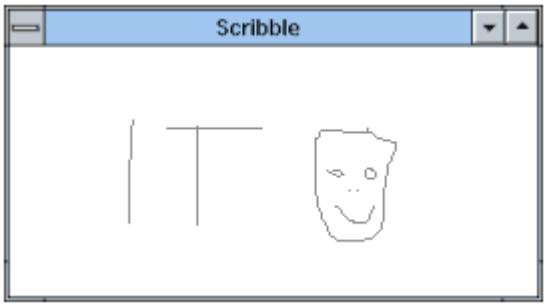
2) `Shift` er enda et biltfield argument som viser til om noen av tastene:
SHIFT, CTRL, ALT ble trykket ned. Se VB-Help eller læreboka side 156. Verdi 1 betyr SHIFT, verdi 2 CTRL, verdi 3 betyr SHIFT + CTRL, verdi 4 betyr ALT osv.

3) `X og Y` er musmarkørens koordinater når hendelsen skjer.

MouseMove har den samme syntaks som MouseDown og MouseUp:

```
Private Sub Form_MouseMove (Button As Integer, Shift As Integer, X As
Single, Y As Single)
Private Sub ctlname_MouseMove ([Index As Integer,]Button As Integer, Shift
As Integer, X As Single, Y As Single)
```

Eksempel



"Scribble" lar deg tegne mens du holder den venstre musknappen ned og drar med musen. Hvis du trykker ned den høyre musknappen fjernes tegningen.
Koden til "Scribble" er slik:

```
' General
Option Explicit
Dim DrawNow As Integer
Const LEFT_BUTTON = 1
Const RIGHT_BUTTON = 2
Private Sub Form_MouseDown (Button As Integer, Shift As Integer, X As
Single, Y As Single)
If Button = LEFT_BUTTON Then
DrawNow = True
CurrentX = X
CurrentY = Y
ElseIf Button = RIGHT_BUTTON Then
Cls
End If
End Sub
```

```
Private Sub Form_MouseUp (Button As Integer, Shift As Integer, X As Single,
Y As Single)
If Button = LEFT_BUTTON Then
DrawNow = False
End If
End Sub

Private Sub Form_MouseMove (Button As Integer, Shift As Integer, X As
Single, Y As Single)
If DrawNow Then
Line -(X, Y)
End If
End Sub
```

7. Mer om farger: RGB-funksjonen

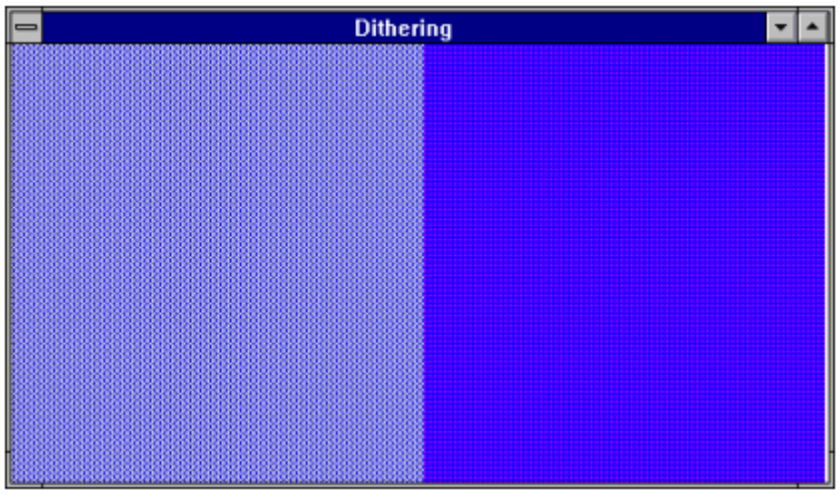
RGB-funksjonen
RGB-funksjonen bygger opp farger ved de tre basisfargene Rød, Grønn og Blå. RGB-funksjonen tar som parameter en RGB-triplett.
RGB-funksjonen gjør om en triplett til en RGB-fargeverdi:

```
rgbColor = RGB(R, G, B);
```

der fargeparameterne R, G, B er egentlig data av type byte d.v.s. Tall i intervallet 0..255.

RGB-tripletter blir ikke alltid framstilt eksakt av skjermkortet. I stedet kan verdien blir tilnærmet framstilt v.h.a. fargeblanding (dithering, se nedenfor).
Du kan oppgi fargeverdien direkte som et heksadesimalt tall. Eksempler:

Farge	**RGB-verdi (&H-format)**	**bRed-verdi**	**bGreen-verdi**	**bBlue-verdi**
Black	&H00	0	0	0
Blue	&HFF000	0	0	255
Green	&HFF00	0	255	0
Cyan	&HFFFF00	0	255	255
Red	&HFF	255	0	0
Magenta	&HFF00FF	255	0	255
Yellow	&HFFF	255	255	0
White	&HFFFFFF	255	255	255



Når Windows ikke kan realisere en spesifisert farge direkte, tilnærmes denne v.h.a. av kombinasjon av piksler i forskjellige farger som plasseres tett til hverandre (dithering). Eksemplet til høyre viser resultatet av å generere to nyanser av blå: RGB(128,128,223) og RGB(64,0,255)

DrawMode: Binære fargeoperasjoner
DrawMode-egenskapen bestemmer hva som skjer når du tegner eller overfører et mønster på en gitt bakgrunn eller mønster. Du kan sette DrawMode til en verdi mellom 1 og 16. De mest brukte verdier er:

DrawMode-verdi	**Navn**	**Forklaring**
1	Black Pen	Tegner med sort penn.
4	Not Copy Pen	Tegner inversen av ønsket farge, uansett hva som måtte ha vært tegnet før.
7	XOR Pen	Kombinerer fargebitsverdiene til tegne- mønsteret med fargeverdiene i det eksisterende mønster v.h.a. binær XOR-operasjon.
11	NOP	= No Operation. Ingenting skjer.
13	Copy Pen	Default verdi. Tegner med ønsket farge, uansett hva som måtte ha vært tegnet før.
16	White Pen	Tegner med hvit penn.

DrawMode-verdi 7 (XOR Pen) er nyttig som animasjonsverktøy (f.eks. tegning med "gummistrikk", flytting av et objekt over bakgrunnen).

Del 5: Løkker og matriser

Les: Bernt Bertheussen: Visual Basic 6.0 i teori og praksis. EDB-kunnskap 1999. Kap 5 og 6.

1. Løkker

Ved siden av valg (if-tester, se del 2) er **løkker** (engelsk loops) et svært viktig begrep i all programmering. Løkker blir brukt for å **gjenta** programkode inntil en betingelse er oppfylt, eller et gitt antall gjennomløp er utført. På mange måter minner dette om klokka i Timer-objektet. I VB kan slike gjentakelser gjøres på flere måter, men i prinsipp er det det samme som skjer, og ofte er det valgfritt hva for løkke-setning som kan brukes. Løkker er omtalt i "Visual Basic boka", kap. 5. Studenter som har liten erfaring med programmering anbefales å gå nøye igjennom dette kapittel. Vi gir her en kort oversikt. Det er to hovedtyper av løkker:

A. For-setningen:

```
Dim Telletall As Integer
For Telletall = 1 To 100
setninger
setninger
....
Next TelleTall
```

For-løkka har et fast antall gjennomløp, som er bestemt når løkka starter. Programbiten som blir utført for hvert gjennomløp, er setningene mellom de reserverte ordene `For` og `Next`. Det er viktig å være klar over at tellevariabelen i løkka automatisk blir oppdatert for hvert gjennomløp. Denne økingen av tellevariabelens verdi treng altså ikke programmeres.

A. While-setningen:

```
Dim Telletall As Integer
Telletall = 1
Do While Telletall < 100
setninger
setninger
.....
Telletall = Telletall + 1
Loop
```

Denne setningen kan også skrives på andre måter, se VB Help:

```
Do
setninger
setninger
.....
Loop While Telletall < 100 And Telletall > 0
```

While-løkka repeterer instruksjonene, så lenge betingelsen er sann. Programmet må selv sørge for at betingelsen blir usann, for å stoppe løkka. Det er ingen automatisk oppdatering av tellevariable i while-løkker, slik som det er i for-løkker. En tellevariabel kan økes inne i løkka ved f.eks å sette `Telletall = Telletall + 1`, for at testen i løkka (Telletall <100) skal bli usann, og dermed stoppe løkka. Dersom denne økingen av variabelen "Telletall" blir uteglemt, vil løkka ikke stoppe, fordi testen i starten av løkka vil alltid være sann.

2. Matriser

Mens en variabel kan sees på som en slags "beholder" for data som er strukturert på en bestemt måte, kan vi si at en matrise er en rad med beholdere som alle er strukturert på **samme** måte. Vi sier da at hver "beholder" inneholder elementer av samme datatype. Matriser kalles gjerne også **tabeller** eller arrayer. En matrise erstatter mange variabler av samme type.

Indeks er betegnelsen på det nummeret som brukes for å nå hvert enkelt element i en matrise. Hvert eneste element kan nås ved hjelp av sin indeks. Vanligvis lager vi en tabell fra 0 til det antall elementer vi trenger, minus 1. Noen ganger er det hensiktsmessig å starte på et annet tall en null. Vi skal se på noen eksempler:

```
Dim Aldersfordeling(0 To 120) As Integer ' matrise med 121 elementer
Dim Temperatur(-50 To 50) As Integer ' Temperaturmålinger i Norge
Dim AntallPåKlassetrinn(8 To 10) As Integer ' matrise med 3 elementer
(ungdomsskole)
```

Den siste matrisen kan vi tenke oss ser slik ut når det er lagt inn data:

Indeks:	8	9	10
Data:	45	41	39

En matrise kan også ha mer enn én dimensjon. Det finnes mange eksempler på nytten av **flerdimensjonale** matriser. Et lite bondesjakkspill er delt inn i tre ganger tre ruter. Skal vi lage et slikt spill, kan vi ha stor nytte av å strukturere spillebrettet slik:

```
Bondesjakk(1 To 3, 1 To 3) As Variant
```

Kapittel 8 av læreboka i Visual Basic 6.0 inneholder ferdige øvingsfiler som du kan laste ned fra nettet.

Del 6: OLE-kontrollen

Les: Bernt Bertheussen: Visual Basic 6.0 i teori og praksis. EDB-kunnskap 1998. Bonuskapittel 4.

Forkortelsen *OLE* står for **O**bject **L**inking or **E**mbedding, og brukes for å muliggjøre samspill og utveksling av data mellom dataprogrammer. Dette betyr at vi kan lage nye programmer der vi benytter data og en del funksjonalitet fra f.eks. Word, Excel, Powerpoint eller et tegneprogram. Ofte er dette en stor lettelse, fordi man slipper å finne opp hjulet på nytt. Kommunikasjonen mellom programmene foregår etter omtrent samme prinsipp som mellom en tjener og en klient i et nettverk. Det programmet vi lager kaller vi gjerne **klientprogrammet**, og det programmet vi "låner" funksjonalitet fra kalles **tjenerprogrammet**. Metoden OLE er i dag en standardisert måte å sende og bruke data mellom ulike Windows-programmer.

1. Tidligere teknikker

Mange dataprogrammer har under Rediger-menyen valget Lim inn Link. Dette brukes for å overføre data fra klienten til tjeneren via utklippstavla. Denne metoden er fortsatt til stor nytte i den daglige bruk av Windows.

DDE var neste utviklingstrinn for utveksling av data mellom programmer. Dette står for *Dynamic Data Exhange*. DDE brukes fortsatt når du vil etablere kontakt og utveksle data mellom to programmer.

2. Ny teknikk: OLE

Den nyeste teknologien er altså OLE. Dette er en teknologi som kan brukes dersom du vil la brukeren av programmet ditt kunne redigere kompliserte data som du ikke kan eller vil lage redigeringsprogram til på egen hånd. Et eksempel kan være å lage et program som bl.a. trenger muligheter for bildebehandling, eller å ha bildebehandling som en liten del av en større applikasjon du setter sammen. Billedbehandlingen blir da en integrert del av programmet ditt og kan brukes innenfor dette.

En viktig forskjell mellom OLE og DDE er at i tilfelle OLE gir vårt program styringen midlertidig fra seg til et annet Windows program. Etter at brukeren har gjort seg ferdig med dette Windows programmet, går styringen tilbake til vårt Visual Basic program.

3. Kobling og innebygging av objekter

Det finnes to måter å håndtere OLE objekter på v.h.a. av OLE kontrollere: Gjennom kobling (Linking) og gjennom innebygging (Embedding).

Ved **kobling** inneholder OLE kontrolleren en referanse til objektet. Hvis du f.eks. lenker et område i et regneark så lagres de aktuelle data til regnearket i en ekstern fil. OLE kontrolleren inneholder kun en referanse (link) til dataene samt en framstilling (evt. et bilde) av dataene. Ved å dobbeltklikke på det koblede objektet startes regneark-programmet automatisk og brukeren kan benytte og redigere de aktuelle cellene.

De data vi jobber med blir dermed et selvstendig dokument som også kan brukes av andre programmer. Ett og samme objektet kan være lenket til flere klientprogrammer. Dersom brukeren redigererer objektet ved å dobbeltklikke på det og deretter gjøre endringer v.h.a. av tjenerprogrammet, så vil alle de klientprogrammer som har lenket dette objektet bli berørt av redigeringen.

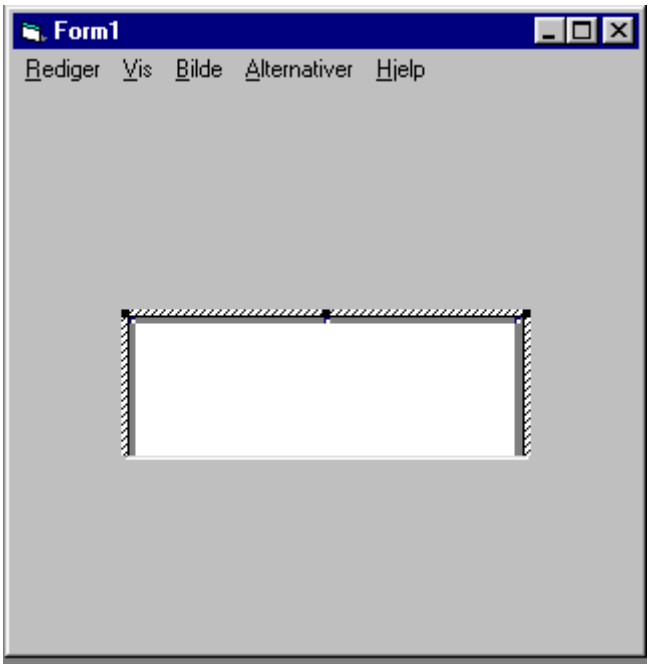
Ved **innebygging** inneholder OLE kontrolleren alle data til objektet. Hvis du f.eks. innebygger et Paint- bilde så inneholder ditt klientprogram all informasjon om objektet inkludert navnet til tjenerprogrammet (Paint). Ingen annen applikasjon har direkte aksess til det innebygde objektet (ikke engang Paint). Du kan redigere objektet, men da kun ved å starte ditt klientprogram og derfra redigere objektet.

4. Eksempel

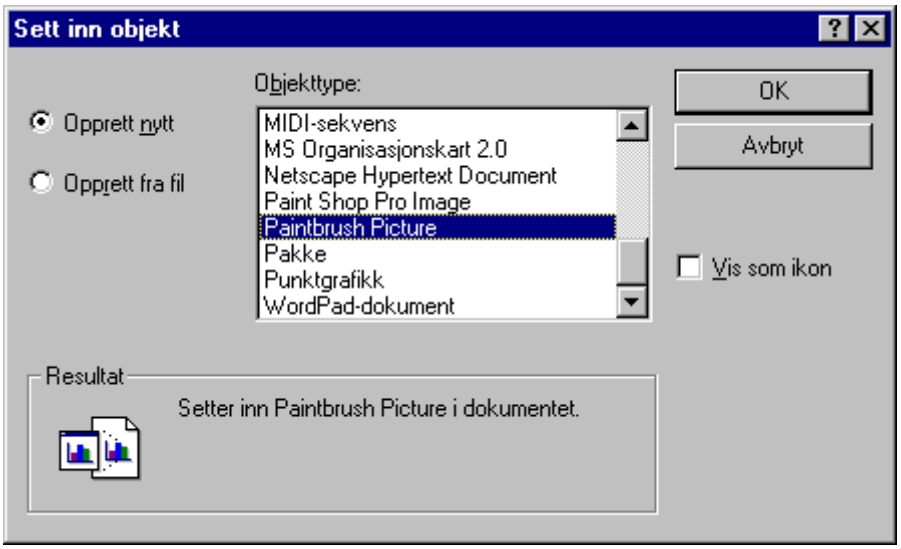
Temaet OLE er egentlig ganske omfattende og med mange muligheter. Vi skal først illustrere emnet med et enkelt eksempel: Muligheten for å bruke redigeringsprogrammet Paint for bitmaps som følger med Windows.

For å velge en OLE-komponent, kan du starte med å dobbeltklikke på dette ikonet fra verktøyboksen:

For å demonstrere en såkalt front-end av enkleste slag er det nok å lage et program som har en eneste OLE-komponent. Figuren nedenfor viser resultatet etter at brukeren har dobbeltklikket på kontrollen OLE:



For å legge inn denne kontrollen er det nok å velge OLE-kontroll fra verktøyboksen, for deretter å velge hvilken type data du ønsker å vise.



Når programmet vårt nå kjøres kan brukeren dobbeltklikke på kontrollen og får da tilgang til tegnemulighetene i programmet for bitmaps fra Windows.

Det er opp til hvert enkelt program å implementere OLE-teknologi, slik at andre front-end-progammer kan benytte det som tjener. Windows sørger for å presentere programmer som har meldt seg med støtte for OLE (tjenere) i ovenstående liste. Det er hvert enkelt programs ansvar å oppføre seg pent og ordentlig som OLE-tjener. Dette er ikke alltid like vellykket. Når du legger innen OLE-kobling lønner det seg å teste tjeneren en stund, før du går god for den.

5. Egenskaper
Når du plasserer en OLE klientkontroller på en form, inneholder kontrolleren ennå ingen data. Det er først når programmet kjøres at objektets data framstilles.

Et OLE objekt har følgende viktige egenskaper:

Egenskap	**Beskrivelse**
Class	Identifiserer tjenerprogrammet med støtte for OLE.
SourceDoc	Bestemmer om objektet kobles eller innebygges.

Class-egenskapen bestemmer tjenerapplikasjonen og datatypen. F.eks. står Excel.Sheet.8 for regnearkobjekter som skapes/redigeres av Excel. Tjenerprogrammet må da være tilgjengelig. For mer informasjon om egenskaper bruk Help.

OLE-objekter lever midlertidig. Når formen som inneholder OLE klientkontrolleren lukkes forsvinner all data som er assosiert med OLE objektet. Data lagres ikke automatisk. For å finne ut mer om lagring, bruk Help

6. Andre eksempler
I lærebokas bonuskapittel 4 gjennomgås hvordan man kan få et Excel regneark til å oppdatere innholdet i en kontroll i et Visual Basic program. For å prøve eksemplet, må du starte Excel, åpne et nytt ark, og legge litt data i de første kolonner. I Visual Basic lager du et program som inneholder en OLEkontroll. Dersom du klarer å vise begge programmene i hvert sitt vindu på likt, vil du se at metoden virker. Du må klikke i Visual Basic programmet for å se at dataene oppdateres fra regnearket.

Et annet eksempel med visning av video i en OLE-kontroll kan du hente ved å lagre de to filene: tv.frm og tv.frx , samt videofila Et.avi med høyre musetast. Dette eksemplet forutsetter at du har et program med OLE-støtte for visning av videofiler av typen AVI på din PC.

7. Databaser (Bonuskap. 2)
Databaseprogrammet Access er et stort og dyrt programm som ikke er inkludert i standardutgaven av Microsoft Office (men kun i Proff-versjonen). Dermed vil mange stå uten tilgang til en database på sin PC. Men databasemotoren i Microsoft Access, *Jet Engin*, er faktisk inkludert i Visual Basic. Denne kan brukes til alt som har med lagring og framhenting av data fra en database av typen access. Bonuskapittel 2 gir en trinnvis innføring i hvordan dette gjøres, og overlates herved til selvstudium.

Del 7: Oppsummerende eksempel og sammendrag.

Visuell programmering bygger på at **hendelser (events)** styrer hva programmet skal gjøre. Hendelser står for all slags input fra mus, tastatur, harddisk m.m. Eksempler er at brukeren klikker på musa eller at det trykkes på en tast på tastaturet. En hendelse registreres alltid av kun ett av programmets objekter. Alle objektene i VB har et forhåndsdefinert sett med hendelser de har mulighet for å registrere.

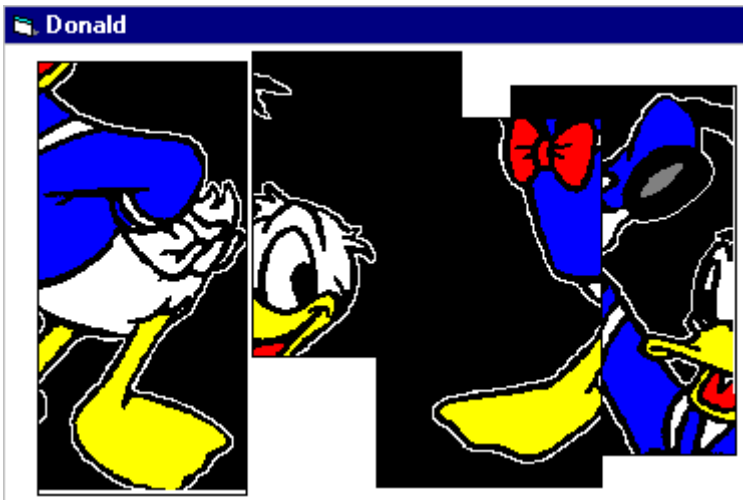
Windowsprogrammet registrerer alle disse hendelsene og omsetter de til de ønskede **meldinger (methods)** i programmet. Dessuten kan de hendelsesprosedyrene vi lager også brukes til å endere **egenskaper (atributter)** til objektene. Dette gjøres ved å referere til objektnavnet og egenskapen med punktum i mellom:

```
picBrikke1.Visible = True
```

Denne koblingen mellom **hendelser** og **meldinger/egenskaper** kalles gjerne programmets DYNAMIKK.

Som illustrasjon skal vi se på et program for å bevege bitene i et **puslespill** rundt på skjermen.

For å få til en skikkelig "dra- og slipp-effekt", må vi legge det hele opp litt annerledes en det som ble gjort i løsningsforslaget til oppgaven om "Tryllerommet" i del 3.



De objekter som vi definerer er:

A) et *formobjekt* med fire variable av typen Single (=desimaltall).

B) fire *billedobjekter* som skal inneholde hver sin pusslespillbrikke. Man lager flere slike brikker etter behov. Disse bildene må ha DragMode property satt til Manuell. De nødvendige puslespillbrikker og billedmateriale av Donald kan du laste ned til din PC med å klikke med **høyre** museknapp på disse filene: d1.gif , d2.gif , d3.gif , d4.gif , Donald.gif .

Koden til dette programmet blir slik:

```
' A). Formobjektet:
'Variable under defineres som generelle i formen,
'og kan dermed brukes av alle prosedyrer som hører til denne form.
'Her er koordinater som viser der musa står i bildet når knappen trykkes
ned:
Dim NedX As Single, NedY As Single

'Her er koordinater som viser der bildet befinner seg på formen:
Dim KantX As Single, KantY As Single

Private Sub Form_DragDrop(Source As Control, X As Single, Y As Single)
Source.Move (X - NedX), (Y - NedY)
End Sub


' B). Billedobjekter:
Private Sub pic1_MouseDown(Button As Integer, Shift As Integer, X As
Single, Y As Single)
pic1.Drag 1 'Melding til systemet om å begynne (dvs 1) en draopperasjon
NedX = X 'Posisjon i bildet der musknapp trykkes ned
NedY = Y '-----------------"----------------
End Sub

Private Sub pic1_DragDrop(Source As Control, X As Single, Y As Single)
KantX = pic1.Left 'Billedkantens posisjon på sitt formobjekt
KantY = pic1.Top '-----------------"----------------
Source.Move (X - NedX + KantX), (Y - NedY + KantY)
End Sub
```

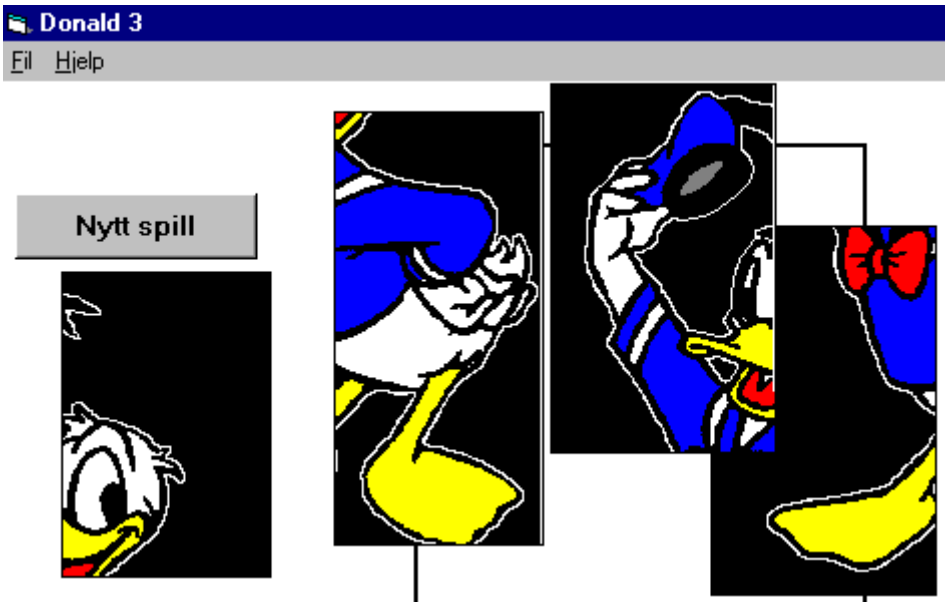
Helt tilsvarende for bilde 2, 3 og 4.
Programkoden for puslespill kan du hente her: pusle1.frm og pusle1.frx .

Vi bør merke oss at Visual Basic lar hvert objekt ha sitt eget koordinatsystem med origo (0,0) oppe i venstre hjørne. Dette må vi ta hensyn til når objekter skal slippes (droppes) ned i ønsket posisjon. Denne posisjon finnes ved å ta hensyn til posisjon når Museknapp trykkes ned (NedX), hvor objektet droppes (X) og evt. kanten på det objekt der vi slipper ned objektet som flyttes (KantX). Det enkleste tilfellet er når objektet droppes ned på bakgrunnsformen. Men muligheten foreligger også for at objektet vi flytter, droppes ned på seg selv, eller på et av de andre bildene. I Visual Basic vil kun et av objektene registrere droppehendelsen. Studer de ulike tilfellene i koden over. Det er litt geometrisk tankearbeid å komme fram til det rette bevegelsesmønster, men det er gøy når det blir ferdig...

I løsningen over har vi gitt hver av brikkene i puslespillet samme kode. Dette er tungvint. En smartere og mer generell løsning er å bruke **kontrollmatrise** (se læreboka). I VB oppretter man kontrollmatriser ved først å lage ett objekt, og så kopiere dette en eller flere ganger på formen. Da spør VB om du vil opprette en kontrollmatrise. Kontrollmatrisen holder rede på de ulike objektene ved en **index** der første eksemplar gjerne har nummer 0.

Programkoden for puslespill med kontrollmatrise kan du hente her: pusle2.frm og pusle2.frx .

Et slikt program er det også fristene å bygge ut med flere egenskaper som automatisk spredning av brikker osv. En slik videreutviklet versjon kan dere hente her: pusle3.frm og pusle3.frx :



Sammendrag av Visual Basic

VB er en form for **objektorientert** programmering der objektene har *egenskaper, hendelser og metoder*. Disse varierer fra objekttype til objekttype. Vi lister opp noen viktige:

A. Egenskaper - Properties.

Name	En kontrolls navn (brukes når kontrollen refereres i koden)
Caption	Teksten som står på kontrollen når den vises på skjermen
Font...	Egenskaper ved teksten som vises i kontrollen
...Color	Farger på forskjellige deler av kontrollen
Alignment	Om teksten skal være venstrejustert, sentrert eller høyrejustert
Borderstyle	Hvilken rammetype (om noen)
Enabled	Om kontrollen skal vær tilgjengelig for brukeren
Height/Left/Top/Width	Angir størrelsen
Index	Hvilket nr denne kontrollen har i en kontrollmatrise
Multiline	Om teksten kan fordeles over flere linjer
Multiselect	Om flere elementer kan velges samtidig
Visible	Om kontrollen skal være synlig for øyeblikket
Value	Verdien som returneres av kontrollen

Viktig skrivemåte: Vi referere til egenskapen på en kontroll slik: `KontrollNavn.Egenskap`
Vi kan også referere til kontroll på *annen* form: `Formnavn!KontrollNavn.Egenskap`

B. Hendelser - Events

_Click	Musa klikket på kontrollen
_DblClick	Musa dobbelklikket på kontrollen
_GotFocus	Kontrollen fikk akkurat fokus
_LostFocus	Kontrollen mistet akkurat fokus
_KeyPress	En tast ble trykket mens kontrollen hadde fokus
_Activate	En form ble akkurat aktivert igjen etter å ha ligget i bakgrunnen
_Paint	Oppdatering av et grafisk objekt
_Load	En form ble akkurat lastet opp
_Unload	En form ble akkurat lukket

C. Metoder - Methods

Show	Vis en form på skjermen
SetFocus	Setter fokus til en bestemt kontroll
Clear	Blanker en liste
AddItem	Legger et element til en kontroll
RemoveItem	Fjerner et element fra en kontroll
Unload	Lukker en form
Drag	Flytter en kontroll på skjermen
Hide	Gjemmer en form

D. Variable

```
Dim i,j As Integer
Dim i gir Variant '(Default)
Static i As Integer 'Verdien beholdes etter prosedyrekall
Public i As Integer 'global variabel
Dim Tabell(13) As Strin g'Matrise. Default startes indekseringen på 0.
(i=0,1,2..)
```

Variabeltyper:
String ($), Integer (%), Long (&), Single (!), Double (#) , Boolean, Variant .
Obligatorisk deklarasjon av alle variable: Legg inn `Option Explicit` i deklarasjonsdelen til en form.

E. Valg

If-test:

`If ... Then Statement` '(NB! Ett Statement, enkleste form)

```
If ... Then
Statements
ElseIf...
Statements
Else
```

```
Statments
End If
```

Case-test:

```
Select Case Variabel
Case 0
Statements
Case 1
Statements
[Case Else
Statements]
End Select
```

F. Løkker. Flere skrivemåter:

`For i=1 to 10`
`Statements`
`Next i`
	`Do`
	`Statements`
	`Loop Until`
		`Do Until n=5`
		`Statements`
		`Loop`
			`Do`
			`Statements`
			`Loop While n<5`
				`Do While n<5`
				`Statements`
				`Loop`

G. Funksjoner

Noen funksjoner er innebygd:

`Val("3001")` Gir 3001. Bla. nyttig fordi VB ikke har mulighet for direkte innlesning av tall
`Str$(123)` Gir tesktstrengen "123"
`Int(5500.12)` Gir heltallsvardien (=5500)
`Rnd` Gir et tilfeldig desimaltall mellom 0 og 1
`IsNumeric(x)` Tester om gyldig tallverdi
`Format$` Gir mange muligheter for presentasjon av tall
`InPutBox` Eks: `StrengVariabel = InPutBox$(Tekst)`
`MsgBox` *kan* være funksjon

Vi kan også definere egne funksjoner (Function) og prosedyrer (Sub):

```
Privat Function f(x,y as Single, t As String) As String
Kall: a=f(x1, y1, t1)
```

Ved problemer med variabeltype, kan typedefinisjon sløyfes i funksjonshodet. Tolkes da som `Variant`.

```
Private Sub ProsedyreNavn(a, b as Integer)
Kall : ProsedyreNavn 1, 2 Ikke nødvendig med parenteser!
```

Parameterne er pr. default var-parametre (verdien kobles både inn og ut av prosedyra). Verdi-parametre (verdier kopieres kun *inn* til prosedyra) angis med `ByVal` i deklarasjonen Eks:
`..ByVal Ny As String...`

Funksjoner og objekter er egentlig teknikker som brukes for å **strukturere** og holde orden på programmet, noe som blir viktigere og viktigere jo større programmet er. Ellers er også Visual Basic en god illustrasjon på at datamaskiner kun kjører regelbaserte programmer. Det er formelt bevist (C.Bøhm og G.Jacopini, 1966), at alle problemer som lar seg løse ved hjelp av datamaskin, kan løses ved bruk av tre grunnbegreper:

-sekvens (Input, Output, Tilordning)
-valg (If..then..else..)
-repetisjon (Do while..)

Dette i kombinasjon med *objektbegrepet*, er grunnfunksjonene i alle strukturerte programmeringsspråk. Programutvikling dreier seg altså dypest sett om en planmessig og fornuftig bruk og kombinasjon av disse tre begreper!

Del 8: Brukerorientert programdesign.

Av Steinar Thorvaldsen, HiTø. 6.3.99. Forelesningsnotat.
(Takk til Lill-Janne Bendiksen som laget deler av dette referatet)

Les i læreboka: Ivar Minken & Børre Stenseth: *Brukerorientert programdesign.* Nasjonalt Læremiddelsenter. 1998, del 1, 2 og 4.

1. Et historisk tilbakeblikk

To epokegjørende oppfinnelser:

A. 1984: Mac'.
Apples datamaskin Macintosh regnes som epokegjørende da den kom med et **grafisk brukergrensesnitt** (GUI = Graphical User Interface) som ligner på nåværende Windows. Oppfinnelsen har røtter tilbake til 60-tallet. Rank Xerox var av de som var aller først ute, men ingen har fått patent på oppfinnelsen. Det er omtrent som med hjulet: en naturlig oppfinnelse som før eller siden måtte komme. Den norskættede Douglas Engelbardt var først ute med musestyring helt tilbake i 1963.

Fra gjennombruddet rundt 1984 og i løpet av en periode på ca. 10 år, ble datamaskinene endret radikalt. Interaksjonen mellom bruker og maskin utviklet seg fra å være et skrivemaskinliknende og tekstbasert system, til de nå så velkjente vindusbaserte menyene og ikonene med pek og klikk.

B. 1994: Veven (WWW)
Veven ble først laget ved Cern i Sveits. **Cern** er et stort forskningssenter for fysikk som bokstavelig talt ligger på grensa mellom Frankrike og Sveits. Ved et av prosjektene på Cern ble det lagde et system som gjorde det mulig for ulike maskintyper (Mac, PC, Unix) å lese og forstå filer fra forskjellige databaser på Internett. Filene kunne produseres og oppdateres med øyeblikkelig virkning. Systemet fikk sitt gjennombrudd i 1994, bl.a. i forbindelse med OL på Lillehammer.

Disse to viktige oppdagelser deler datahistorien grovt inn i tre pedagogiske retninger:

Tre pedagogiske epoker:

1. Skinner-epoken (Før Mac)
Læremaskiner og programmert læring. Programmene var tekstbaserte og sekvensielle. Det ble satset mye på dette innen matematikk. Elevene startet på sitt faglige ståsted og gikk videre ut fra dette, i sitt eget tilpassede tempo. Det ble laget hele læreverk på grunnlag av denne tankegang. Men dette var en pedagogisk katastrofe. Elevene fant smutthull for å komme seg gjennom programmene uten å lære noe, bare bli ferdig med oppgavene. Det sosiale læringsfellesskap ble borte da elevene ble overlatt til seg selv og sin læringsmaskin.

2. Piaget-epoken (Mellom Mac og Veven)
Et mer konkret-operasjonelt stadium: Halvkonkreter, visuell læring. Mer åpen modell (som Windows). Programmene er ikke lenger sekvensielle, men brukeren har større valg og

variasjonsmuligheter. Men, datamaskinen brukt på denne måten har fortsatt et forholdsvis snevert læringspotensiale.

3. Rousseau-epoken (etter veven, navnet en liten tilsnikelse)
Rousseau mente man ikke skulle bruke vanlige lærebøker i skolen. Den eneste tillatte boka var Robinson Cruse. Elevene måtte oppleve naturen når de skulle lære om den.

Gjennom veven har hele verden blitt tilgjengelig via en datamaskin. Elevens rolle kan derfor bli annerledes, selv om eleven fortsatt må holde seg på skolen. Utforskende læring. Elevene blir oppdagere i større grad. Prosjektarbeider. Læreren skal være tilrettelegger.

En av nøkkelbegrepene bak veven er **hyperstrukturer**- en dynamisk eller flerdimensjonal framstilling av tekst, bilde og andre medier. Hypertekstene er ikke sekvensiell, dvs. man kan følge mange alternative ruter gjennom materialet. Omtrent som navigasjon på et veinett. Brukerens behov for intuitiv og assosiativ informasjon imøtekommes på denne måten. Aktuelle metaforer er: Navigasjon, hand, bokmerke, forstørrelsesglass.

Hvordan vil slike multimediasystemer fungere når nyhetens interesse har forsvunnet? Videoen skulle jo også være revolusjonerende da den kom på 80-tallet. Vil det bli det samme med moderne IKT? Forskjellen med klassisk video er klart tilstede ved at i bruk av IKT/CD-rom kan elevene være aktive. Interaktivitet – at elevene selv kan styrer læringen, mens videoen er et sekvensielt og mye mer begrenset medium.

2. Brukergrensesnittet. (Kap 12)

Brukergrensesnittet er syntesen av de ulike måtene datasystemet kommuniserer med brukerne. To viktige komponenter i brukergrensesnittet er:

Dialogen: Den konkrete delen av kommunikasjonen slik den kommer til uttrykk i prinsipper og muligheter for interaksjon.

Begrepsmessig modell: Den delen av brukergrensesnittet som spiller på brukerens assosiasjoner, begreper og forventninger. Programmet kan bruke tekst, bilder, farger, animasjon og lyd for å kommunisere med brukeren, mens brukeren har kun tastatur og mus. Programutvikleren har derfor ansvar for en tilrettelegging slik at brukeren kan overskue programmets grunntrekk.

Det kan være gunstig å dele opp designprosessen for læremidler i *informasjonsdesign* og *grafisk* design.

A. Informasjonsdesign
Common User Access (**CUA)** er navnet på en viktig standard for brukergrensesnitt som ble laget av IBM i 1989. Denne beskrivelsen danner faktisk en felles bakgrunnen for de fleste av dagens opreativsystemer. CUA har to hovedprinsipper:

1. Bygg opp **begrepsmessig modell** hos brukeren. Til dette brukes metaforer, dialog, igjenkjenning og konsistens. Dvs identisk terminologi i menyer, hjelp, etc. Konsistente kommandoer

2. Brukeren skal ha **kontrollen.** Her brukes visuelt og tilgivende brukergrensesnitt, hurtige tilbakemeldinger osv.

Programmene som lagres bør gi rask respons når brukeren har gjort sitt valg. Regel: Først fanger man folk. Det skal gå fort (3-10 sek. maks). Så skal det være rimelig klart hvordan man skal komme seg videre – og hvordan kommer man seg ut av programmet. Hjemmesider med trege oppstartbilder og programmer som lar det gå mange sekunder uten at det kommer tilbalemelding på skjermen, skaper både irritasjon og usikkerhet hos brukeren. Brukeren bør i størst mulig grad ha kontroll med tiden i et dataprogram.

Ofte benytter programmene en form for personifisering, ved at elevene skriver inn navnet sitt, og kan få en personlig henvendelse fra maskinen etterpå. Mange ønsker også å gi programmet en dramatiske dimensjon, slik at bruken av programmet skal inneholde nye elementer fra gang til gang.

B. Grafisk design

Persepsjonspsykologien lærer oss mye om menneskets informasjonsbearbeiding, bla. skillet mellom korttidshukommelsen og langtidshukommelsen. Konsekvenser av dette er regelen om maksimum sju valg for styring av oppmerksomhet. Bevegelse stjeler oppmerksomhet. Bilder som ikoner og illustrasjoner må derfor ha en gjennomtenkt funksjon. Lyd utvider og forsterker kommunikasjonen. Spesielle målgrupper krever her spesiell varsomhet.

Menyer:

- Grafisk design og knapper må ikke ta opp for mye plass. Grafisk design skal være tiltalende, sammenbindende og identitetsskapende.
- Knapper bør ligge nær hverandre slik at det ikke blir behov for store bevegelser med musa.
- Navigasjonsredskapene bør ofte legges på siden for ikke å stjele høyde fra skjermformatet
- Farger og symboler for linking må kun brukes som det, og ikke finnes i ikke-klikkbar sammenheng
- Antall felter i menyen bør være 7 (maks. 10). Dersom flere valg: bruk undermenyer
- Sorter etter brukssekvens deretter alfabetisk
- Bruk alltid samsvarende ord og ikoner

Tekst:

- Blanding av Små og Store bokstavere lettere å lese enn bare STORE.
- Linjeavstanden må være minst like stor som avstanden mellom ordene
- *Kursiv* senker lesehastigheten!
- Lange linjer senker lesehastigheten. 40-50 tegn på ei linje er ofte nok.
- Vær varsom med blå tekster. Mørk blå er standard for lenking. Øyet har problemer med å fokusere på lyse blåfarger. Blå bakgrunnsfarge er derimot gunstig.

Det finnes utrolig mange skrifttyper (fonter). Hver av dem har sitt navn og sitt særpreg, men vi kan uansett dele dem opp i to hovedgrupper: **Serif (antikva)** og **Sans Serif (grotesk)** .

Serif kjennetegnes ved at alle tegn har tynne hårstreker på tegnene og en liten kurve i endene. Dette gir teksten et klassisk og tradisjonelt utseende. Man regner også skrifter med serifer som lettere å lese når det dreier seg om store mengder tekst. De er derfor mye brukt i bøker og det typografene kalles brødtekst (hovedteksten - den som i gamle dager skaffet typografene brødet). Bokstaver med serifer kalles også antikva. Times og Palatino er eksempler på skrifter med serifer: **Æ.**

Sans Serif mangler disse serifene. Sans betyr da også uten. De gir et renere, mer moderne utseende og brukes gjerne i overskrifter, i tekst som skal utheves o.l. Bl.a. er de mye brukt i markedsføring da grotesker er noe øyet stopper opp for. Sans serifer kalles også grotesker.

Hvis vi bruker grotesker i store mengder tekst, krever det større linjeavstand enn serif-fonter for å gi en tilfredsstillende lesbarhet. Helvetica og Geneva er eksempler på skrifter uten serifer.

Farger:

Vår fargereaksjon er så fundamental at når barn blir bedt om å sortere fargede former, begynner de automatisk å dele dem inn etter farge og ikke etter form. Men farger blir ofte misbrukt. Mange designere sørger for at designet fungerer i sort/hvitt før du legger på farger.

- Lys blå bør unngås for tynne linjer/streker
- Unngå bare blånyanser
- Unngå flere mettede farger samtidig (Det norske flagg er et hederlig unntak!)
- Unngå rødt og grønt i utkant av skjerm. Det tar vekk fokus fra skjermene.
- Farget bakgrunn fungerer best med en nøytral (sort/hvit) tekst.
- Bruk farger for å presentere ting slik de fremstår i naturen
- Farger tiltrekker og styrker oppmerksomheten
- Farger skape en spesiell stemning

Men husk:

> "In a room of 15 designers, if two of them agree that's a majority!"
>
> *(Bill Courtis, 1989, Int. Conf. on Software Engineering)*

3. Designmetoden. (Kap 5-11)

Med *pedagogisk programvare* mener vi et program som brukes pedagogisk. Tilretteleggelsen er det avgjørende! Vi har ulike typer pedagogiske programmer:

1. Drill og øving
2. Instruksjon

3. Verktøyprogrammer (word, excel etc.)
4. Informasjonsbaser (mat på data etc.)
5. Simulering
6. Kommunikasjonsprogrammer

I dag snakker vi ikke bare om IT-programmer, men om **IKT-programmer**: Informasjons og Kommunikasjons Teknologi. I begrepet IKT ligger mer enn bare tradisjonelle dataprogrammer. Internett er et medium for IT-basert kommunikasjon mellom mennesker.

Edutainment er en slags "CD-rom pedagogikk" – blanding av underholdning og undervisning.

Programvare kan brukes **nakent**, dvs. uten en helhetlig og pedagogisk sammenheng, eller den kan brukes **kontekstuelt**. Noen programmer har også **multippelbruk**, dvs. kan brukes i andre sammenhenger enn det produsenten har tenkt på.

Designmetoden som framstilles i del 2 av læreboka kalles gjerne også "Torgmodellen". Den skriver seg i store trekk fra Les Green (Canada). Det er en systemeringsteori for programutvikling som har vist seg nyttig i mange sammenhenger.

Navne og idé til modellen hentet Les Green fra fiskebryggen i Bergen. Bryggen har mange boder der en kan finne mye spennende! Elevene skal på samme måte selv kunne bestemme seg for hvilke deler av stoffet de vil øve seg i - og hvor lang tid de vil bruke på stoffet.

Faser:

1. Idéutforming.
Er idéen enkel slik at vi kan forstå den etter max 5 minutter? Detaljene krever selvsagt lenger tid...
Tittel?

2. Målformulering
Betraktingsmåte. Relevans til skolens planer?

3. Metaforen og erfaringsbildet
Tegn noen bilder for å få fram metaforer som en kan jobbe videre med.

4. Scenariobeskrivelser
Hvilke krav til funksjonalitet stilles til systemet?

5. Markedet
Et slags torg med boder der aktiviteten skal foregå.
F.eks. Forandre, finne, bilder, info, valg
For at et program skal være vellykket bør det ha mer en én aktivitet i det.
En aktivitetstabell forteller hva henholdsvis elev, lærer og datamaskin skal gjøre.
Det bør være like spennende som "bryggen i Bergen".

6. Nøkkelvinduet.
Skal alltid være et fast og trygt sted der du kan komme tilbake som ikke blir endret når brukerne "roter seg bort".

7. Bukergrensesnitt
Menyer, grafer, bilder, ikoner etc. Kan gode standarder brukes?

8. Detaljene
Beskrivelse av tilstand, valg og koder.
Program-kommandoer.
Utvikling av prototyp – testing av prototyp – justering av prototyp.

Det kan være greit å ha noen virkelige skeptikere med på laget. Pedagogene må samarbeide!
Utviklingsmiljøet må ha detaljerte kunnskaper om brukergrensesnitt og standarder.
Ikke finn på ting selv dersom det finnes gode standarder!

Eks. Vikingebyen2 – et program som fungerer. Eleven er "Vikingekongen".
Kjøpes på Nasjonalt læremiddelsenter.

Eks. Språkprogram:

- Gå inn i torgbod, butikk
- Ordhjelp
- Velge rolle
- Slutte av
- Ordbok
- Bildeframstilling

4. Objektorientert design (Kap 21).

Et objekt er en lukket enhet med egne, lokale egenskaper og parametre. Det har en veldefinert kommunikasjonsflate mot omgivelsene. Bruk av objekter kan skape problemer i designfasen for et program. Det er et dialektisk forhold mellom designmodellens helhetstenkning og objektets individualitet. Skal vi begynne med helheten eller bitene? Objektorientert tenkning er et supplement til Torgmodellen. Det krever mye trening å arbeide vekselvis med oversikt og med detaljer. Objektorientert design og Torgmodellen utfyller derfor gjensidig hverandres svakheter.

5. Opphavsretten

Kilder: SOFF-rapport 3/95 og 1/93

Hva er lov og hva er ikke lov å legge ut på nettet?

Åndsverksloven er både faglig og økonomisk begrunnet.
Ikke alt er beskyttet. Verket må ha "**verkshøyde**" dvs. en viss "originalitet".

Vernede elementer:

- Litterære verk
- Billedkunst, fotografier og logoer
- Datamaskinprogrammer
- Film og videogrammer
- Multimediaverkers struktur
- Musikkverk
- Samleverk og kataloger

Opphavsmannen:

- Enerett til eksemplarframstilling
- Kan nekte krenkende endringer
- Vernetid 70 år etter død. (Etter dette "faller det i det fri", vanligvis allemannseie)

  Framfor å forvalte opphavsretten til sine verk selv, kan en overdra forvaltningen til f.eks et forlag.Viktige norske forvatningsorganer: Kopinor og Tono (Se Tonos Claraprosjekt: http://www.tono.no/)

Låntaker:

- Kan viderespre et eksemplar som er kjøpt, altså selge det videre.
- Har lov til eksemplarframstilling til eget bruk.
- Sitatretten. Noen forlag setter grensen ved 500 ord.

Publikasjoner vil som regel inneholde vernede uttrykk fra flere rettighetshavere
En regner med at elektroniske medier vil svekke opphavsretten.
Et elektronisk marked vil kreve elektronisk rettighetsadministrasjon.
Det arbeides internasjonalt med å utvikle "elektroniske vannmerker", jfr. frimerker og papirpenger.
Automatisk registrering av bruk kan knyttes til fakturering.
Det store problemet er forholdet til personvernet.

Les mer i artikkelen:

Jon Bing: *MENNESKERS VERK Informasjonshefte utarbeidet for ansatte i skoleverket*. Om åndsverk, opphavsrett og kopiering. http://www.kopinor.no/dokumentbank/menneske.html

Oppgaver til del 1-6

Merk: lenkene fungerer kun i nettversjonen av kompendiet. Oppgavene er nummerert 4-9 selv om de gjelder for del 1-6 av kompendiet.

INNSENDINSOPPGAVE 4
Visual Basic.

Les: Bernt Bertheussen: Visual Basic 6.0 i teori og praksis. EDB-kunnskap 1999. Kap 1-3.

I første omgangen skal du gjøre deg kjent med Visual Basic. Du får derfor ingen "vanlig" Visual Basic oppgave, men jeg ber deg gå gjennom kapittel 1 og 2 i læreboka, og ser på eksemplene der. Spesielt er det viktig at du lærer deg å lagre **prosjekt-fil** og **form-fil.** Disse har etternavn .vbp og .frm . Å programmere i VB er en modningssak, og du vil derfor trenge tid for å gjøre deg kjent med programmeringsmiljøet og denne måten å lage program på.

OPPGAVE

Når du utvikler et Windows program med Visual Basic skjer dette i to trinn. Kall det første å lage en Visual Basic prototyp og det andre å generere et selvstendig Windows program (EXE-fil).

a). Nevn de stegene som må gjennomgås for å lage en Visual Basic prototyp.

b). Skissér hvordan du går videre for å generere et selvstendig Windows program (EXE-fil).

c). Hva menes med form og med kontroller i Visual Basic. Gi (evt. tegn) eksempler på begge.

d). Gi en kort beskrivelse av ditt førsteinntrykk av VB. Hva synes du er vanskelig eller lett? Hva du synes om læreboka? osv..

e). Gensesnitt design (særlig Windows design) bygger på grunnleggende prinsipper om å være *brukersentrert, intuitivt, konsistent og tilgivende*. Forklar hva vi legger i disse betegnelsene.

Alle studenter sender noen linjer om dette til meg på e-post: steinar@hitos.no . Sammarbeid gjerne i grupper på 2-4 og send i så fall felles besvarelse.

P.S. De tre eksemplene vi laget på samlinga i Tromsø kan du hente og lagre her ved å trykke på **høyre** museknapp:

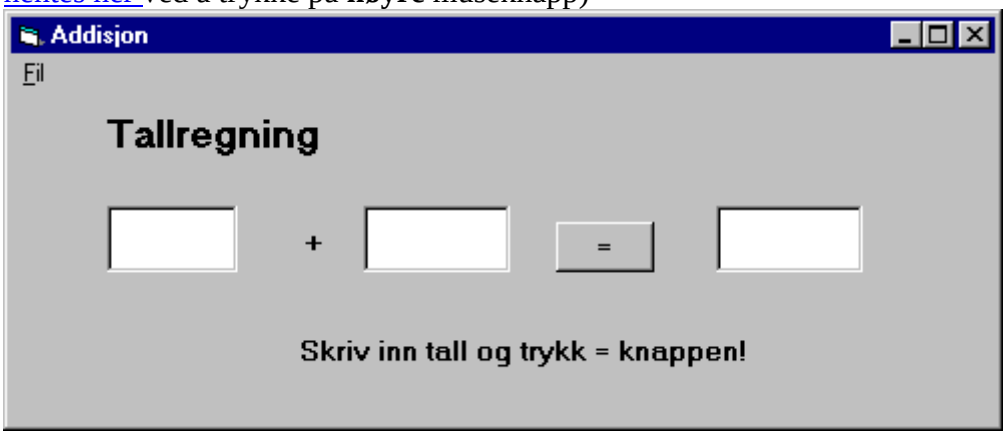
Klokka: klokka.frm
Regenoppgave: addisjon.frm
Sirkelberegning: sirkel.frm

Disse filene kan du lagre på din PC, og hente direkte inn i VB. Når du skal hente inn disse form-filene i VB, så husk å be VB vise alle filer (ikke bare vbp-filer). Filene er lagret i versjon 4.0, men VB konverterer automatisk til versjon 6.0.

INNSENDINSOPPGAVE 5
Visual Basic.

Les: Bernt Bertheussen: Visual Basic 6.0 i teori og praksis. EDB-kunnskap 1999. Kap 4.

På forrige samling laget vi et lite regneprogram som vist under: (Formen addisjon.frm kan hentes her ved å trykke på **høyre** museknapp)



I de hvite feltene til venstre skal brukeren kunne skrive inn tall. Når en trykker på "="-knappen, skal de to tallene summeres, og skrives ut i feltet til høyre for "="-knappen. Den nødvendige kode er plassert på denne knappen.

Utvid dette programmet ved å legge til en ramme (frame-objekt, ikke å forveksle med form) med fire valgmuligheter (option-objekter):

Ved endring/valg i knapperammen skal feltet mellom de to tallfeltene endre tekst til henholdsvis "+", "-", "*" og "/" , samt addere, subtrahere, multiplisere eller dividere de to innskrevne tallene, avhengig av hva for regningsart som er valgt. Vis hvordan programmet nå kan settes opp for å utføre ønsket beregning.

Tips: Bruk **if-tester** knyttet til option-objektene for å finne ut hva for regningsart som skal brukes.

INNSENDINSOPPGAVE 6
Visual Basic.

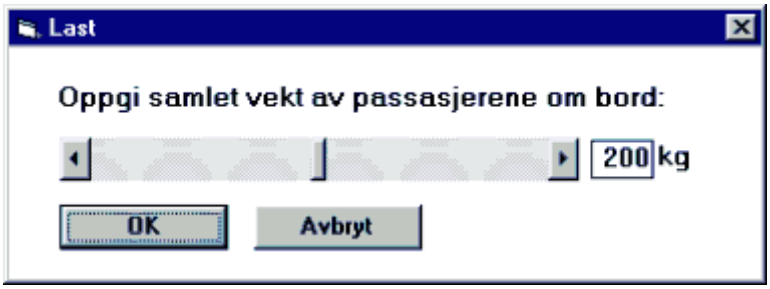
Les: Bernt Bertheussen: Visual Basic 6.0 i teori og praksis. EDB-kunnskap 1999, kap 12 og 9.

EN MUS-STYRT DIALOG.

BESKRIVELSE AV DIALOGBOKSEN

I noen situasjoner kan det være hensiktsmessig at et program er fullstendig mus-styrt. I denne oppgaven skal vi se på hvordan vi kan gjøre en modul i programmet "*Fysikk med varmluftballong*", nemlig modulen for innlesing av masse (vekt av passasjerer), mus-styrt. (Dere kan laste ned en engelsk versjon av selve programmet fra Høgskolen i Tromsø sine hjemmesider på nettadressen: http://www.hitos.no/fou/naturfag/ballong.htm).

Lag en dialogboks (form) for innlesing av masse som vist nedenfor:



Formen skal inneholde følgende kontroller:

Kontroll	**Navn**	**Oppgave**
Label	lbl_Forklaring	Overskrift til rullefelt
ScrollBar	hsb_Masse	Gi valgmulighet for masse (Maks. 400, Min 0)
Label	lbl_Masse	Vise verdien i rullefeltet (j fr eks med rullefelt i kap 9)
Label	lbl_Enhet	Angi enheten (kg)
CommandButton	cmd_OK	Avslutte (bekrefte) valg av masse
CommandButton	cmd_Avbryt	Avbryte valg av masse

DIALOGBOKSENS FUNKSJONALITET

Ved gyldig valg:
Når brukeren klikker på OK-knappen, blir masse-verdien vurdert og den må oppfylle visse krav for at programmet skal gå videre. Dersom verdien er OK, får brukeren først opp en meldingsboks der den totale last er beregnet (masse av passasjerer pluss hylster, brenner etc.) For denne type ballong utgjør massen av hylster, brenner etc 140 kg.

Når *meldingsboksen* lukkes, skal også *dialogboksen* lukkes automatisk.
(Bruk f eks *Unload Me* j fr eksempel i boka s 74))

For å illudere at programmet nå fortsetter, kan du la det komme opp en ny form med f eks et fint bilde av en varmluftballong. Du finner sikkert et passende bilde på internett

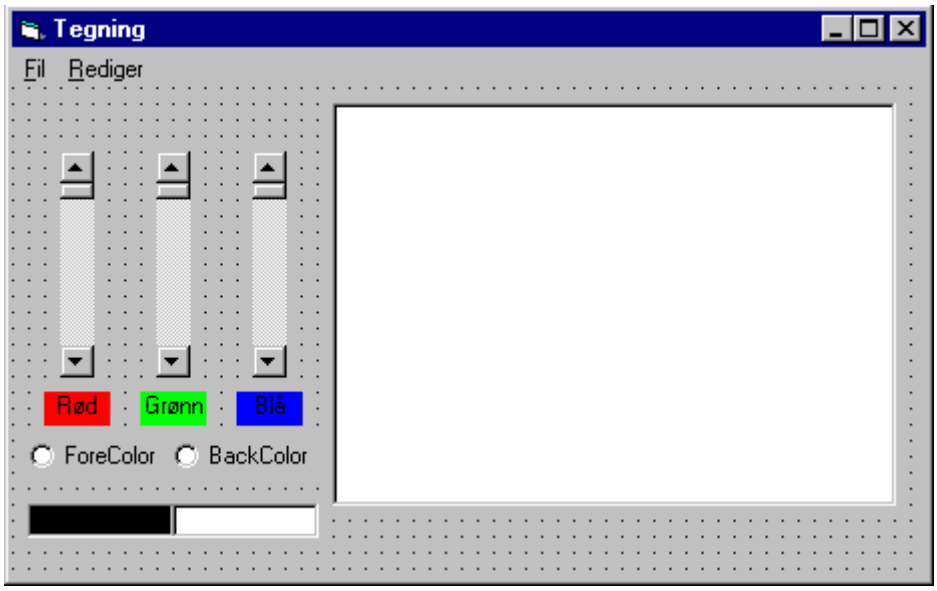
Ved ugyldig valg:
For enkelhets skyld holder vi oss til en ”sommerdag” (bakketemperatur 5 grader C), og da er følgende valg ugyldige:

Verdi-område	**Respons**
m <= 10	Meldingsboks med f eks beskjed om å oppgi gyldig tallverdi
10 < m <50	F eks beskjed om at det er uforsvarlig å sende opp mindreårige alleine
382.2 < m	Beskjed om å studer løftdiagrammet bedre

INNSENDINSOPPGAVE 7
Visual Basic.

Les: Bernt Bertheussen: Visual Basic 6.0 i teori og praksis. EDB-kunnskap 1999; Bonuskap. 6.

Et tegneprogram.
I denne oppgaven skal vi lage et enkelt tegneprogram. I første omgang konsentrerer vi oss om å lage det nødvendigste kontrollerne med tillhørend kode. Senere kan man legge på muligheter for å endre pennfarge og evt. også penntykkelse.



For å gi programmet noe nytt (?) i forhold til andre tegneprogrammer, velger vi at det skal tegne på to ulike måter avhengig av hva for museknapp som trykkes ned:

Venstre museknapp gir vanlige krumme linjer, **Høyre** museknapp gir rette linjer.

Tegneflaten til høyre er et objekt som vi kan kalle `Picture1`. For å få til den ønskede dynamikk, bruker vi tre hendelser:

MouseDown, MouseUp og MouseMove.

I hovedformens generelle deklarasjonsdel legges dette inn:

```
Dim TegnNo As Integer
Const LEFT_BUTTON = 1
Const RIGHT_BUTTON =2
```

Variabelen TegnNo (=TegnNå) brukes til å slå tegneoperasjon på/av. De navngitte konstantene brukes for å slippe å bruke uforståelige tallbetegnelser.

Prosedyra MouseDown har ulik aksjon avhengig av hva for museknapp som ble trykket:

```
Private Sub Picture1_MouseDown (Button As Integer;Sift As Integer, X
As Single, Y As Single)
If Button = LEFT_BUTTON Then
TegnNo = 1
CurrentX = X
CurrentY = Y
ElseIf Button = RIGHT_BUTTON Then
Picture1.Line -(X,Y)
End If
End Sub
```

```
Private Sub Picture1_MouseUp (Button As Integer;Sift As Integer, X As
Single, Y As Single)
If Button = LEFT_BUTTON Then
TegnNo = 0
End If
End Sub
```

```
Private Sub Picture1_MouseMove (Button As Integer;Sift As Integer, X
As Single, Y As Single)
If TegnNo Then
Picture1.Line -(X,Y)
End If
End Sub
```

Test ut at denne koden fungerer.

Så kan man gå videre å legge til nye objekter for justering av farger. Til dette er det vanlig å bruke objekter av typen Scroll Bar med verdier i intervallet 0..255 (se side 197). Med kommandoen `Picture1.Cls` kan man viske ut den gamle tegningen. Se ellers tips som følger:

```
Private Sub VScrollRed_Change()
If OptBakgrunn.Value Then
Picture1.BackColor = RGB(VScrollRed.Value, VScrollGreen.Value,
VScrollBlue.Value)
picBakgrunn.BackColor = RGB(VScrollRed.Value, VScrollGreen.Value,
VScrollBlue.Value)
End If
If OptForgrunn.Value Then
Picture1.ForeColor = RGB(VScrollRed.Value, VScrollGreen.Value,
VScrollBlue.Value)
picForgrunn.BackColor = RGB(VScrollRed.Value, VScrollGreen.Value,
VScrollBlue.Value)
End If
End Sub
```

INNSENDINSOPPGAVE 8
Visual Basic.

Les: Bernt Bertheussen: Visual Basic 6.0 i praksis. EDB-kunnskap 1999. Kap 5, + bonuskap. 5.

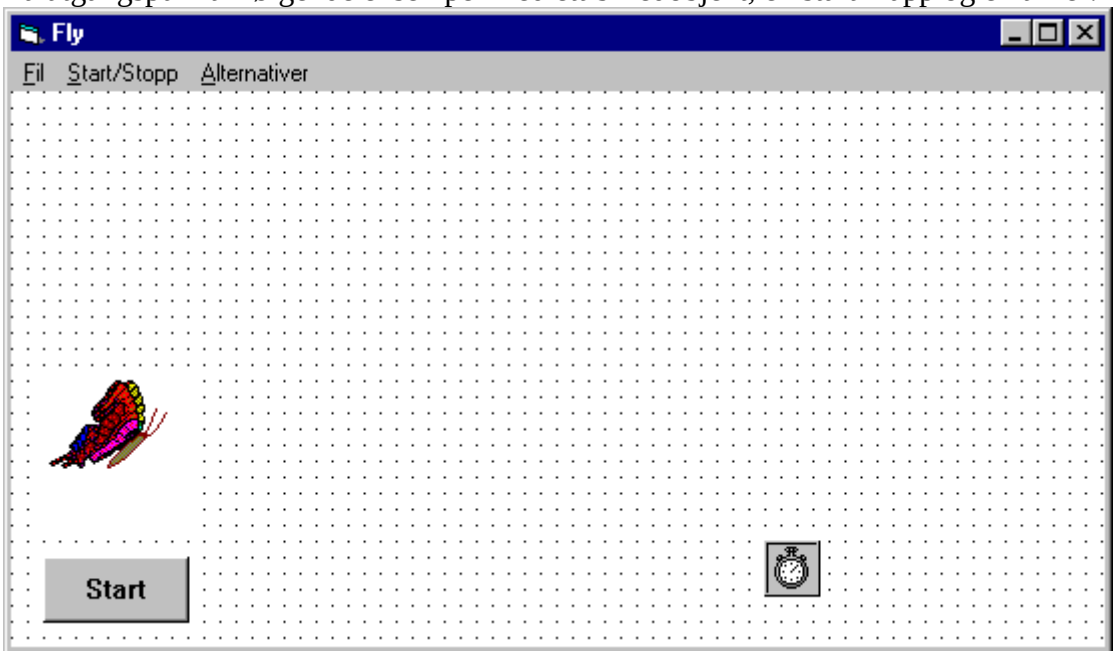
Timer-objektet.

I denne oppgaven skal vi lage et enkelt animasjonsprogram som kan rulle over skjermen. I læreboka finner du i bonuskap. 5 et eksempel med en sommerfugl som flyr over skjermen. Koden til dette finner du i boka eller på løsningsforslagene som er utarbeidet til boka på adressenen: Visual Basic 6.0 .

Til slike animasjoner bruker vi Timer-objektet som du finner i verktøykassa i VB i form av en stoppeklokke. Timer-objektet har en viktig egenskap, og det er at det slås på eller av (Enabeled= True/False). Bevegelsene begynner når kolokka startes, og slutter når den stoppes. Timer-objektet er ellers usynlig under kjøring. Koden for bevegelse av billedobjektet **ImgFugl** langs x-aksen mot høyre kan være slik:

```
Private Sub Timer1_Timer()
ImgFugl.Move ImgFugl.Left + 10
End Sub
```

Ta utgangspunkt i følgende eksempel med ett billedobjekt, en startknapp og en timer:



Filene kan du henter her: fly.frm og fly.frx (den siste inneholder bildet) ved å klikke med **høyre** museknapp.

Studer dette eksemplet. Utvid det med et ekstra billedobjekt som skal bevege seg over skjermen samtidig med sommerfuglen, f.eks. en rullende tekst: **IT-2 er gøy! ,** en gullfisk eller et annet yndlingobjekt.

INNSENDINSOPPGAVE 9
Visual Basic.

Les: Bernt Bertheussen: Visual Basic 6.0 i praksis. EDB-kunnskap 1999; Bonuskap. 4.

Video/lyd program.

I denne oppgaven skal vi lage et enkelt program som kan vise en video- eller lydfil.

På forelesningen om bruk av OLE, ble et eksempel med visning av video i en OLE-kontroll demonstrert. Eksemplet kan hentes ved å lagre de tre filene: tv.frm , tv.frx og Et.avi med høyre musetast. Denne oppgaven forutsetter at du har et program med OLE-støtte for visning av videofiler av typen AVI eller MPG på din PC, og lydfiler av typen WAV, MIDI eller CD. Programmet *Medieavspilling* leveres sammen med Windows 95/98 og har disse egenskapene.

Finn fram til en en videofil eller lydfil på Internett eller på en CD-plate. Lag et program i Visual Basic som spiller av denne filmen eller lyden. Du velger selv hvilke muligheter du ellers vil legge inn i programmet ditt. Vær oppmerksom på at videofiler fort blir veldig store. Ett minutts film med liten komprimering tar ofte 10 Mbyte på hardisken!

Løsningsforslag til oppgaver

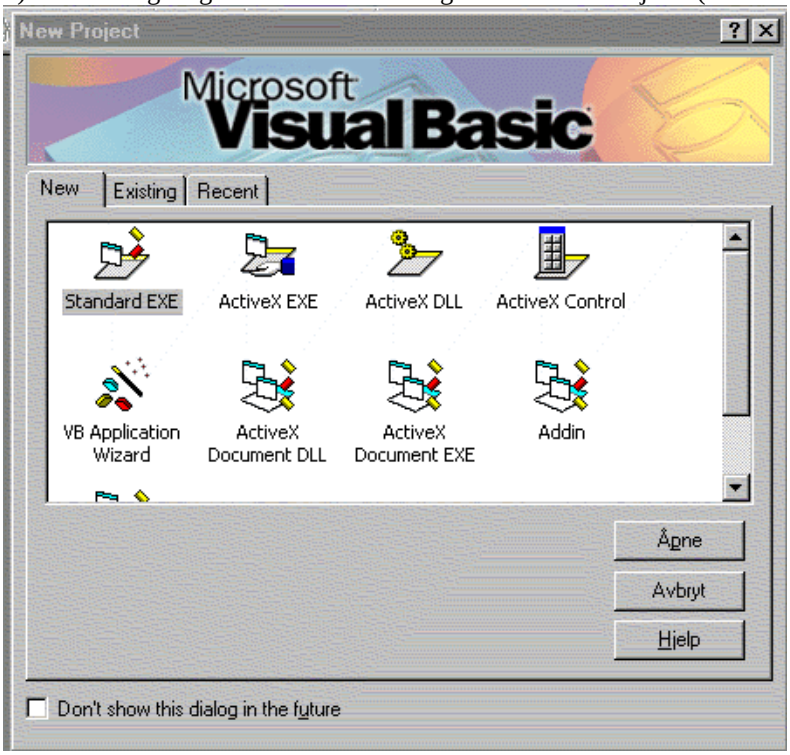
SVAR PÅ INNSENDINSOPPGAVE 4
Visual Basic.

OPPGAVE

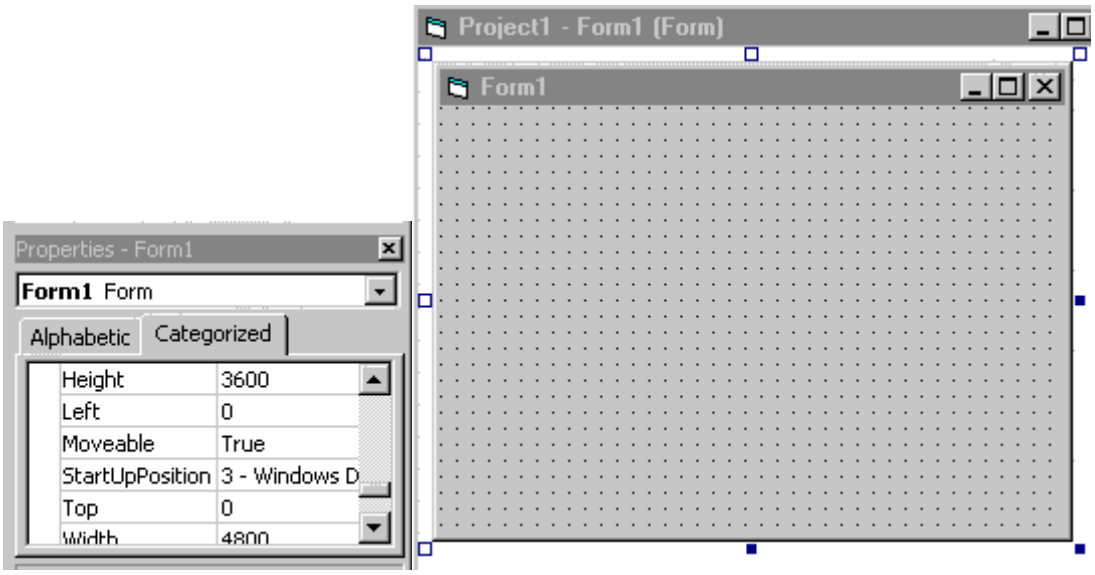
Når du utvikler et Windows program med Visual Basic skjer dette i to trinn. Kall det første å lage en Visual Basic prototyp og det andre å generere et selvstendig Windows program (EXE-fil).

a) Nevn de stegene som må gjennomgås for å lage en Visual Basic prototyp.

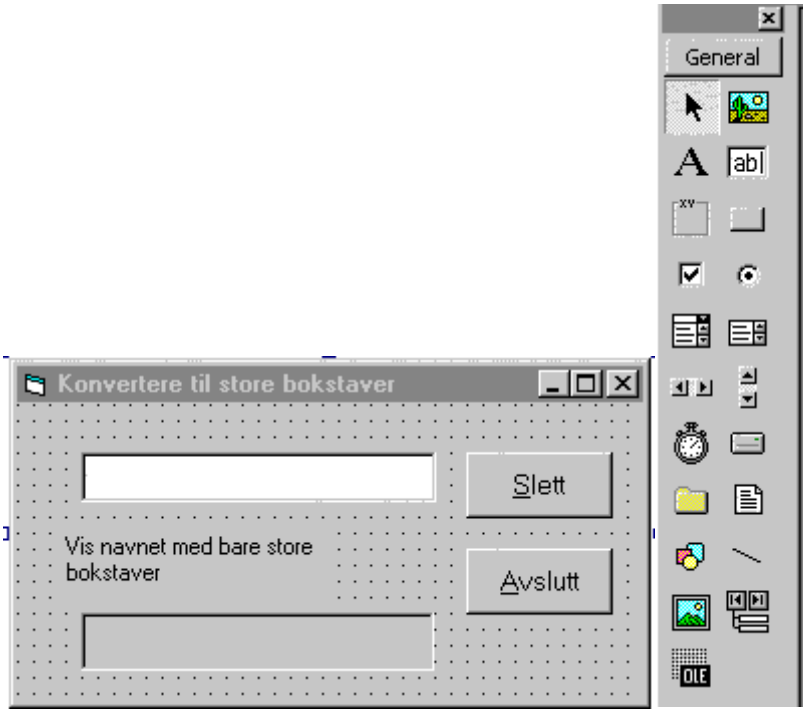
1) Start VB og velg StandardEXE i dialogboksen "New Project" (Det vi arbeider med nå)



Du kan også starte programmet enten ved å velge "Fil" på menyen og deretter "New Project" eller du kan velge "standard EXE" på menyen under fil. Du kan arbeide videre med et prosjekt som er påbegynt ved å velge "Recent" i dialogboksen "New Project"



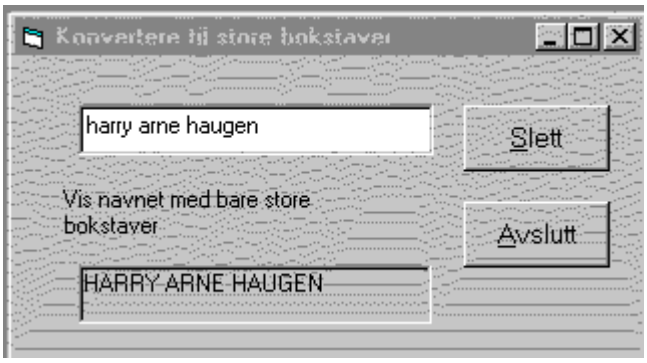
2) Du får opp en "form" som du kan videreutvikle, det vil si sette inn de ønskede "funksjonaliteter" som du vil at det nyutviklede programet skal ha. Du kan for eksempel gi "formen" en eksakt størrelse ved å gå til "properties" og sette inn "Height" og "width".



Du setter også inn "kontrollere" i formen, for eksempel "ettiketter", "tekstbokser", "kommandoknapper" etc. Dette kan du for eksempel gjøre ved å dra "kontrollerne" fra "vektøykassa" og inn på "formen". For eksempel en "textbox", en "ettiketter og to "knapper".

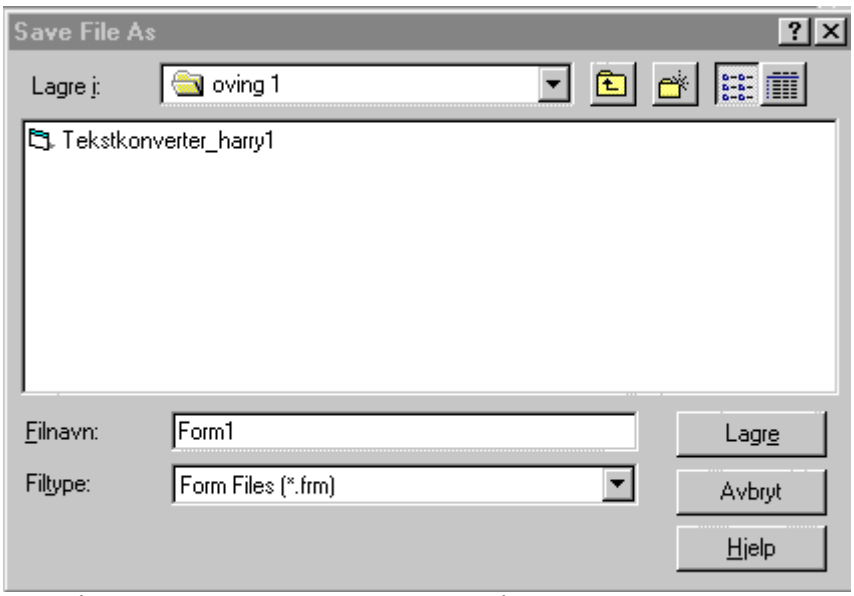
3) Du kan skrive hendelsesprosedyrer som i dette eksemplet der det som skrives i den øverste tekstboksen med små bokstaver automatisk konverteres til store bokstaver i "ettiketten". Eksempel på "hendelsesprosedyre "

```
Option Explicit
Private Sub avsluttknapp_Click()
End 'avslutter programmet
End Sub

Private Sub boks_sletteknapp_Click()
Text1.Text = "" 'tømmer text1.text og text2.text for innho
Text1.SetFocus 'Setter skrivemerket i boksen text1
End Sub

Private Sub Text1_Change()
'UCase bytter til store bokstaver
'caption setter inn innholdet av text1 i text2
Storeboks.Caption = UCase(Text1.Text)
End Sub
```

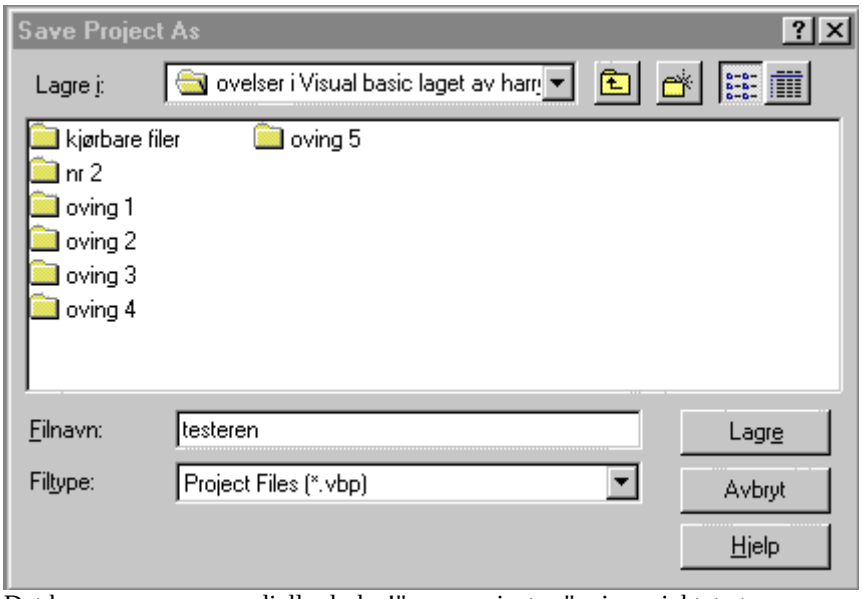


4) Prosjektet må deretter testes. Dette gjøres ved å velge Run-Start, ved å klikke startknappen på verktøylinja eller å trykke F5 på tastaturet.

5) Til slutt må prototypen lagres. I Visual Basic lagres formene og prosjektet hver for seg.



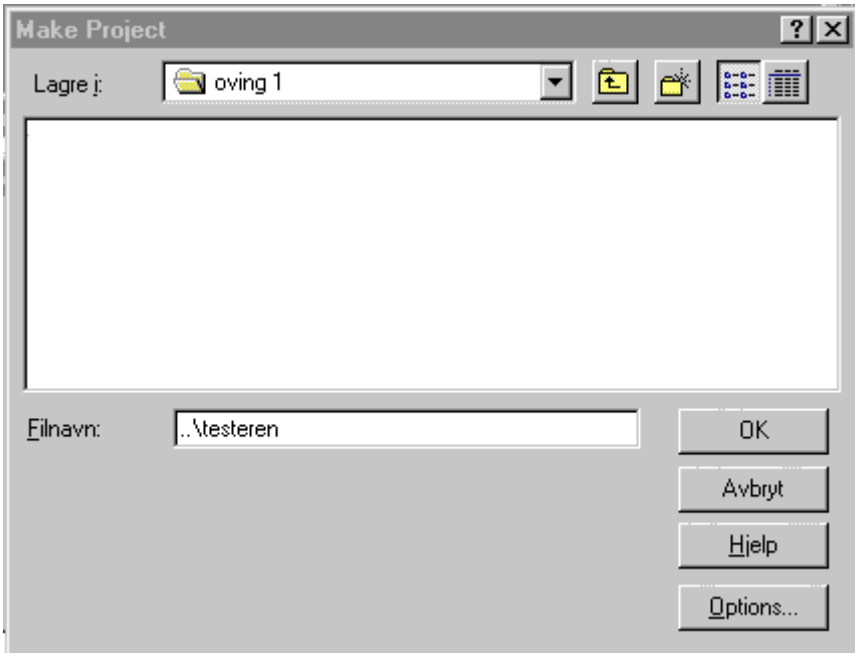
Velg på "fil" menyen "save project as" og du får opp denne dialogboksen. Sett inn navn på "formen" og velg "save"



Det kommer opp en ny diallogboks !"save project as", gi prosjektet et navn og velg lagre.

b). Skissér hvordan du går videre for å generere et selvstendig Windows program (EXE-fil).

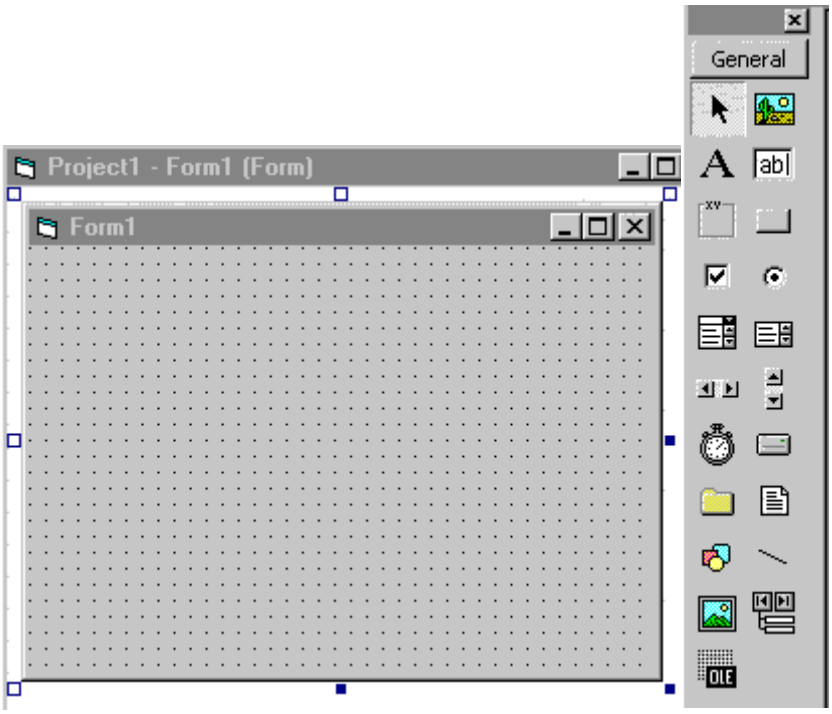
Etter grundig testing kan man lage en exe fil går jeg på "Fil" menyen og velg "make *******.exe" (i mitt eksempel "make testeren.exe)



Og du får opp denne dialogboksen. Gi programmet ditt et navn og lagre det på egnet sted.

Nå har vi laget et kjørbart program.

c). Hva menes med form og med kontroller i Visual Basic. Gi (evt. tegn) eksempler på begge.



En "form" er hjørnesteinen i et hvert program, en form kan brukes til dataregistrering, rapporter eller som en dialogboks. Den størrelsen du gir formen er den samme som vises på skjermen din når du kjører programmet. (En av studnetnene på IT-2 kalte de for "monteringsplata".

En "Kontroller" kan være en ettikett med opplysninger, en tekstboks til å skrive data i, en komandoknapp som utfører en handling av ett eller annet slag. Kontrollerne finner du i "verktøykassa". Både form og kontroller lages visuelt uten kodeskriving. De kan beholdes uten kode hvis en ønsker at de skal være statiske.

d). Gi en kort beskrivelse av ditt førsteinntrykk av VB. Hva synes du er vanskelig eller lett? Hva du synes om læreboka? osv..

Den store fordelen med programering i VB er at du slipper å bruke mye tid på utforming av brukergrensesnittet. En annen fordel er at hendelsesprosedyrene lett kan identifiseres og modifiseres.Foreløpig har jeg ikke funnet noen åpenbare svakheter i forhold til "Basic", eller "pascal" Det grafiske brukergrensesnittet i VB gjør det relativt oversiktilg og enklere å lage gode skjermbilder enn for eksempel programmene som er nevnt forran. Forstår det slik at programmet egentlig består av an mengde små Sub-program.

Har ikke funnet noe som er direkte vanskelig, men det er en stor forbedring i VB5.0 i forhold til VB4.0 det er at du får en liste med kommandoene, denne lista finner du ved for eksempel å dobbelklikke på den kontrolleren du ønsker å gi en funksjon. Dialogboksen "Project form code" åpner seg, ved å sette skrivemerket på ønsket sted og klikke høyre mustast og deretter velge for eksempel "list properties", får en oversikt over kodene.

Læreboka virker forholdsvis bra, det går å bruke den til selvstudium.

e). Gensesnitt design (særlig Windows design) bygger på grunnleggende prinsipper om å være *brukersentrert, intuitivt, konsistent og tilgivende*. Forklar hva vi legger i disse betegnelsene.

Brukersentrert, intuitivt, konsistent og tilgivende betyr:

- Brukeren står sentralt og kontrollerer programmet (ikke motsatt)
- Brukeren skal kunne håndtere skjermbildene mest mulig intuitivt og i samsvar med Windows-standarden.

- Helhetlig og ensartet (konsistent), dvs. lett å finne logikken i programmet.
- Klar logisk og estetisk oppbygging av skjermbilder bygget over samme lest.
- Rask tilbakemelding til brukeren
- Skjermbilder med angremulighet, ikke straffende for evt. feil.

SVAR PÅ INNSENDINSOPPGAVE 5

En ferdig løsning: regning.frm kan hentes her ved å trykke på **høyre** museknapp.

SVAR PÅ INNSENDINSOPPGAVE 6

En ferdig løsning kan hentes ved å trykke på **høyre** museknapp og laste ned de 4 filene: LesMasse.vbp (prosjektfil), LesMasse.frm (formfil), Frm_Bilde.frm (formfil), Frm_Bilde.frx (grafikkfil). Denne løsningen bruker to former. Pass på at alle blir lagret i samme mappe på din PC.

SVAR PÅ INNSENDINSOPPGAVE 7

Oppgaven med å lage et enkelt tegneprogram kan løses på mange måter.

En ferdig standardløsning: tegning.frm kan hentes her ved å trykke på **høyre** museknapp.
En mer proff løsning kan også hentes her: ToreGrTegning.frm .

SVAR PÅ INNSENDINSOPPGAVE 8

En ferdig løsning: fly2.frm og fly2.frx kan nå hentes ved å trykke på **høyre** museknapp.
To andre morsomme løsninger kan dere også se på:

Løsning3: Bengtanimasjon.frm og Bengtanimasjon.frx
Løsning4: flykollisjon.frm og flykollisjon.frx

Jeg sender også med to andre programmer med animasjoner:
En tank som tømmes for vann: Tank.frm og Tank.frx
Et lite spill som tester din hurtighet: muselek.frm og muselek.frx
Studer disse og gjør gjerne forandringer og forbedringer.

SVAR PÅ INNSENDINSOPPGAVE 9
Visual Basic.

En ferdig løsning bestående av de 4 filene: Video.frm , Video.frx , Norge.mid og Cd.avi kan hentes ved å trykke på **høyre** museknapp. Oppgaven viser både avspilling av videofil og lydfil. Du må manuelt tilpasse VB-egenskapenen slik at den finner fram til lyd- og videofiler i den mappen der du laster de med på din PC. I denne løsningen vises hvordan man spiller av OLE-objekter ved direkte valg i menyen ved å bruke `DoVerb` metoden.