NORGES TEKNISK-NATURVITENSKAPELIGE UNIVERSITET
INSTITUTT FOR DATATEKNIKK OG INFORMASJONSVITENSKAP

Faglig kontakt under eksamen:
Dag Svanæs, Tlf: 73 59 18 42

EKSAMEN I FAG
SIF8040 - MMI OG GRAFIKK

Lørdag 16. august 2003
Tid: kl. 0900-1400

Bokmål

Sensuren faller 8. september

Hjelpemiddelkode: **D** Ingen trykte eller håndskrevne hjelpemidler tillatt.
Bestemt enkel kalkulator tillatt.

Oppgave 1. Layout av brukergrensesnittelementer (30%)

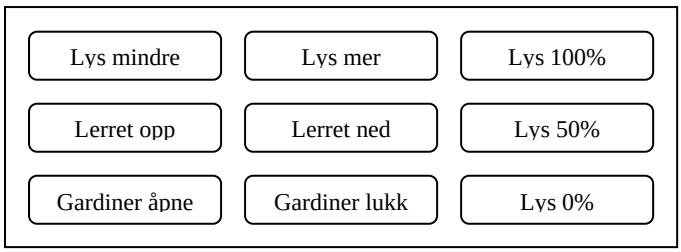
a) Hvorfor er kunnskap om gestaltprinsippene for visuell persepsjon viktig i forhold til utforming av grafiske skjermbilder?

Gestaltprinsippene forteller oss hvordan hjernen struktuerer visuelle inntrykk. Dette er det viktig å forstå når man lager brukergrensesnitt fordi man ellers lett ender opp med å lage et layout som signaliserer en annen sammenheng mellom de grafiske elemenetene enn det man hadde til hensikt.

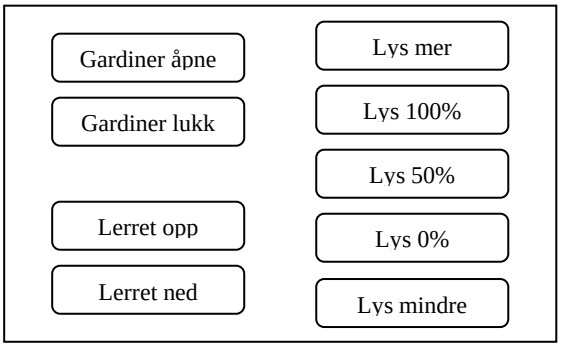
b) Gitt at det skal lages et layout for et brukergrensesnitt med 9 knapper for å styre lys, gardiner og lerret i en forelesningssal. Knappene er:

Lys mer (Øk lysstyrke)
Lys mindre (Minsk lysstyrke)
Lys av (0%)
Lys 50%
Lys 100%
Lerret opp
Lerret ned
Gardiner igjen
Gardiner åpne

Alle knappene skal på samme skjermbilde, d.v.s. det skal ikke være nødvendig å skifte mellom skjermbilder. Det er kommet to forslag til skjermbilder.



Forslag A



Forslag B

Vurder forslagene med utgangspunkt i gestaltprinsippene og Don Norman's begrep "mapping".

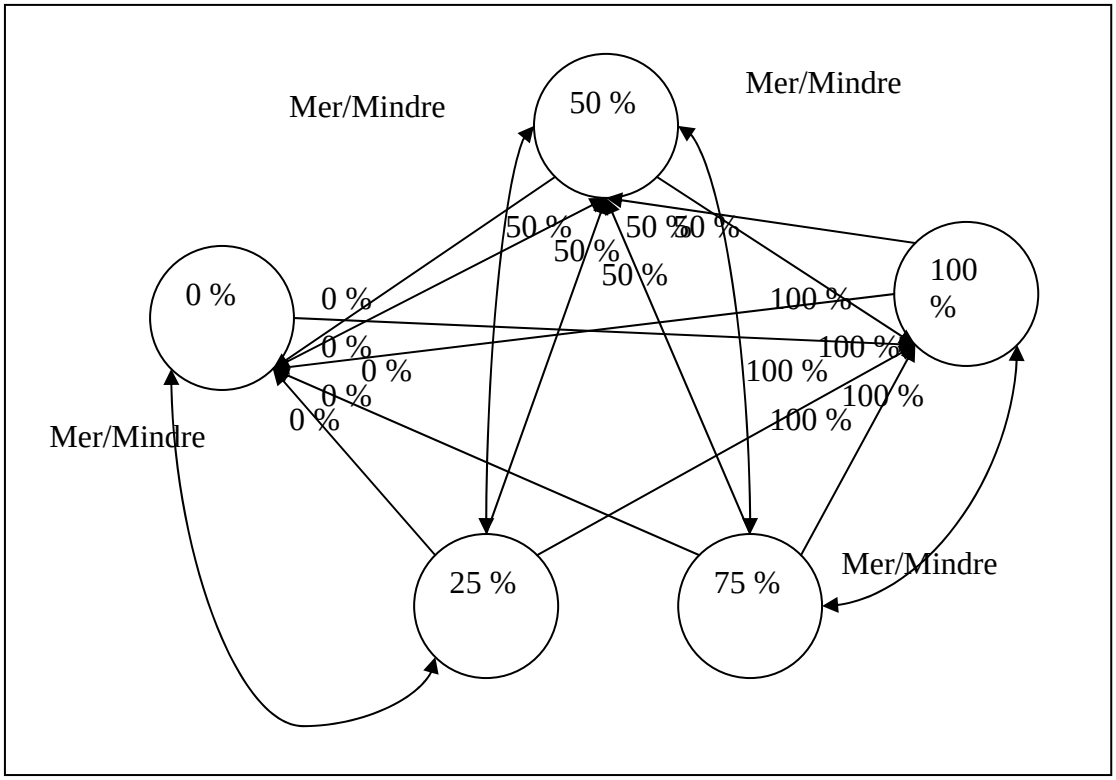
Hvis vi antar at knappene naturlig faller i tre grupper: gardiner, lerret og lys, så får ikke forslag A dette fram visuelt. Det gjør derimot forslag B.

c) Forslagene over er kun forskjellige mht. layout/posisjonering av knappene. Hvilke andre visuelle effekter/gestaltprinsipper kan brukes for å gjøre grensesnittene over lettere å oppfatte?

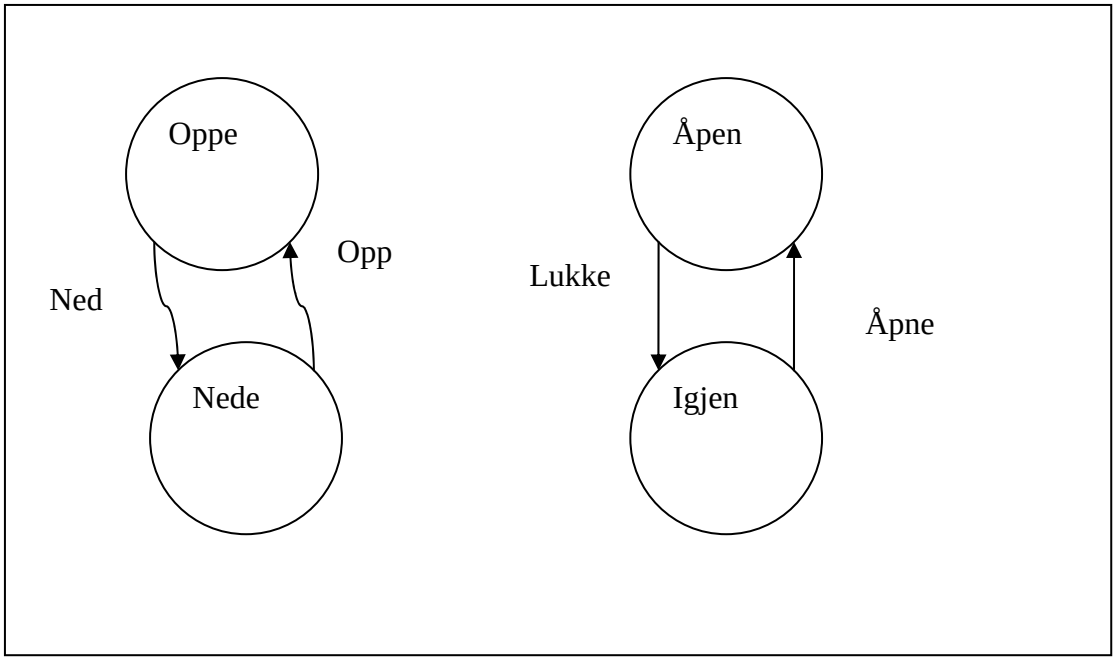
Vi kunne brukt farge, form, innramming, og evt. en grafisk metafor.

Oppgave 2. Tilstandsmaskiner (15%)

a) I oppgave 1 b) ble det beskrevet en kontroll av lysstyrke v.h.a. 5 knapper (mindre, mer, av (0%), 50% og 100%). Gitt at lyset i rommet kun har 5 nivåer (i prosent: 0,25,50,75, og 100). Lag en tilstandsmaskin (bruk gjerne UML Statecharts) som beskriver funksjonen til de 5 knappene i forhold til de 5 lysnivåene.



b) Beskriv hvordan man vha. UML Statecharts kan utvide tilstandsmaskinen i a) med tilstander som beskriver funksjonen til knappene for gardiner og lerret.



Oppgave 3. Evaluering (20%)

a) Forklar forskjellen på heuristisk evaluering (ekspert-evaluering) og brukbarhetstest.

En heuristisk evaluering tar utgangspunkt i generelle retningslinjer og lar en ekspert evaluere grensesnittet i forhold til disse. En empirisk evaluering gjør en test med reelle brukere.

b) Angi fordeler og ulemper ved henholdsvis heuristisk evaluering og brukbarhetstest i forhold til utvikling av brukergrensesnitt.

Fordelen med heuristisk er at man ikke behøver å rekruttere brukere, pluss at man kan få en mer omfattende gjennomgang billig. Ulempen er at man ikke får testet det med reelle brukere slik at man får tilgang til deres spesielle kunnskap.

Oppgave 4. GUI-konstruksjon (35%)

Du skal realisere et GUI for styring av gardiner, lerret og lys, som det beskrevet i oppgave 1 b). Noen andre har allerede implementert tilstandsmaskinen (tilsvarende den i oppgave 2 a) og b), med metoder for å spørre hvilke(n) tilstand(er) den er i, trigge en transisjon og lytte på endringer i tilstand(ene). Din jobb er å bygge GUI'et som skal knyttes til tilstandsmaskinen, både vise tilstanden(e) og styre den. Du regner med å måtte brukbarhetsteste flere forslag til design av GUI'et og ønsker å kunne bytte ut ett designalternativ med et annet, med et trykk på en knapp.

a) Beskriv kort de 3 elementene i MVC (Model-View-Controller)-arkitekturen. Hvilken av disse tre rollene vil tilstandsmaskinen ha? Beskriv hvordan MVC-arkitekturen gjør det enkelt å bytte designalternativ.

Modellen i MVC representerer komponentens bilde av de underliggende dataene som skal vises frem og evt. manipuleres (grensesnittet til datakilden).Modellen vil typisk bestå av klasser for navigering i dataene, operasjoner som kan utføres på dem, og hendelser som beskriver endringer. View-delen håndterer visning/visualisering av dataene, altså output-delen av GUI'et. Controlleren tolker bruker input og oversetter til operasjoner på den underliggende modellen.

Tilstandsmaskinen nevnt i oppgaven vil ha rollen som Modell.

Ideen bak MVC er at de tre elementene har definerte grensesnitt ift. hverandre og hver del kan byttes ut, så lenge grensesnittet håndteres. F.eks. vil det være enkelt å bytte View-delen uten at Modell-delen trenger å endres, eller omvendt: å knytte samme Modell til et annet View.

Grensesnittforslagene i 1b) utnytter kun én knappetype (aksjonsknapp), mens Swing tilbyr flere typer knapper, bl.a. aksjonsknapper (JButton), avkrysnings (JCheckBox)- og radioknapper (JRadioButton).

b) Hvilke typer knapper er det naturlig å bruke for styring av henholdsvis gardiner, lerret og lysstyrke? Det finnes flere alternativer, så begrunn svarene.

Både gardinene og lerretet styres indirekte, dvs. en setter igang en bevegelse i en av to retninger, en endrer ikke tilstanden direkte til fra-trukket/oppe eller for-trukket/nede. Derfor vil det være naturlig med aksjonsknapper for disse, to for hver. Det vil være naturlig å deaktivere den retningen som ikke er relevant. Lysintensiteten kan styres på lignende måte, men i tillegg kan en stille til ønsket intensitet direkte. Siden en kan velge en av flere, er det naturlig å bruke radioknapper i én grupp (funksjonelt lik liste med enkeltvalg eller nedtrekksmeny).

Swing tilbyr også to typer lister ("vanlige" lister og nedtrekkslister) og du lurer på om vanlige lister (JList) fungerer bedre enn knappene for valg av lysstyrke. Du ønsker altså å lage et GUI bygget oppå den samme tilstandsmaskinen som tidligere, med en JList som erstatter for knappene som styrer lysstyrke, og som i tillegg viser nåværende lysstyrke gjennom det valgte listeelementet.

c) Forklar hvordan du vil utnytte MVC-arkitekturen som både JList og GUI'et er bygget på, for å realisere den nye brukergrensesnittutformingen. Beskriv spesielt hvordan listeinnhold og seleksjonen håndteres og bruk UML Collaboration-diagrammer til å illustrere konstruksjonen. Gjør nødvendige antagelser om implementasjonen av tilstandsmaskinen.

Modell-grensesnittet som JList bruker heter ListModel, og enhver datastruktur som skal vises som en liste, må implementere dette grensesnittet. Dette betyr at en må få (den delen av) tilstandsmaskinen (som angir lysintensitet) til å "se ut" som en ListModel, slik at denne plugges direkte inn i JList (som dennes modell). Dette gjøres vanligvis på en av to måter: enten implementeres grensesnittet direkte, eller en lager en adapterklasse (wrapper) som pakker inn tilstandsmaskinen, og som implementerer ListModel-grensesnittet. Den siste måten er å foretrekke, siden den ikke rører tilstandsmaskinen (som jo er noen annens ansvar).

Dette betyr altså at en må lage en klasse, f.eks. StateListModel, som implementerer metoder for å returnere antall lysintensitetsnivåer og hvert intensitetsnivå gitt nummeret i rekken. Dette gir JList nok informasjon til å vise frem listen. Vi antar her at antall nivåer er statisk, slik at en ikke trenger å håndtere hendelser for endring i listeelementene.

En skal også kunne endre lysintensiteten, og dette gjøres ved å lytte til seleksjonshendelser (implementere ListSelectionListener) og trigge tilsvarende tilstandsovergang i tilstandsmaskinen. Merk at en også må lytte på tilstandsendringer i maskinen og oppdatere seleksjonen tilsvarende (og unngå evig løkke), i tilfelle maskinen endrer tilstand på annen måte enn via GUI'et (f.eks. et fysisk panel). Lyttingen på seleksjonsendringer og tilstandsendringer (kun de for lysintensitet er relevante) og endring av henholdsvis tilstand og seleksjon bør implementeres av én klasse, f.eks. LightStateController, for å gjøre det enklere å implementere en konsistent logikk.