1. Uvod
2. Lastnosti uporabniških vmesnikov
3. X11
4. Motif
5. GTK+/GNOME
6. Principi in navodila za načrtovanje grafičnih uporabniških vmesnikov
7. Standardi in testiranje uporabnosti
8. Objektno usmerjeni uporabniški vmesniki
9. Uporabniški vmesniki za internet
10. Uvod v multimedijo
11. Uvod v navidezno resničnost

PRINCIPI
↓
NAVODILA
↓
STANDARDI

Literatura:

- Mandel: *The Elements of User Interface Design*, 1997, John Wiley & Sons, Inc.
- Mione: *CDE and Motif, A practical Primer*, 1998, Prentice hall, Inc.
- Schneiderman: *Designing the User Interface*, Adison Wesley, 1998.
- Galitz: *It is Time to Clean Your Windows*, 1994, John Wiley & Sons, Inc.
- Fowler: *GUI Design Handbook*, 1998, McGraw-Hill, Inc.
- Wright: *GTK+/GNOME Programming*, 2000, Wrox Press.

Uporaba

- Interaktivno risanje v znanosti, tehnologiji, ekonomiji
- Računalniško podprto snovanje in načrtovanje (CAD) mehanskih, električnih in elektronskih naprav
- Nadzor in krmiljenje procesov
- Navidezna resničnost
- Simulacije in animacije pri znanstvenih vizualizacijah
- Gradbeništvo
- Kemija
- Medicina
- Ekonomija
- Umetnost
- Avtomatizirane pisarne in elektronsko založništvo
- Računalniško podprto učenje
- "Industrija" prostega časa

Laboratorij za biomedicinske računalniške sisteme in oslikave

- http://mimi.fri.uni-lj.si
- franc.jager@fri.uni-lj.si
- Oprema
- Tekoči projekti
- Mednarodno sodelovanje
- Diplomske naloge
- Uspehi

1. UVOD

- Kaj je uporabniški vmesnik ?
- Psihologija človeka in računalnikov
- Grafični prikaz informacij
- Zmogljivosti človeka in računalnikov

Kaj je uporabniški vmesnik ?

Vmesnik - (*Interface*)

Vmesnik je mesto kjer se srečujejo in delujejo neodvisni sistemi, ki med seboj lahko komunicirajo.

Kaj je **programski uporabniški vmesnik** ?

Programski uporabniški vmesnik združuje vhodne in izhodne naprave ter programsko opremo, te resurse uporablja in podpira.

Programski uporabniški vmesnik obsega vse kar oblikuje izkušnje uporabnikov z računalnikom vključno z dokumentacijo, uvajanjem in podporo.

- Metafora delovne mize
- "Look and feel"

Psihologija človeka in računalnikov (1)

Računalnik (še ne) misli kot človek, istočasno pa ljudje ne morejo delati stvari, ki jih računalniki dobro obvladajo.

Da je računalnik učinkovit pripomoček človeku, moramo vedeti:
- Kako človek **zaznava**
- Kako človek **procesira informacije** (spomin, razumevanje - poznavanje)

Kognitivna psihologija ali psihologija razumevanja (kako dela razum, misli, pomnenje, učenje) je informacijsko procesni model človekovega razumevanja.

Enotna teorija računanja za raziskave in načrtovanje v psihologiji kot v računalniških znanostih.

1. Zaznavanje

Je postavljanje novih izkušenj v relacijo s starimi izkušnjami in pričakovanji.
Primer 1: *cocktail party phenomenon.*

Primer 2:

Psihologija človeka in računalnikov (2)

2. Procesiranje informacij

1. Senzorni spomin (čutila -> procesorji za zaznavanje -> senzorni spomin -> kratkotrajni spomin)
2. Kratkotrajni spomin (STM) (STM, dolgotrajni spomin -> procesorji za razumevanje -> STM)
3. Dolgotrajni spomin (LTM)

Senzorni spomin:
Sprejema ogromno količino informacij, ki se jih niti ne zavedamo (film, računalniški monitor, animacija). Na zaznavanje vplivajo *spremembe*: svetloba, zvok, gibanje, barve.

Kratkotrajni spomin:
Hrani 7 $\pm$ 2 dejstev (pojmov, besed, števil) približno 30 sekund.
(STM, LTM -> procesorji za razumevanje -> STM)
(Informacija na ekranu, informacija od okna do okna.)

Dolgotrajni spomin:
Informacije so kodirane. Problem dosega informacije. TOT fenomen. 2000 do 3000 besed, sicer 100.000 besed. Vmesnik pomaga človeku s razpoznavanjem.

Zaznavanje in razumavanje -> Mentalni model -> Principi in navodila za gradnjo UV

Grafični prikaz informacij

- Človek sprejema $\approx$ 80% informacij vizualno (močan slikovni spomin)
- Hitrejše razpoznavanje
- Hitrejše učenje
- Hitrejša uporaba orodij
- Minimizacija abstraktnega razmišljanja (ni ukazov s kompliciranosintakso)
- Manj napak
- Takojšnja povratna informacija
- Predvidljiv odziv
- Ni jezikovnih ovir
- Atraktivnost
- Manj stresov

Zmogljivosti človeka in računalnikov

	Prednosti	*Slabosti*
Človek:	Razpoznavanje vzorcev	Majhna kapaciteta STM
	Selektivnost	Hitro pozabljanje v STM
	Učenje	Počasno procesiranje
	Velika kapaciteta LTM	Napake
	Množica ključev za dostop v LTM	Nezanesljiv dostop v LTM
Rač.:	Velika kapaciteta pomnilnika	Omejena kapaciteta
	Zanesljiv dostop do pomnilnika	
	Permanentni spomin	Slabo ujemanje vzorcev
	Hitro procesiranje	Omejene možnosti učenja
	Procesiranje brez napak	Omejena integracija podatkov

2. LASTNOSTI UPORABNIŠKIH VMESNIKOV

- Zgodovina komunikacije človek - računalnik
- Obstoječi grafični sistemi
- Modeli pri načrtovanju uporabniških vmesnikov
- Koncept direktne manipulacije
- Grafični uporabniški vmesniki (GUV)
- Objektno usmerjeni uporabniški vmesniki (OUUV)

Zgodovina komunikacije človek - računalnik (1)

Ivan Sutherland 1962 na **MIT** razvije interaktivne metode za uporabo tipkovnice in svetlobnega peresa, ki so v uporabi še danes (premik, skaliranje, rotacija, ostrenje). Sledijo projekti na MIT, General Motors, Bell Telephone Laboratories.

Grafični zasloni se pojavijo po letu 1975. Eksplozija grafičnih aplikacij (tekstovni urejevalniki, programi za risanje, programi za delo s preglednicami) sledi po letu 1980 (Macintosh, IBM/PC - pravokotna predstavitev polja pikslov z ničlami in enicami - bitmap).

Razvoj grafičnih uporabniških vmesnikov spreminja način uporabe računalnikov. Vmesnik je **interaktiven** (miška, svetlobno pero, palica, grafična tablica).

- Interaktivno delo omogoča ekstremno učinovito komunikacijo med človekom in računalnikom.

- Prikaz informacij na grafični način je najbolj primerna in razumljiva metoda obravnave informacij.

Zgodovina komunikacije človek - računalnik (2)

(• Geste)

(• Govorjena beseda)

(• Pisava)

(• Pisalni stroj)

- Računalnik (ukazni jezik, vprašanje/odgovor, menuji, funkcijske tipke, polja za vnos)
- 1980, Xerox STAR (Xerox, Palo Alto Research Center), uporabljen koncept kazanja in izbire (miška), sistem ni prodrl. (Uporabljena metafora namiznega računalnika; konceptualni model komunikacije uporabnika z računalnikom je bil razvit pred strojno in programsko opremo; miška na kovinsko kroglico, Ron Rider, 1974.)
- 1984, Apple (Macintosh), rojstvo koncepta uporabniškega vmesnika, menuji, izvlečni menu.

⇒ Koncepti:

- Akcija → Objekt
- Objekt → Akcija

Obstoječi grafični sistemi

- **Macintosh** (1984, Apple, prvi široko sprejet grafični sistem, uporabniško usmerjen, direktna manipulacije)
- **Microsoft Windows** (1985, Microsoft, Windows 1.0, grafično usmerjena alternativa MS-DOS in PC, uporabniško usmerjen)
- **DECwindows** (1987, DEC, VMS in UNIX, X okenski sistem - koncept strežnika in odjemalca)
- **OS/2 Presentation Manager** (1987, Microsoft in IBM, grafično usmerjena alternatriva MS-DOS in PC)
- **NeXTStep** (1988, NeXT, uporabniško usmerjen, prva simulacija 3-D zaslona)
- **OPEN LOOK** (1989, AT&T in Sun Microsystems, razvit za UNIX Sytem V.4, bolj preimeren za eksperte, mnogo inovativnih lastnosti)
- **OSF/Motif** (1989, DEC in Hewlett-Packard za OSF (Open Software Foundation), UNIX, temelji na OS/2 Presentation Manager-ju, simulacija 3-D zaslona)
- **Microsoft Windows** (1989, Microsoft, Windows 3.0)
- **OS/2 Workplace Shell** (1992, IBM, grafični uporabniški vmesnik za OS/2)
- **Microsoft Windows** (1992, Microsoft, Windows 3.1)
- **Windows 95, 98, OS/2 Warp**
- **Windows 2000, xp, AIX UNIX, LINUX**

Modeli pri načrtovanju uporabniških vmesnikov (1)

Ni možno zgraditi enega samega optimalnega uporabniškega vmesnika za vse zahteve uporabnika.

Ni možno zgraditi enega samega optimalnega uporabniškega vmesnika za vse uporabnike.

Modeli:

1. Model uporabnika (konceptualni, mentalni)
2. Model programerja
3. Model načrtovalca (dizajnerja)

Uporabnik	**Načrtovalec**	**Programer**
- Izkušnje realnega sveta	- Upor. koncept. model	- Platforma
- Naloge	- Programerjev model	- Operacijski sitem
- Procesi	- Principi načrtovanja uporabniških vmesnikov in navodila	- Lupina
- Orodja		- Razvojna orodja
- Rezultati		- Navodila

Model uporabnika:
Je predstavitev tega kako uporabniki razumejo in komunicirajo s sistemom. Mentalni modeli obsegajo interakcijo med uporabniki in računalniki in tako formirajo temelj za principe in navodila pri načrtovanju vmesnikov.

Metafore - prenašanje lastnosti objektov ali akcij realnega sveta na druge objekte z namenom sugeriranja podobnosti ali analogije med njimi.

Modeli pri načrtovanju uporabniških vmesnikov (2)

Odzivi uporabnika na slabo načrtan sistem:

- Zbeganost
- Panika
- Dolgčas
- Frustracija
- Neizkoriščenost sistema

Modeli pri načrtovanju uporabniških vmesnikov (3)

Model programerja:
Programerjevi komentarji na programsko opremo in uporabnike:

- Prepozno, da bi to spreminjali. Preveč časa smo porabili.
- To dela, uporabniki ne znajo uporabljati sistema.
- Uporabnik mora znati vsaj osnovne stvari.
- To ni napaka, to je tisto kaj naj bi vmesnik delal.
- Tako se nismo dogovorili.
- Če to spremenimo, bo treba delati vse od začetka.
- To bo pa v priročniku napisano.
- To bomo rešili v naslednji verziji vmesnika.

Model načrtovalca:
Načrtovalec sprejema želje, ideje in potrebe uporabnikov, (WYKIWYL), doda znanje in materiale, ki so na razpolago programerju in načrtuje programski produkt, ki ga je možno zgraditi in ga uporabnik lahko uporablja.

Koncept direktne manipulacije (1)

- Sistem je predstavljen kot podaljšek realnega sveta
- Zvezna vidljivost objektov in akcij (WYSIWYG - predstavitev realnosti, ki jo je lahko spreminjati; množica alternativ, ki jih uporabnik lahko doseže direktno ali indirektno; WYSINWYGBAIRVTWYGE)
- Uporablja "prepoznavanje"
- Hitri in vidni rezultati akcij
- Povleci - spusti
- Dano akcijo je možno izničiti

Slabe lastnosti:

- Težko načrtovanje
- Delovna domena je sedanjost (WYSIAWYG)
- Ikone niso vedno razumljive

Koncept direktne manipulacije (2)

Študije:

1. Preprosta opravila:
   - Koncept ukazov daje isti uspeh kot direktna manipulacije, če sta sistema dobro načrtana
   - Ikone hitreje naučene kot napisi
   - Uporabniki imajo raje napis kot ikono

2. Zahtevna opravila:
   - Koncept ukazov daje manjši uspeh kot direktna manipulacija
   - Ikone hitreje naučene kot napisi
   - Uporabniki imajo raje ikone

Grafični uporabniški vmesniki (GUV)

(Aplikativno usmerjeni uporabniški vmesniki)

Kaj je **grafični uporabniški vmesnik** ?

Je grafična predstavitev in interakcija s programi, podatki in objekti na računalniškem zaslonu.

WIMP: okna - pogled v aplikacije, ikone - pomanjšana okna, menuji, kazalec.
Klik po ikoni pomeni pognati aplikacijo.

Lastnosti:

- Zaslon z visoko resolucijo in pripravo za kazanje
- Tekst in grafika prikazana tako kot iztiskano
- Sledi interaktivni paradigmi objekt-akcija
- Ikone in objekti (okna) so prikazani vizualno
- Omogoča direktno manipulacijo ikon in objektov (oken)
- Ima menuje in dialoge za komunikacijo
- Rezultati akcij so vizualni
- Obstoja grafični prikaz sistemskih in uporabniških akcij in načinov (menuji, palete)
- Omogoča spremembo lastnosti vmesnika in interakcij
- Omogoča vhod preko tipkovnike, grafičnih gradnikov in drugih naprav

Objektno usmerjeni uporabniški vmesniki (OUUV)

Kaj je **objektno usmerjen uporabniški vmesnik** ?

Uporabnik se ukvarja z objekti, ki predstavljajo realni svet in ni obremenjen z arhitekturo računalnika ali kako uporabljati aplikacije in objekte.

- Je simulacija realnosti oziroma realizacija objektov realnega sveta
- Vmesnik ponuja množico *objektov*
- Uporabnik manipulira z objekti
- Okno je *pogled* v objekt
- Ikona je objekt
- Klik po ikoni pomeni odpreti objekt

3. X11

- X okenski sistem
- X okenski sistem - primera

X okenski sistem (1)

Povezati množico grafičnih postaj v mrežo in jih uporabiti kot učne pripomočke (MIT, DEC, IBM: Athena, 1984)

Cilj: poganjati aplikacije na oddaljenih sistemih

Rezultat: X okenski sistem (X Window System), 1988, prvo strojno neodvisno okolje (Athena danes: 6000 do 8000 postaj)

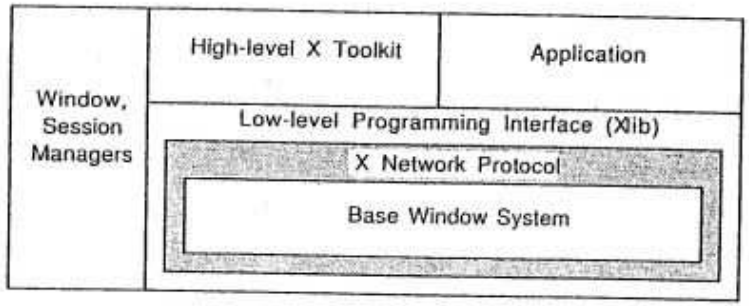
Lastnosti:

- Je neodvisen od operacijskega sistema (UNIX, MS-DOS, VMS)
- Je grafično okolje naslednjih firm (Apollo, Apple, AT&T, Bull, DEC, Fujitsu, HP, IBM, NCR, Sony, SUN)
- Temelji na sistemu okna (window) in konceptu odjemalec/strežnik
- Odjemalec in strežnik sta lahko na različnih strojih
- Uporablja ga več aplikacij istočasno

X okenski sistem (2)

Sestavni deli:

- Xlib - grafična knjižnica (enaka na vsakem stroju - 300 rutin, risanje, barve, teksti, interakcija z upravniki oken, dogodki, menuji, grafični kontekst)
- X mrežni protokol - definira podatkovne strukture za prenos podatkov med odjemalci in strežniki, zagotavlja neodvisnost od stroja in s tem možnost prikaza na oddaljenem stroju, omogoča uporabo več postaj naenkrat, uporabniki oken so tretirani kot aplikacije
- Xt orodja - dodatna programska knjižnica za podobe (drsniki, gumbi, menuji, okvirji oken, dialogi), njihovo geometrijo in delo z dogodki
- upravniki oken (uwm)



X okenski sistem (3)

Prikazovalnik (display) je abstrakcija, ki predstavlja vhodne in izhodne naprave (zaslon, tipkovnica, miška), enega uporabnika.

Strežnik kontrolira prikazovalnik:

- Nadzoruje dostop do prikazovalnika za odjemalce
- Pošilja sporočila po mreži
- Sledi zahtevam odjemalcev in ažurira okna
- Dvodimenzionalno risanje
- Omogoča distribuirano procesiranje
- Multiprocesiranje
- Sprejema vhodne dogodke in jih posreduje odjemalcem

Odjemalci imajo dostop do prikazovalnika:

- Tečejo simultano - več oken istočasno, editiranje več datotek naenkrat, poganjanje aplikacij na več računalnikih istočasno, aplikacije so neodvisne
- Uporabljajo mrežni protokol preko programskega vmesnika Xlib
- Med seboj komunicirajo preko strežnika - delijo si podatke
- Vsak posreduje zahtevo strežniku za kreiranje vsaj enega okna, ki je naslednik osnovnega okna - zaslon (root window)

X okenski sistem - primera

a. Kako vzpostaviti povezavo z X strežnikom

b. Postavljanje atributov okna

c. Napotki upravniku oken

d. Kreiranje grafičnega konteksta

e. Prikaz okna

f. Risanje

g. Prekinitev povezave z X strežnikom

- Delo z dogodki


```
/* draw.c - glavni program */
#include <X11/Xlib.h>

extern Display *theDisplay;

int main()
{
   Window theWindow; /*  openWindow(); */
   GC     theGC;

/* Vzpostavi zvezo z X serverjem */
   initX();

/* Odpri okno */
    theWindow = openWindow(100, 100, /* x, y lokacija */
                           300,300,  /* sirina, visina */
                           0,        /* to ni menu */
                           &theGC);  /* graficni kontekst */
/* Risi v okno */
   drawLine(theWindow, theGC,
            10, 10,      /* x1, y1 */
            100, 100);   /* x2, y2 */
   drawRectangle(theWindow, theGC, 100,100,100,100);
   fillRectangle(theWindow, theGC, 100,150,50,100);

/* Poslji izhod na prikazovalnik */
   XFlush(theDisplay);
   sleep(10);

/* Razkroji graficni kontekst */
   XFreeGC(theDisplay, theGC);
/* Razkroji okno */
   XDestroyWindow(theDisplay, theWindow);
/* Prekini povezavo z X serverjem */
   quitX();
}
```

```
/* connectX.c koda za komunikacijo z X streznikom */
#include <stdlib.h>
#include <stdio.h>
#include <X11/Xlib.h>   /* /usr/include/X11, /usr/local/include/X11 */
#include <X11/Xutil.h>

/* program - wide globals */
Display *theDisplay;            /* kazalec na strukturo prikazovalnika */
int     theScreen;              /* kateri zaslon na prikazovalniku */
int     theDepth;               /* stevilo bitnih ravnin - */
                                /* globina barvne ravnine */
unsigned long theBlackPixel;    /* sistemska crna barva */
unsigned long theWhitePixel;    /* sistemska bela barva */

/*
** initx(): vzpostavi zvezo z X streznikom
** (na mrezi) in shrani informacijo o okolju
*/

void initX()
{
    theDisplay = XOpenDisplay(NULL); /* ime prikazovalnika */
                                     /* kam gre izhod X streznika */
    if (theDisplay == NULL) {        /* preveri povezavo */
    fprintf(stderr,
                "Error: Cannot connect to the X server %s\n",
                 XDisplayName(NULL));
        exit(1);
    }

   theScreen = DefaultScreen(theDisplay);
                       /* kateri zaslon je v uporabi */
   theDepth  = DefaultDepth(theDisplay, theScreen);
                       /* 1: monokromatski zaslon */
   theBlackPixel = BlackPixel(theDisplay, theScreen);
   theWhitePixel = WhitePixel(theDisplay, theScreen);
}

/*
** Prekine zvezo z X streznikom
*/
void quitX()
  {
   XCloseDisplay(theDisplay);
 }
```

```
/* window.c - kreiranje okna */
#include <X11/Xlib.h>    /* prototipi k Xlib knjiznici */
#include <X11/Xutil.h>   /* pomozne operacije */
#include <stdio.h>

extern  Display  *theDisplay;       /* kazalec na strukturo prikazovalnika */
extern  int      theScreen;         /* kateri zaslon na prikazovalniku */
extern  int      theDepth;          /* stevilo bitnih ravnin */
extern unsigned long theBlackPixel; /* crna barva sistema */
extern unsigned long theWhitePixel; /* bela barva sistema */

#define BORDER_WIDTH 2              /* sirina roba okna */
#define LINE_WIDTH   1              /* sirina crte */

/* Kreiraj okno */
Window  openWindow(
      int x, int y,           /* x, y lokacija */
      int width, int height, /* sirina, visina */
      int flag,               /* to ni menu */
      GC  *theNewGC)          /* graficni kontekst */
{
/* */
XSetWindowAttributes  theWindowAttributes;
unsigned long         theWindowMask;
XSizeHints            theSizeHints;
Window                theNewWindow;

/* 1. Postavi zeljena atribute za okno */
theWindowAttributes.border_pixel     = theBlackPixel;
theWindowAttributes.background_pixel = theWhitePixel;
   /* Upravnik oken lahko zavrne nekatere nase atribute */
   /* True: okno bo tako kot si ga zelimo in brez naslova*/
theWindowAttributes.override_redirect = True;
   /* Katere komponente v strukturi postavljamo */
theWindowMask = CWBackPixel | CWBorderPixel | CWOverrideRedirect;
   /* Struktura je zatem poslana X strezniku */

/* 2. Odpri okno */
/* Vrne identifikacijo okna tipa Window (0: ce neuspeh) */
theNewWindow = XCreateWindow (theDisplay,          /* kateri prikazovalnik */
               RootWindow(theDisplay, theScreen), /* predhodnik okna - ozadje */
               x, y,                              /* kam, kako */
               width, height, BORDER_WIDTH, theDepth,
               InputOutput,                       /* razred okna (InputOnly */
         CopyFromParent,  /* razred barvnega okolja (GrayScale, TrueColor,.)*/
               theWindowMask,    /* maska sama ni del strukture */
                                 /* katere atribute uporabiti */
               &theWindowAttributes);

/* 3. Napotki upravniku oken */
theSizeHints.flags = PPosition | PSize; /* katere komponente so uporabljene */
                                  /* PPosition - program izbere pozicijo */
                                        /* PSize - program izbere velikost okna
*/
theSizeHints.x = x;   theSizeHints.y = y;
theSizeHints.width = width; theSizeHints.height = height;
            /* Poslji napotke upravniku oken */
            /* maska je del strukture */
XSetNormalHints(theDisplay, theNewWindow, &theSizeHints);
```

```
/* 4. Kreiraj graficni kontekst za dano okno */
if  (createGC(theNewWindow, theNewGC) == 0) {
      XDestroyWindow(theDisplay, theNewWindow);
      return((Window)0);
}

/* 5. Prikaz okna na zaslon (upravnik oken lahko intervenira) */
XMapWindow(theDisplay, theNewWindow);

/* 6. Komunikacija po mrezi je paketna - vrsta izhodnih zahtevkov X serverju */
/* Server skusa prenesti vec zahtevkov v enem komunikacijske paketu */
XFlush(theDisplay);

return(theNewWindow);
}

/* Kreiraj graficni kontekst */
createGC(
     Window  theNewWindow,
     /* Barva ospredja, ozadja, sirina crte, stil crte, font*/
     GC      *theNewGC)
{
   XGCValues  theGCValues;
   unsigned long theGCMask; /* maska */

   /* napotki za graficni kontekst pridruzen danemu oknu */
               /* katere komponente so uporabljene */
   theGCMask = GCLineWidth | GCLineStyle | GCForeground;
   /* artibuti za graficni kontekst */
   theGCValues.line_width = LINE_WIDTH;     /* sirina crt (0) */
   theGCValues.line_style = LineOnOffDash; /* stil crte (LineSolid) */
   theGCValues.foreground = theBlackPixel; /* barva ospredja */

   /* kreiraj now graficni kontekst za dano okno */
   *theNewGC = XCreateGC(theDisplay, theNewWindow,
               /* maska ni del strukture */
               theGCMask, &theGCValues);

   if(*theNewGC == 0) return(0);
   else {                                /* barva ozadja */
      XSetBackground(theDisplay, *theNewGC, theWhitePixel);
      return(1);
   }
}
```

```
/* drawx.c - rutine za risanje */
#include <X11/Xlib.h>
#include <X11/Xutil.h>

extern Display *theDisplay;

drawLine(
     Window   theWindow,
     GC       theGC,
     int x1, int y1,
     int x2, int y2)
{
   XDrawLine(theDisplay,
        theWindow, theGC, x1, y1, x2, y2);
}

drawRectangle(
     Window   theWindow,
     GC       theGC,
     int x, int y,
     int width, int height)
{
   XDrawRectangle(theDisplay,
        theWindow, theGC, x, y, width, height);
}

fillRectangle(
     Window   theWindow,
     GC       theGC,
     int x, int y,
     int width, int height)
{
  XFillRectangle(theDisplay,
        theWindow, theGC, x, y, width, height);
}
```

```
/* Deklaracije potrebne za definicijo podatkov */
/* za uporabo z zahtevki X okenskega sistema */
#include <stdio.h>
#include <string.h>
#include <X11/Xlib.h>
#include <X11/Xutil.h>

/* Deklaracije */
char hello[] = {"Hello, World."};
char hi[]    = {"Hi!"};

int main(int argc, char **argv)
{

Display  *mydisplay; /* Strukt. kontrolira povezavo s prikazovalnikom */
Window   mywindow;   /* Okno, ki ga aplikacija kreira */
GC       mygc;       /* Graficni kontekst */
XEvent   myevent;    /* Unija struktur za razlicne tipe dogodkov */
                     /* Postaja (dogodek), X sistem, opis dogodka */
XSizeHints myhint;   /* Tu proces posreduje informacijo o oknu */
                     /* upravniku oken */
int      myscreen;   /* steviXSlka zaslona */
unsigned long myforeground, mybackground;
                     /* Barvi ospredja in ozadja za risanje*/
int      i;
char     text[10];
int      done;       /* */

/* Inicializacija */

mydisplay = XOpenDisplay(""); /* Inicializira povezavo med */
                              /* aplikacijo in X streznikom, shrani */
                              /* opis povezave. Vzame ime prikazovalnika */

                    /* kateri zaslon je v uporabi */
myscreen     = DefaultScreen(mydisplay);
mybackground = WhitePixel(mydisplay, myscreen);
myforeground = BlackPixel(mydisplay, myscreen);
                    /* Barve za risanje ozadja */
                    /* tock, teksta in robov */

        /* Kam ? levo zgoraj */
myhint.x = 200;     myhint.y = 300;
        /* Sirina, dolzina */
myhint.width = 350; myhint.height = 250;
        /* Katere komponente v strukturi XSizeHints */
        /* se uporabljajo. Program izbira pozicijo */
        /* in velikost okna */
myhint.flags = PPosition | PSize;

/* Zahteva postaji, da kreira okno aplikacije */
/* Okno je kreirano v oknu, ki pokriva cel ekran */
/* Rob okna je sirok pet tock in crn */
mywindow = XCreateSimpleWindow(mydisplay,
                    DefaultRootWindow(mydisplay), /* predhodnik */
                    myhint.x, myhint.y, myhint.width, myhint.height,
                    5, myforeground, mybackground);

/* Informacija o na novo kreiranem oknu za upravnika oken */
XSetStandardProperties(mydisplay, mywindow, hello, hello,

                       0, argv, argc, &myhint);
```

```
/* kreira graficni kontekst */
mygc = XCreateGC(mydisplay, mywindow, 0, 0);
XSetBackground(mydisplay, mygc, mybackground);
XSetForeground(mydisplay, mygc, myforeground);

/* Program zeli biti obvescen o vsakem dogodku z miske, */
/* tipkovnice, ali ob osvezevanju zaslona */
XSelectInput(mydisplay, mywindow,
             ( ButtonPressMask | KeyPressMask | ExposureMask));

/* prikaz okna na zaslon */
XMapRaised(mydisplay, mywindow);

done = 0;
while (done == 0) {
   XNextEvent (mydisplay, &myevent);  /* naslednji dogodek */
      switch (myevent.type) {
         case KeyPress:      /* dogodek s tipkovnice */
             done = 1;
             break;
         case Expose:        /* dogodek ob zagonu programa */
            if (myevent.xexpose.count == 0)
                 XDrawImageString(
                     myevent.xexpose.display,
                     myevent.xexpose.window,
                     mygc, 50, 50,              /* kam v oknu zapise tekst */
                     hello, strlen(hello) );
             break;
          case ButtonPress:     /* dogodek z miske */
              XDrawImageString(
                   myevent.xbutton.display,
                   myevent.xbutton.window,
                   mygc, myevent.xbutton.x, myevent.xbutton.y,
                   hi, strlen(hi));
             break;
          } /* switch */
 } /* while */

/* Razkroji graficni kontekst */
XFreeGC(mydisplay,mygc);
/* Razkroji okno */
XDestroyWindow(mydisplay, mywindow);
/* Prekine zvezo z X streznikom */
XCloseDisplay(mydisplay);

return 0;

} /* main */
```

• Motif (uporaba liste argumentov)

```
Arg  args[5];
int  n = 0;
. . .
XtSetArg(args[n], XmNlabelString, label); n++;
. . .



. . .
button = XtCreateManagedWidget("pushme",
        xmPushButtonWidgetClass, toplevel, args, n);
. . .
```


• Motif (glavno okno - MainWindow)

Je običajno prva podoba aplikacije. Načrtana je z namenom podpreti predlagani stil aplikacije (*Motif Style Guide*), ki vključuje:

- Vrstični menu (*MenuBar*)
- Delovno polje z ali brez drsnikov za glavne gradnike aplikacije (npr.: *DrawingArea, ScrolledList, Text*)
- Ukazno polje
- Polje za obvestila

Razred podobe je izveden iz razreda *ScrolledWindow*. Deduje vse atribute vključno z resursi.

Core->Composite->Constraint->
->Manager->ScrolledWindow->MainWindow

Ima dodane lastnosti za vsebovanje vrstičnega menuja, polja za ukaze in polja za obvestila.

• Motif (glavno okno - gradnja)

```
#include <Xm/MainW.h>
...
Widget toplevel, main_w;
...
main_w = XtVaCreateManagedWidget("main_window",
        xmMainWindowWidgetClass, toplevel,
        XmNscrollBarDisplayPolicy, XmAS_NEEDED,
        XmNscrollingPolicy, XmAUTOMATIC,
        NULL);
...
```

Drsniki se pojavijo po potrebi.
100 X 100 pikslov; sicer XmNwidth, XmNheight

XmNscrollBarDisplayPolicy:
XmAS_NEEDED, XmSTATIC

XmNscrollingPolicy:
XmAUTOMATIC: Xt notranjost kreira in upravlja drsnike.
XmAPPLICATION_DEFINED:
Aplikacija ali npr.: *ScrolledList* ali *ScrolledText* kreirajo in upravljajo drsnike.

• Motif (vrstični menu - MenuBar)

Ni samostojna podoba pač pa množica kaskadnih gumbov (*CascadeButton*) razvrščenih horizontalno v podobi vrstica-kolona (*RowColumn*). Vsak gumb je povezan z menujem tipa *PulldownMenu*, ki lahko vsebuje gradnike tipa: *PushButton, ToggleButton, Label* in *Separator*.

```
XmString  file, help;
Widget    toplevel, main_w, menubar;
...
/* Vrsticni menu z dvema menujema */
file = XmStringCreateLocalized("File");
help = XmStringCreateLocalized("Help");
menubar = XmVaCreateSimpleMenuBar(main_w, "menubar",
      XmVaCASCADEBUTTON, file, 'F',
      XmVaCASCADEBUTTON, help, 'H',
      NULL);
XmStringFree(file);
XmStringFree(help);
...
```

XmVaCASCADEBUTTON, file, 'F': vedno, oznaka, mnemonik

Vsak gumb ima pridruženo ime, ki ni ime podobe: "button_*n*"

```
...
if (widget = XtNameToWidget(menubar, "button_1"))
   XtVaSetValues(menubar, XmNmenuHelpWidget, widget, NULL);
...
```

• Motif (izvlečni menu)

Vsak gumb tipa *CascadeButton* vrstičnega menuja mora imeti pridružen menu tipa *PulldownMenu*.

```
XmString  open, quit;
Widget    toplevel, main_w, menubar, menu;
...
open = XmStringCreateLocalized("Open...");
quit = XmStringCreateLocalized("Quit");
menu = XmVaCreateSimplePulldownMenu(menubar, "file_menu",
      0,          /* Indeks gumba v vrsticnem menuju */
      file_cb,    /* Odzivna funkcija na vsak gumb*/
      /* konstanta, sest. niz, mnemonik, posp., sest. niz */
      XmVaPUSHBUTTON, open, 'O', NULL, NULL,
      XmVaSEPARATOR,
      XmVaPUSHBUTTON, quit, 'Q', NULL, NULL,
      NULL);
XmStringFree(open);
XmStringFree(quit);
...
XtManageChild(menubar)
XtVaSetValues(main_w, XmNmenuBar, menubar, NULL);
XtRealizeWidget(toplevel);
...
```

Simbolične konstante določajo tip elementov menuja:
XmVaPUSHBUTTON, XmVaTOGLEBUTTON,
XmVaRADIOBUTTON, XmVaCASCADEBUTTON,
XmVaSEPARATOR
Vsak gumb ima pridruženo ime: "button_*n*", "separator_*n*".

• Motif (pospeševalniki)

```
XmString  quit_ac;
...
quit_ac = XmStringCreateLocalized("Ctrl-C");
...
menu = XmVaCreateSimplePulldownMenu(menubar, "file_menu",
      0,
      file_cb,
      /* konstanta, sest. niz, mnemonik, posp., sest. niz */
      XmVaPUSHBUTTON, open, 'O', NULL, NULL,
      XmVaSEPARATOR,
      XmVaPUSHBUTTON, quit, 'Q', "Ctrl<Key>C", quit_ac,
      NULL);
...
XmStringFree(quit_ac);
...
```

• Motif (dialog za izbiro datoteke - FileSelectionDialog (1))

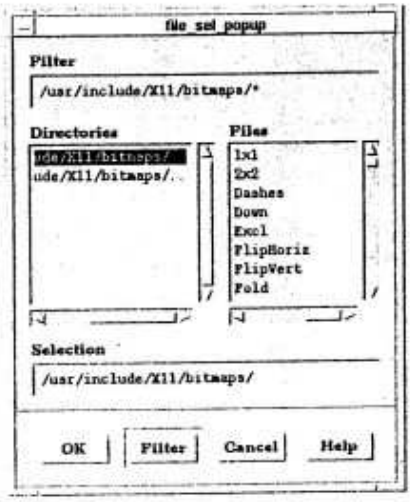
Odzivna funkcija se aktivira na izbiro kateregakoli gumba v menuju.

```
/* Izbrana je bila opcije v menuju "File" */
#include <Xm/FileSB.h>
. . .
void file_cb(widget, client_data, call_data)
Widget   widget;       /* Izbrani menu */
XtPointer client_data;  /* Indeks v menuju */
XtPointer call_data;    /* Neuporabljeno */
{
   static Widget dialog;  /* Za ponovno uporabo */
   int item_no = (int)client_data;
   if (item_no == 0) {     /* Opcija Open */
       /* Kreiraj FileSelectioDialog */
       dialog = XmCreateFileSelectionDialog(toplevel,
            "file_sel", NULL, 0);
       /* Dodaj odzivni funkciji */
       XtAddCallback(dialog,XmNokCallback,
            echo_file,NULL);
       XtAddCallback(dialog,XmNcancelCallback,
            XtUnmanagedChild,NULL);
       XtManageChild(dialog);
   }
   else if (item_nu == 1)  /* Opcija Quit */
      exit(0);
}
```


• Motif (odzivna funkcija izvlečnega menuja)

Odzivna funkcija se aktivira na izbiro kateregakoli gumba v menuju.

```
/* Izbrana je bila opcije v menuju "File" */
void file_cb(widget, client_data, call_data)
Widget   widget;       /* Izbrani menu */
XtPointer client_data;  /* Indeks v menuju */
XtPointer call_data;    /* Neuporabljeno */
{
    int item_no = (int)client_data;
    if (item_no == 0)      /* Opcija Open */
         actionOpen();
    else if (item_nu == 1)  /* Opcija Quit */
         exit(0);
}
```



• Motif (dialog za izbiro datoteke (2))

Odzivna funkcija pridružena opciji OK v dialogu za izbiro datoteke:

```
void echo_file(dialog, client_data, call_data)
Widget    dialog;
XtPointer  client_data;
XtPointer  call_data;
{
      char *filename;
      XmFileSelectionBoxCallbackStruct *cbs =
            XmFileSelectionBoxCallbackStruct *) call_data;

      XmStringGetLtoR(cbs->value,
            XmFONTLIST_DEFAULT_TAG, &filename);

      printf("Filename: %s", filename);
}
```


• Motif (dvižni menu - PopupMenu)

```
. . .
XmString  line, square;
Widget    toplevel, main_w, drawing_a, popup_menu;
. . .
line = XmStringCreateLocalized("Line");
square = XmStringCreateLocalized("Square");
popup_menu = XmVaCreateSimplePopupMenu(drawing_a,
      "popup",
      popup_cb,       /* Odzivna funkcija za vsak gumb*/
      XmVaPUSHBUTTON, line, 'L', NULL, NULL,
      XmVaPUSHBUTTON, square, 'S', NULL, NULL,
      NULL);
. . .
/* Odzivna funkcija input za vse dogodke v drawing_a */
XtAddCallback(drawing_a,XmNinputCallback,
      input, popup_menu);
. . .
XtRealizeWidget(toplevel);
XtAppMainLoop(app);
. . .
```

- **Motif (kaskadni menu - CascadingMenu)**

(*Pulldown* v *Pulldown* ali *Popup* menuju):

```
. . .
XmString  line, width, w1, w2;
Widget    toplevel, main_w, drawing_a, popup_menu, cascade;
. . .
line = XmStringCreateLocalized("Line");
width = XmStringCreateLocalized("Width");
popup_menu = XmVaCreateSimplePopupMenu(drawing_a,
      "popup", popup_cb,   /* Odzivna funkcija na vsak gumb*/
      XmVaPUSHBUTTON, line, 'L', NULL, NULL,
      XmVaCASCADEBUTTON, width, 'W',
      NULL);
. . .
w1 = XmStringCreateLocalized(" 2 ");
w2 = XmStringCreateLocalized(" 4 ");
cascade = XmVaCreateSimplePullDownMenu(popup_menu,
      "cascade", 1,    /* Odmik */
      set_width,       /* Odzivna funkcija za vsak gumb*/
      XmVaPUSHBUTTON, w1, '2', NULL, NULL,
      XmVaPUSHBUTTON, w2, '4', NULL, NULL,
      NULL);
/* Odzivna funkcija input za vse dogodke v drawing_a */
XtAddCallback(drawing_a,XmNinputCallback,
      input, popup_menu);
. . .
XtRealizeWidget(toplevel);
XtAppMainLoop(app);
```


- **Motif (opcijski menu - OptionMenu)**

```
. . .
XmString  line, square, draw_shape;
Widget    toplevel, main_w, rc, drawing_a, option_menu;
. . .
line = XmStringCreateLocalized("Line");
square = XmStringCreateLocalized("Square");
draw_shape = XmStringCreateLocalized("Draw_mode");
option_menu = XmVaCreateSimpleOptionMenu(rc,
      "option_menu",
      draw_shape, 'D', 0,  /* Zacetna izbira v menuju */
      option_cb,           /* Odzivna funkcija za vsak gumb*/
      XmVaPUSHBUTTON, line, 'L', NULL, NULL,
      XmVaPUSHBUTTON, square, 'S', NULL, NULL,
      NULL);
. . .
XtRealizeWidget(toplevel);
XtAppMainLoop(app);
. . .
```

• Motif (lastnosti aplikacije (1))

Možna mesta določanja lastnosti podob:

1. Med gradnjo podob v programu
2. Z argumenti ukazne vrstice
3. V datoteki:
   - Domači direktorij (./<ime razreda>)
   - Kjerkoli (XAPPLRESDIR/<ime razreda>)
   - Na sistemsko določenem direktoriju: (/usr/openwin/lib/X11/app-defaults/<ime razreda>)
4. Rezervne lastnosti v kodi programa

Prioriteta pada od zgoraj navzdol.

• Motif (lastnosti aplikacije (2))

1. Med gradnjo podob v programu:

```
XtVaCreateManagedWidget("pushme",
        xmPushButtonWidgetClass, toplevel,
        XmNlabelString, label,
        XmNwidth, 200,
        XmNheight, 50,        /* Ime lastnosti zacne z XmN */
        NULL);
```

1.a. Po gradnji podobe:

```
XtVaSetValues (widget_id,
        resorce-value-list,
        NULL);
```

1.b. Branje lastnosti podobe:

```
extern Widget  label;
XmString       string;
Dimension      width;
...
XtVaGetValues (label,
        XmNlabelString, &string,
        XmNwidth, &width,
        NULL);
```

• Motif (lastnosti aplikacije (3))

2. Z argumenti ukazne vrstice:

Opcija	**Primer**
-bg	-bg blue
-display	-display zuse:0
-xrm	-xrm "*XmPushButton.background: red"
-xrm	-xrm "*pushme.fontList: -adobe-courier-medium-o-normal - -14-100-100-100-m-90-iso8859-1"

• Motif (lastnosti aplikacije (4))

3. V datoteki:

Funkcija	**Imena povezano s podobo**
XtVaAppInitialize()	"Button"
XmCreateBulletinBoard()	"bboard"
XmCreateManagedWidget()	xmPushButtonWidgetClass, "button"

Vsebina datoteke Button:

```
!
! Splosne lastnosti in obnasanje
!
*fontList:                  -*-courier-medium--*--12-
*shadowThickness:           3
!
Button.bboard.height:       150
*bboard.width:              250
*bboard.background          sky blue
!
*button.background:         goldenrod
*XmPushButton.height:       30
*XmPushButton.width:        100
*XmPushButton.x:            75
Button.bboard.button.y:     60
```

• Motif (lastnosti aplikacije (5))

3. V datoteki (primer s pospeševalniki):

```
. . .
/* "Aplikacija", "main_window", "menubar", "file_menu */
. . .
menu = XmVaCreateSimplePulldownMenu(menubar, "file_menu",
        0,
        file_cb,
        /* konstanta, sest. niz, mnemonik, posp., sest. niz */
        XmVaPUSHBUTTON, open, 'O', NULL, NULL,
        XmVaSEPARATOR,
        XmVaPUSHBUTTON, quit, 'Q', NULL, NULL,
        NULL);
. . .
```

```
*main_window.menubar.file_menu.button_0.accelerator: Ctrl<Key>O
*main_window.menubar.file_menu.button_0.acceleratorText: Ctrl+O
*main_window.menubar.file_menu.button_1.accelerator: Ctrl<Key>C
*main_window.menubar.file_menu.button_1.acceleratorText: Ctrl+C
```


• Motif (lastnosti aplikacije (6))

3. V datoteki (pravila):

Primer: (Aplikacija **XCalc**, datoteka **XCalc**)

```
*XmPushButton.background: orange
XCalc*XmPushButton.background: gray
XCalc*pb1.background: red
XCalc*XmForm*background: blue
XCalc*XmForm.background: yellow
XCalc*background: green
```

Pravila nastavljanja lastnosti:

1. Objekti imajo prednost pred razredi na istem nivoju hierarhije.
2. Komponente levo pogojujejo prednost pred tistimi desno.
3. Eksplicitno izraženi objekti in razredi imajo prednost pred komponentami označenimi z zvezdicami.
4. Tesno povezane komponente v izrazih imajo prednost pred ohlapno povezanimi.

Lastnosti:

- Barva ozadja vsake podobe z imenom pb1 (XmPushButton) bo rdeča in ne siva. (#1)
- Barva ozadja podob tipa XmPushButton bo siva in ne oranžna. (#2).
- Barva ozadja podob, ki niso tipa XmPushButton in so nasledniki podobe tipa XmForm, bo modra in ne zelena. (#3)
- Barva ozadja podobe tipa XmForm bo rumena in ne modra. (#4)

- **Motif (lastnosti aplikacije (7)))**

4. Rezervne lastnosti v kodi programa:

```
...
toplevel = XtVaAppInitialize(&app, "Hello", NULL, 0,
    &argc, argv, fallback_resources,
    NULL);
...
...
String fallback_resources[ ] = {
    "Hello.pushme.fontList: -*-courier-medium--*--12-",
    "Hello*background: red",
    "*XmPushButton.width: 30",
    ...
    NULL};
...
```

• Motif (barve)

```
*bboard*background: sky blue
*bboard*background: #87ceeb
```

/usr/openwin/lib/X11/rgb.txt

```
...
135 206 235  sky blue
255 255 255  white
0   0   0    black
...
```

• Motif (nabori znakov)

xfd, xfontsel

/usr/openwin/lib/X11/fonts/*/fonts.dir

• Motif (okvir - Frame)

Gradnja:

```
#include <Xm/Frame.h>

...
frame = XtVaCreateManagedWidget ("frame",
    xmFrameWidgetClass, rowcol,
    XmNshadowType, XmSHADOW_IN,
    XmNshadowThickness, 3,
    NULL);
...
```

XmNshadowType: (tridimenzionalni izgled)
XmSHADOW_IN
XmSHADOW_OUT

XmNshadowThickness: (debelina okvirja)

• Motif (vrstica-kolona - RowColumn)

- Naslednike razvrsti po vrsticah in (ali) kolonah
- Primeren za implementacijo menujev
- Možno razvrščanje naslednikov po realizaciji

Gradnja:

```
#include <Xm/RowColumn.h>
...
parent = XtVaCreateManagedWidget("rowcolumn",
      xmRowColumnWidgetClass, toplevel,
      XmNpacking,        XmPACK_COLUMN,
      XmNnumColumns,  5,
      XmNorientation,    XmVERTICAL,
      NULL);
      ...
```

XmNpacking: (dimenzije naslednikov)
XmPACK_TIGHT - nasledniki so svojih lastnih dimenzij
XmPACK_COLUMN - vsi nasledniki so istih dimenzij

XmNorientation: (kam dodati naslednjo podobo)
XmVERTICAL, XmHORIZONTAL
XmNnumColumns: (število kolon ali vrstic)

• Motif (vrstica-kolona - odzivna funkcija)

```
...
char *strings[ ] = {"One", "Two", "Three", "Four"};
...
parent = XtVaCreateManagedWidget("rowcolumn",
      xmRowColumnWidgetClass, toplevel, NULL);
XtAddCallback(parent,XmNentryCallback, called, NULL);
...
for (i=0; i < XtNumber(strings); i++) {
      w = XtVaCreateManagedWidget(strings[i],
      xmPushButtonGadgetClass, parent, NULL);
      XtAddCallback(w, XmNactivateCallback, never, i+1);
}
...
...
void called(widget, client_data, call_data);
Widget widget;
XtPointer client_data;
XtPointer call_data;
{
      XmRowColumnCallbackStruct *cbs =
          (XmRowColumnCallbackStruct *) call_data;
      Widget pb = cbs->widget;
...
      printf("%s %s: %d n",
          XtName(widget), Xtname(pb), cbs->data);
...
}
```

• Motif (upravljanje podob (1))

```
static char *labels[ ] = {"Oznaka", "Druga oznaka", "Tretja"};
XmString label;
Widget widget, rc;
Arg args[3];
int i, n = 0;

/* Zgradi neupravljano podobo tipa RowColumn (predhodnik) */
rc = XtCreateWidget("rc", xmRowColumnWidgetClass,
        parent, NULL, 0);

/* Zgradi naslednike; vsi so dimenzij 50 X 50 a razlicnih oznak */
XtSetArg(args[n], XmNwidth, 50); n++;
XtSetArg(args[n], XmNheight, 50); n++;
for (i = 0; i < XtNumber(labels); i++) {
        label = XmStringCreateLocalized(labels[i]);
        XtSetArg(args[n], XmNlabelString, xm_label);
        widget = XtCreateManagedWidget("label",
            xmLabelWidgetClass, rc, args, n+1);
        XmStringFree(label);
}

/* Sedaj upravljaj podobo tipa RowColumn */
XtManageChild(rc);
```


• Motif (upravljanje podob (2))

- Upravnik upravlja svoje naslednike z nadzorom njihove velikosti in njihovega položaja
- Upravniki so zgrajeni in razkrojeni kot vse druge podobe
- Razlike nastopijo, ko je dana podoba deklarirana kot upravljana med procesom gradnje
- Podobi se realizirata, če sta upravnik in naslednik v upravljanem stanju
- Če je naslednik v neupravljanem stanju, ga upravnik preskoči
- Če je upravnik v neupravljanem stanju, potem ne upravlja s svojimi nasledniki

Priporočilo:

XtVaCreateManagedWidget()	za osnovne gradnike,
XtVaCreateWidget()	za upravnike.

- **Motif (upravnik z vertikalno nastavljivimi nasledniki - PanedWindow)**

Gradnja:

```
#include <Xm/Label.h>
#include <Xm/PanedW.h>
#include <Xm/Text.h>
...
paned = XtVaCreateWidget ("pane",
    xmPanedWindowWidgetClass, toplevel,
    NULL);
XtVaCreateManagedWidget ("Hi !", xmLabelWidgetClass, paned,
    XmNpaneMinimum, 25,
    XmNpaneMaximum, 85,
    NULL);
XtVaCreateManagedWidget ("text", xmTextWidgetClass, paned,
    XmNrows, 10,
    XmNcolumns, 60,
    XmNpaneMinimum, 35,
    XmNeditMode, XmMULTI_LINE_EDIT,
    XmNvalue,"This example demonstrates PanedWindow widget.",
    NULL);
XtVaCreateManagedWidget ("Bye, bye", xmLabelWidgetClass, paned,
    XmNpaneMinimum, 25,
    XmNpaneMaximum, 85,
    NULL);
...
```

• Principi in navodila (kaj je kvalitetno načrtovanje ?)

Uporabnost programske opreme kvantificiramo v smislu naslednjih ciljev:

- Lahka za naučiti
- Lahka za uporabljati
- Prijetna za uporabo

Razvijalci in uporabniki danes uporabljajo *beta* verzije operacijskih sistemov, razvojnih orodij, aplikacij.

(Nižji standard od tistega kar konstituira sprejemljivo kvaliteto.)

Kriteriji, ki vplivajo na načrtovanje:

- Izkušnje
- Razumevanje uporabnika
- Učinkovit postopek načrtovanja
- Potrebe (socialni, ekonomski prispevek)
- Lahko za naučiti
- Uporabnost
- Primernost (učinkovitost, praktičnost)
- Estetika (grafika, prijetnost)
- Praktičnost (instalacija, učenje, cena, podpora, nadgradnje)

• Principi (1)

(Apple, 1992; IBM, 1992; Microsoft, 1995; UNIX OSF/Motif, 1993)

So visokonivojski koncepti in navodila za načrtovanje grafičnih uporabniških vmesnikov. Nastali sa na osnovi človekovega mentalnega modela in njegovih fizičnih ter psiholoških lastnosti.

1. Zagotovi nadzor uporabnika
2. Reduciraj obremenitev uporabnikovega spomina
3. Zagotovi konsistenčnost vmesnika

• Principi (2)

1. *Zagotovi nadzor uporabnika:*
   - Omogoči uporabo tipkovnice in miške (fleksibilnost)
   - Omogoči prekinitev danih opravil (prekinljivost)
   - Prikazuj obvestila in tekste (pomoč)
   - Zagotovi takojšnje in ponovljive akcije ter povratno informacijo (odzivnost)
   - Zagotovi značilne poti in izhod (navigacija)
   - Prilagodi se uporabnikom z različnimi nivoji znanja (dostopnost)
   - Zagotovi jasnost in čistost vmesnika (preglednost)
   - Omogoči spreminjanje lastnosti vmesnika (želje)
   - Omogoči direktno manipulacijo z grafičnimi gradniki (interaktivnost)
   - Uporabljaj "načine" pametno (nedvoumnost)

• Principi (3)

2. *Reduciraj obremenitev uporabnikovega spomina:*
   - Razbremenjuj kratkotrajni spomin (pomnenje)
   - Zanašaj se na "razpoznavanje" in ne na spomin (prepoznavanje)
   - Zagotovi vizualne namige (informiranost)
   - Zagotovi vgrajene akcije in razveljavitev ter ponovitev akcij (preprostost, reševanje)
   - Zagotovi bližnjice (hitrost)
   - Podpiraj način gradnik - akcija (intuitivnost)
   - Uporabljaj metafore realnega sveta (prenos)
   - Uporabljaj progresivni dostop (navigabilnost)
   - Podpiraj vizualno čistost (organiziranost)

• Principi (4)

3. *Zagotovi konsistenčnost vmesnika:*
   - Ohranjaj kontekst uporabnikovih opravil (zveznost)
   - Ohranjaj enovitost v predstavitvi informacij, obnašanju gradnikov in tehnikah interakcije (izkušnje)
   - Ohranjaj enovitost rezultatov interakcij (pričakovanje)
   - Zagotovi estetsko privlačnost in polnost (izgled)
   - Vspodbujaj preiskovanje (napovedljivost)

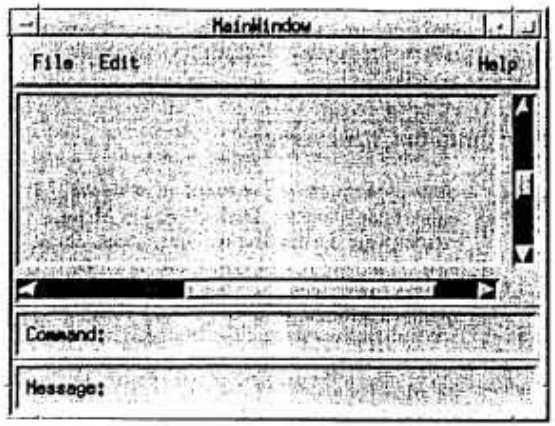
• Navodila (okna (1))

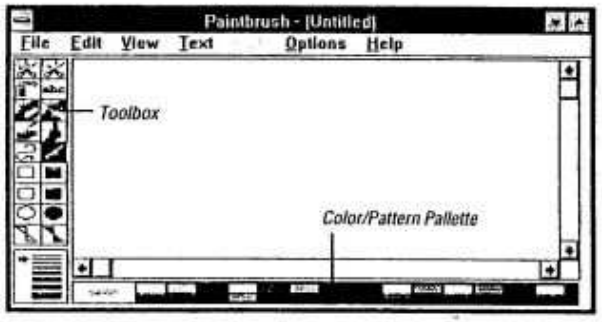
Delovna miza -> računalniški zaslon

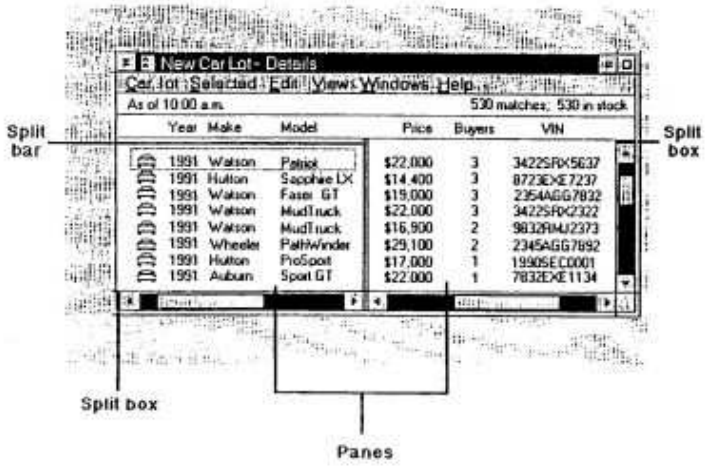
(Različne vrste in nivoji informacij, sekvenčnost, simultan dostop do virov informacij, več opravil naenkrat, razbremenitev STM spomina, preglednost, večkratna predstavitev istega opravila)

Komponente oken:

- Okvir
- Naslov
- Sistemski menu
- Gumbi za velikost okna
- Vrstični menu
- Delovno področje
- Vrstica za izpis statusa
- Vrstica za obvestila
- Vrstica za ukaze
- Drsniki
- Pregrade
- Polje za prikaz permanentnih ukazov ali opcij







• Navodila (okna (2))

Okna glede na vrsto opravil:

- Primarno okno - okno aplikacije (aktivnosti dostopne uporabniku v vsakem trenutku)
- Sekundarna okna za razširitev aplikacije (brez vrstičnega menuja, obravnava enega podatkovnega objekta)
- Dialogi (akcije omejenega konteksta, kritični in paralelni dialogi)

Kompozicije oken:

- Drug poleg drugega (eno opravilo, predvidljiva količina podatkov, omejeno število oken, lažja manipulacija)
- Prekrivajoča okna (več opravil, nepredvidljiva količina podatkov, večja okna, poljubno število oken, težja manipulacija)

• Navodila (menuji (1))

Splošna navodila za načrtovanje menujev glede na:

- Prikaz opcij (permanenten, na zahtevo)
- Organizacijo opcij (splošni menu, prikaz le releventnih alternatitiv na danem nivoju, omejeno število nivojev, neaktivne možnosti prikazano šibko)
- Grupiranje opcij (po hierarhiji in z ozirom na logičnost, različnost in medsebojno izključevanje, uporaba separatorjev)
- Vrstni red opcij (po vrstnem redu uporabe, pogostosti, abecedi, možnem destruktivnem učinku, relativnem položaju glede na sorodne menuje)
- Najverjetnejšo opcijo (kurzor)
- Navigacijo (vnaprejšnji prikaz opcij)
- Indikatorje naslednikov (▶, ...)
- Ekvivalente preko tipkovnice (mnemoniki, pospeševalniki)
- Izvršitev opcij (izbor, izvršitev)

• Navodila (menuji (2))

Vrste menujev:

1. Vrstični menu (menu bar)
2. Ruletni (izvlečni) menu (pulldown menu)
3. Kaskadni menu (cascading menu)
4. Dvižni menu (pop-up menu)
5. Menu z ikonami (iconic menu)

Izbor menuja:

- Glede na število opcij
- Pogostost uporabe
- Pogostost spremembe vsebine menuja

1.) Pogoste akcije, skupne akcije za sekundarna okna

2.) Pogoste akcije, za opcije predstavljene tekstualno, redke spremembe vsebine menuja, 5 - 10 opcij

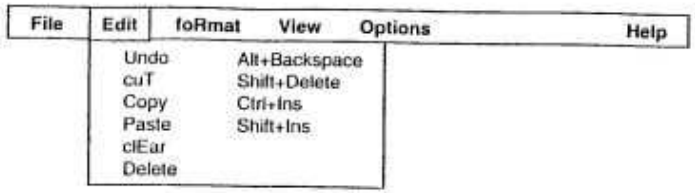
3.) Zagotovi preprostejši in preglednejši predhodni menu; za opcije, ki se medsebojno izključujejo, 1 - 2 kaskadi

4.) Pogosta uporaba akcij v danem kontekstu, redke spremembe vsebine menuja, malo prostora na zaslonu, 5 - 10 opcij

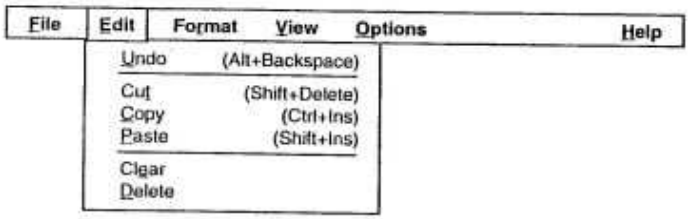
5.) Za izbiro aplikacije, za dosego posebnih funkcij dane aplikacije

• Navodila (menuji (3))

Primer načrtovanja:

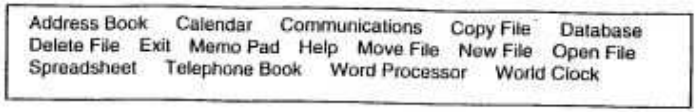


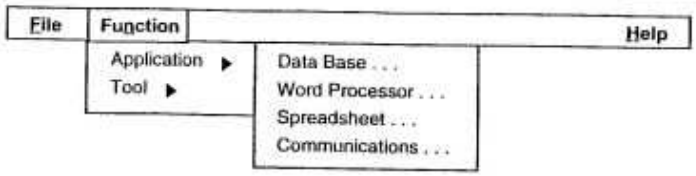
- Mnemoniki
- Grupiranje
- Pospeševalniki

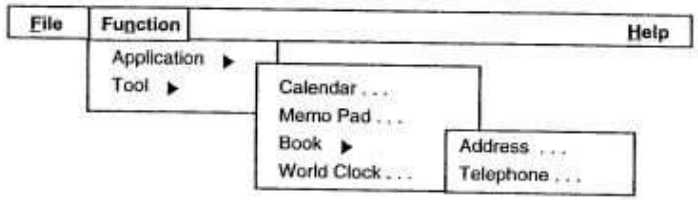


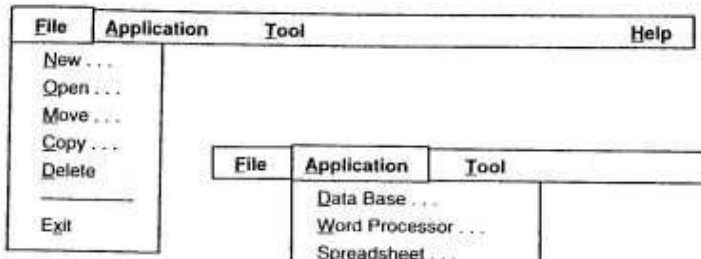
• Navodila (menuji (4))

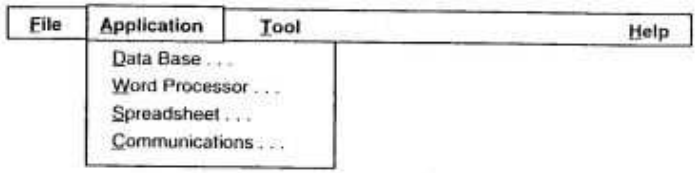
Primer načrtovanja:

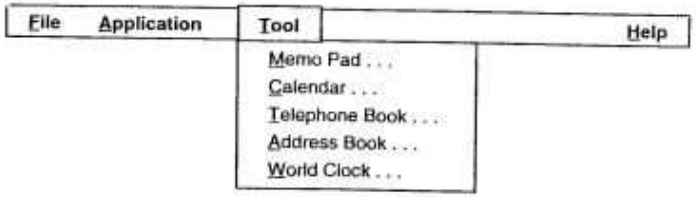










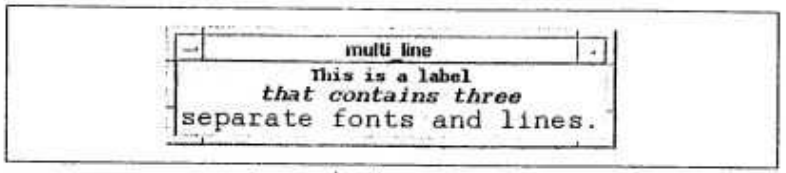


• Motif (osnovni gradniki)

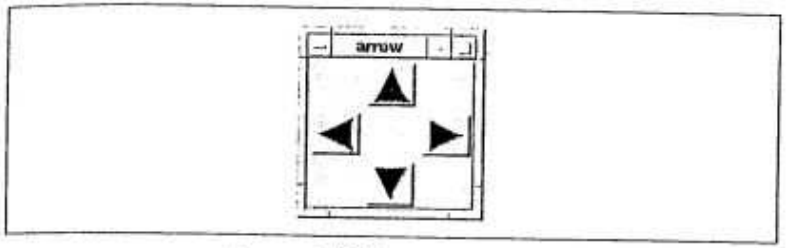
Razredi podob v Xt in Motif:

- Osnovni gradniki (*primitives*) - nimajo naslednikov
  - Puščica (**ArrowButton**) - gumb
  - Oznaka (**Label**) - tekst, ikona
    - * **PushButton** - gumb
    - * **CascadeButton** - kaskadni gumb
    - * **ToggleButton** - stikalo
    - * **DrawnButton** - gumb z ikono
  - Drsniki (**ScrollBars**) - opazovanje omejenega okna, prikaz statičnega teksta ali slike
  - Separator (**Separator**) - ločuje gradnike vmesnika
  - Tekst (**Text**) - prikaz in editiranje teksta
  - Tekst (**TextField**) - prikaz in editiranje teksta ene vrstice
  - Lista (**List**) - polje tekstualnih zapisov z možnostjo izbire

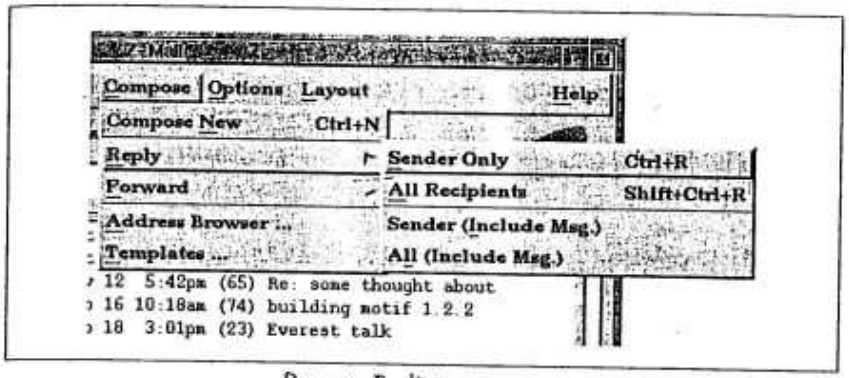
Label



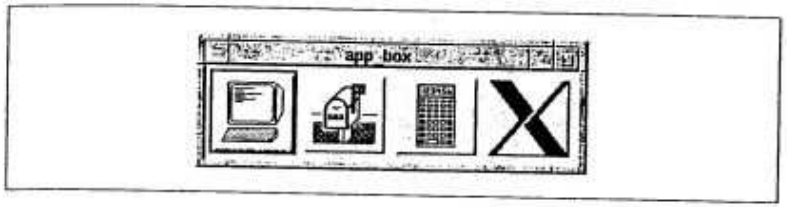
ArrowButton



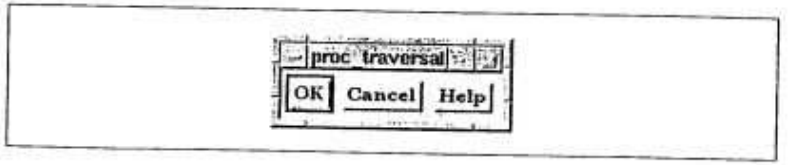
CascadeButton



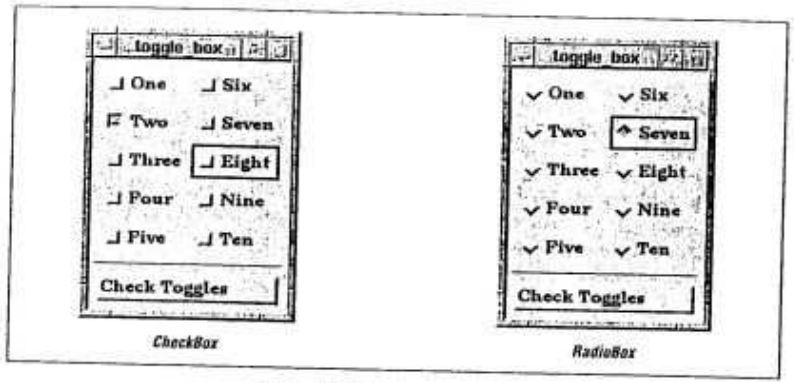
DrawnButton



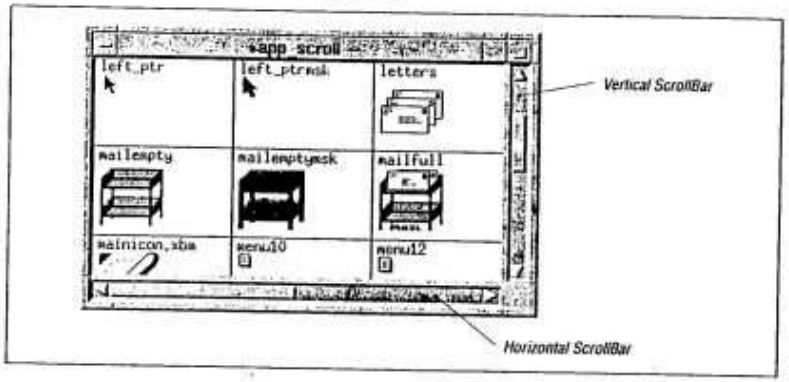
PushButton



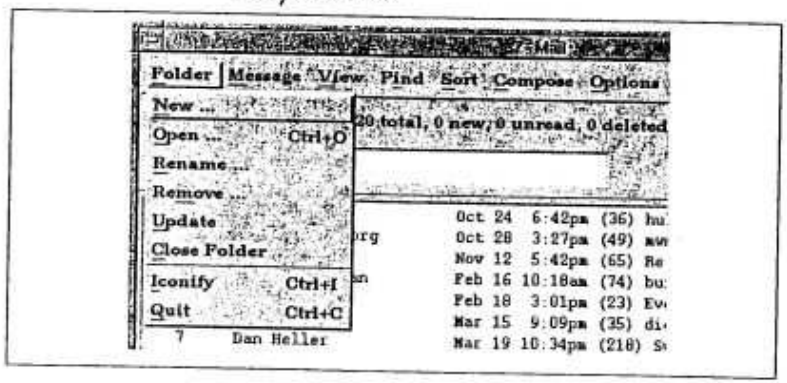
ToggleButton



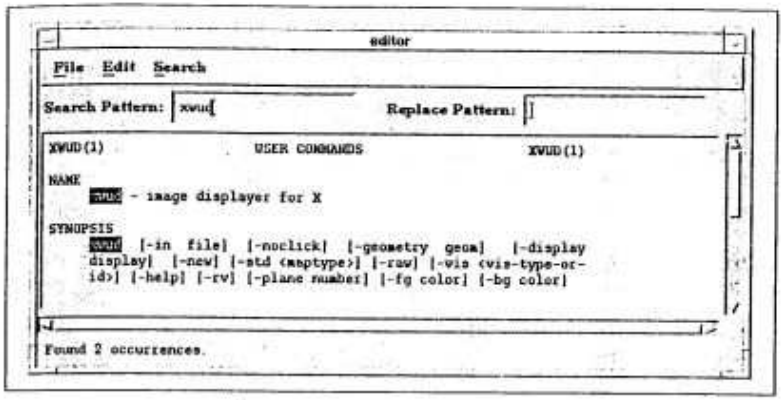
ScrollBars



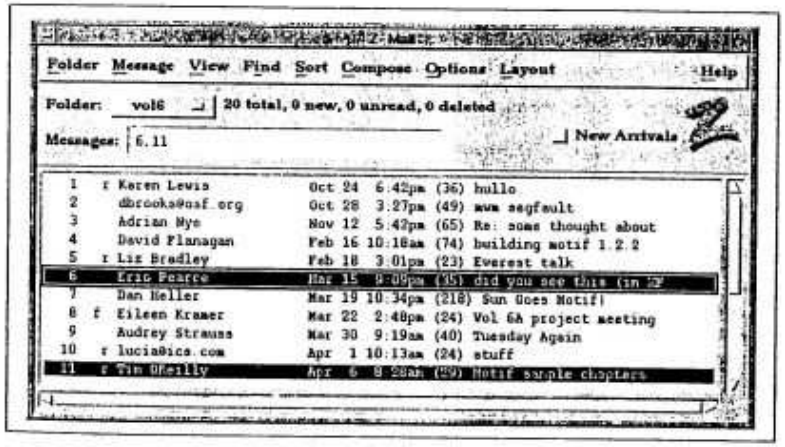
Separator



Text, TextField



List



• Motif (omejeni osnovni gradniki)

Razredi podob v Xt in Motif:

- Omejeni osnovni gradniki (*gadgets*) - nimajo naslednikov, so identični osnovnim gradnikom a z omejenimi resursi:
  1. Barve in prikazovalno matriko dedujejo od svojega predhodnika;
  2. Nadzor nad dogodki ima predhodnik.
  - Puščica (**ArrowButton**) - gumb
  - Oznaka (**LabelGadget**) - tekst, ikona
    * **PushButtonGadget** - gumb
    * **CascadeButtonGadget** - kaskadni gumb
    * **ToggleButtonGadget** - stikalo
  - Separator (**SeparatorGadget**) - ločuje gradnike vmesnika

• Motif (oznake)

Oznake (*Label*) so namenjene prikazu tekstov in ikon.

```
#include <Xm/LabelG.h>
#include <Xm/RowColumn.h>
...
main(argc, argv)
int argc;
char *argv[ ];
{
Pixel fg, bg;
...
XtVaGetValues(rowcol, XmNforeground, &fg,
                      XmNbackground, &bg, NULL);
while (*++argv) {
        Pixmap pixmap = XmGetPixmap(
                XtScreen(rowcol),*argv, fg, bg);
        XtVaCreateManagedWidget(*argv,
                xmLabelGadgetClass, rowcol,
                XmNlabelType, XmPIXMAP,   /* XmSTRING */
                XmNlabelPixmap, pixmap,
                NULL);
}
...
```

/usr/openwin/bin/bitmap

• Motif (stikala (1))

Stikala (*ToggleButton*) so gumbi s pridruženim tekstom ali ikono.

```
#include <Xm/ToggleB.h>
#include <Xm/RowColumn.h>
...
Pixmap on, off;
Pixel fg, bg;
...
rowcol = XtVaCreateWidget("rowcol",
        xmRowColumnWidgetClass, toplevel,
        XmNorientation, XmHORIZONTAL,
        /* XmNradioBehavior, True, */
        NULL);
...
XtVaGetValues(rowcol,
        XmNforeground, &fg, XmNbackground, &bg, NULL);
on = XmGetPixmap (XtScreen(rowcol), "switch_on", fg, bg);
off = XmGetPixmap (XtScreen(rowcol), "switch_off", fg, bg);
...
```

```
#define switch_on_width 16
#define switch_on_height 16
static char switch_on_bits[] = {
   0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x18, 0x00, 0x3c,
   0x00, 0x1e, 0x00, 0x0f, 0x80, 0x07, 0xc0, 0x03, 0xff, 0xff, 0xff, 0xff,
   0xff, 0xff, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00};

#define switch_off_width 16
#define switch_off_height 16
static char switch_off_bits[] = {
   0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x18, 0x00, 0x3c, 0x00,
   0x78, 0x00, 0xf0, 0x00, 0xe0, 0x01, 0xc0, 0x03, 0xff, 0xff, 0xff, 0xff,
   0xff, 0xff, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00};
```


• Motif (stikala (2))

```
...
toggle = XtVaCreateManagedWidget ("toggle1",
        xmToggleButtonWidgetClass, rowcol,
        /* XmNset, True, */   /* Stikalo izbrano na zacetku */
        XmNlabelType, XmPIXMAP,
        XmNlabelPixmap, off,
        XmNselectPixmap, on,  /*Ikona, ko je stikalo izbrano*/
        NULL);
XtAddCallback (toggle, XmNvalueChangedCallback, toggled, NULL);
...
..............................."toggle2",
.....................................
...
        /* switch = XmStringCreateLocalized("switch"); */
                /* XmNlabelString, switch, */
...
...
void toggled(widget, client_data, call_data);
Widget widget;
XtPointer client_data;
XtPointer call_data;
{
    XmToggleButtonCallbackStruct *state =
            (XmToggleButtonCallbackStruct *) call_data;
    printf("%s: %s\n",
            XtName(widget), state->set ? "on" : "off");
}
```

• Motif (stikala (3))

Primer: Izpis izbire stikal

```
. . .
unsigned long toggles_set;
char *strings[ ] = {"One", "Two", "Three", "Four"};
. . .
rowcol = XtVaCreateManagedWidget("rowcolumn",
        xmRowColumnWidgetClass, toplevel, NULL);
. . .
toggle_box = XtVaCreateManagedWidget("togglebox",
        xmRowColumnWidgetClass, rowcol, NULL);
. . .
for (i=0; i < XtNumber(strings); i++) {
        w = XtVaCreateManagedWidget(strings[i],
        xmToggleButtonGadgetClass, toggle_box, NULL);
        XtAddCallback(w, XmNvalueChangedCallback, toggled, i);
}
. . .
w = XtVaCreateManagedWidget("Check Toggles",
        xmPushButtonGadgetClass, rowcol, NULL);
XtAddCallback(w, XmNactivateCallback, check_bits, NULL);
. . .
```


• Motif (stikala (4))

```
void toggled(widget, client_data, call_data);
Widget widget;
XtPointer client_data;
XtPointer call_data;
{
        int bit = (int) client_data;
        XmToggleButtonCallbackStruct *state =
                (XmToggleButtonCallbackStruct *) call_data;
        if (state->set)
                toggles_set  |= (1 << bit);
        else
                toggles_set &= ~(1 << bit);
}

void check_bits(widget, client_data, call_data);
Widget widget;
XtPointer client_data;
XtPointer call_data;
{
        int i;
        printf("Toggles set:");
        for (i=0; i < XtNumber(strings); i++)
                if (toggles_set & (1 <<i))
                        printf("%s", strings[i]);
}
```

• Motif (podoba lista (1))

Podoba lista (*List*) je namenjena prikazu liste tekstovnih možnosti in njih izbire.

Gradnja:

```
#include <Xm/List.h>
. . .
XtSetArg (arg[n], XmNvisibleItemCount, 5); n++;
list_widget = XmCreateScrolledList (rowcol,"scrolled_list",
                    args, n);
XtManageChild (list_widget);
```

Branje lastnosti:

```
int upper;
XmString *strlist;
. . .
XtVaGetValues (list_widget,
            XmNitemCount, &upper,
            XmNitems, &strlist,
            NULL);
```


• Motif (podoba lista (2))

Funkcije za delo z nizi v podobi lista:

- void XmListAddItem (list_w, item, item_position)
- Boolean XmListItemExists (list_w, item)
- void XmListReplaceItems (list_w, old_items, item_count, new_items)
- void XmListDeleteItem (list_w, item)

list_w	–	Podoba lista
item	–	Sestavljen niz
item_position	–	Indeks polja sestavljenih nizov
old_items	–	Kazalec na polje sestavljenih nizov
new_items	–	Kazalec na polje sestavljenih nizov
item_count	–	Število sestavljenih nizov polja na katerega kaže kazalec old_items

Odzivne lastnosti:

XmNdefaultActionCallback

Dva klika z levo tipko miške ali RETURN
Razlog za odziv: XmCR_DEFAULT_ACTION

XmListCallbackStruct

• Motif (podoba tekst (1))

Omogoča urejevanje teksta.

Ne omogoča:

- Istočasnega prikaza večih barv
- Istočasnega prikaza različnih oblik znakov
- Ni emulator terminala
- Ne more poganjati interaktivnih programov
- Ne more prikazovati multimedijskih objektov

Gradnja:

```
#include <Xm/Text.h>    /* <Xm/TextF.h> */
...
n = 0;
XtSetArg(args[n], XmNrows, 15); n++;
XtSetArg(args[n], XmNcolumns, 80); n++;
XtSetArg(args[n], XmNeditable, True); n++;
XtSetArg(args[n], XmNeditMode, XmMULTI_LINE_EDIT); n++;
XtSetArg(args[n], XmNcursorPositionVisible, True); n++;
text_widget = XmCreateScrolledText (rowcolumn,
                                    "text_widget", args, n);
...
```


• Motif (podoba tekst (2))

Nekatere **lastnosti** podobe tekst (*Text*):

- XmNmaxLength
  Maksimalno število znakov v podobi.
- XmNeditMode
  - XmSINGLE_LINE_EDIT - privzeto
  - XmMULTI_LINE_EDIT
- XmNeditable
  - True - privzeto
- XmNcursorPositionVisible
  - True - privzeto
- XmNrows
- XmNcolumns
- XmNwordWrap
  - True (podoba deli vrstice teksta pri belih presledkih)
  - False - privzeto

• Motif (podoba tekst (3))

Nekatere **lastnosti** podobe tekst (*Text*):

- XmNresizeWidth
  - True (število kolon podobe se ustrezno poveča, če tako zahteva nov dodan tekst)
  - False - privzeto (ta vrednost je stalna, če ima lastnost XmNwordWrap vrednost True)
- XmNresizeHeight
  - True (število vrstic podobe se ustrezno poveča, če tako zahteva nov dodan tekst)
  - False - privzeto (ta vrednost je stalna pri objektu ScrolledText)

• Motif (podoba tekst (4))

Funkcije za delo s tekstom v podobi tekst:

- char *XmTextGetString (text_w)
- void XmTextSetString (text_w, string)
- int XmTextGetCursorPosition (text_w)
- void XmTextSetCursorPosition (text_w, position)
- Boolean XmTextFindString (text_w, start, string, direction, found)
- void XmTextInsert (text_w, position, string)
- void XmTextReplace (text_w, from_pos, to_pos, string)
- char *XmTextGetSelection (text_w)
- Boolean XmTextGetSelectionPosition (text_w, left, right)

text_w	–	Podoba tekst
string	–	Tekstovni niz
position	–	Položaj v tekstu (0 - prvi znak)
start	–	Začetni položaj za iskanje
direction	–	Smer iskanja (XmTEXT_FORWARD, XmTEXT_BACKWARD)
found	–	Začetni položaj najdenega teksta
from_pos	–	Začetni položaj teksta v podobi
to_pos	–	Končni položaj teksta v podobi
left	–	Začetni položaj izbranega niza
right	–	Končni položaj izbranega niza

• Motif (podoba tekst (5))

Odzivne lastnosti:

- XmNactivateCallback
  Ob aktiviranju podobe.
  Veljavna samo, če je prisotna ena sama vrstica.
  (XmAnyCallbackStruct; XmCR_ACTIVATE)
- XmNmodifyVerifyCallback in XmNvalueChangedCallback
  Ob spreminjanju teksta.
  (XmTextVerifyCallbackStruct;
  XmCR_MODIFYING_TEXT_VALUE
  in XmAnyCallbackStruct; XmCR_VALUE_CHANGED)
- XmNmotionVerifyCallback
  Ob premikih kurzorja.
  (XmTextVerifyCallbackStruct;
  XmCR_MOVING_INSERT_CURSOR)

• Navodila (elementi za interakcijo preko zaslona (1))

Omogočajo uporabniku interakcijo z aplikacijo in manipulacijo nad podatki.

Sestavni deli elementov:

- Oznaka za identifikacijo
- Vsebina (podatki ali neka druga informacija)
- (Gumbi za manipulacijo)

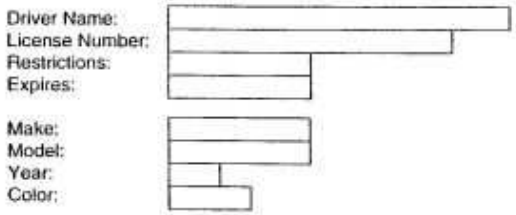
Vrste elementov:

1. Gumbi
2. Polja za vnos
3. Polja za izbiro
   3.a Stikala (ena možnost)
   3.b Stikala (več možnosti)
   3.c Palete
4. Liste
5. Izvlečne in dvižne liste
6. Kombinirana polja za vnos ali izbiro (krožna polja)
7. Kombinirane liste
8. Kombinirane izvlečne in dvižne liste
9. Drsniki za parametre aplikacije
10. Drsniki

• Navodila (elementi za interakcijo preko zaslona (2))

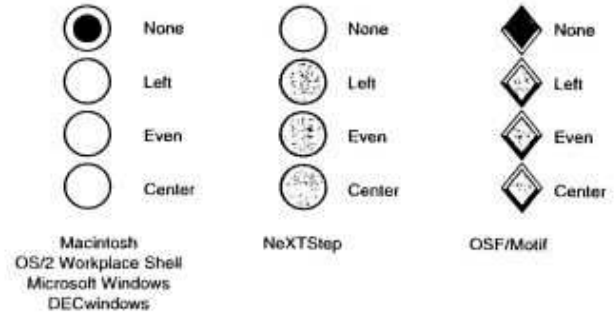
1. Gumbi:
Za ukaze, prikaz menujev, manipulacijo z oknom(i).
+: Vedno vidni, hitra akcije, razumljiv opis, 3D,
omogočajo ekvivalente preko tipkovnice.
−: Zasedajo veliko prostora, veliko premikov kurzorja.

2. Polja za vnos:
Vnos informacije preko tipkovnice.
+: Fleksibilnost, malo prostora.
−: Uporaba tipkovnice.

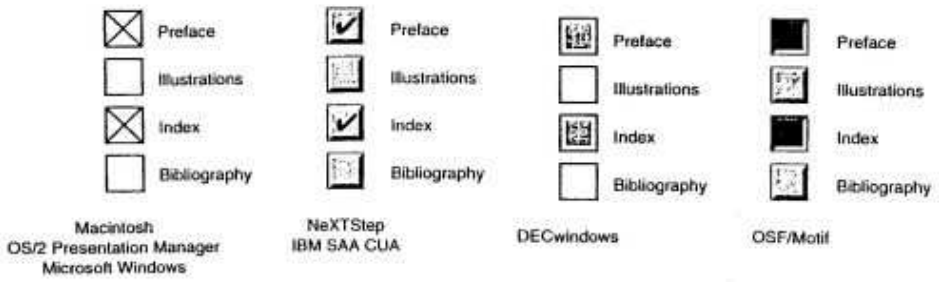


• Navodila (elementi za interakcijo preko zaslona (3))

3.a Polja za izbiro - stikala (ena možnost):
Postavljanje atributov, lastnosti.
+: Lahek dostop, lahka primerjava, razumljivost.
−: Zasedajo veliko prostora, omejeno število možnosti.



3.b Polja za izbiro - stikala (več možnosti):
Postavljanje enega ali več atributov.
+: Lahek dostop, lahka primerjava, razumljivost.
−: Zasedajo veliko prostora, omejeno število možnosti.



• Navodila (elementi za interakcijo preko zaslona (4))

3.c Palete:

Grafične alternative, ikone.

+: Prijetnost, lahka primerjava, manj prostora kot tekst.

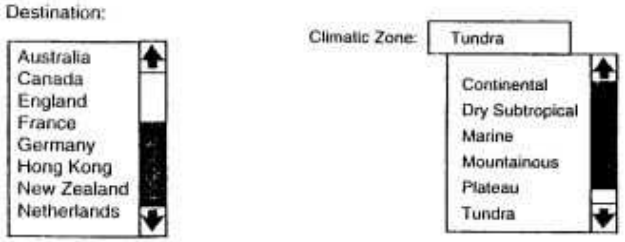
−: Potrebne izkušnje za atraktivno predst., omejeno število možnosti.

4. Liste, 7. Kombinirane liste:

Izbira ene ali več opcij, stalen prikaz, (vnos preko tipkovnice).

+: Neomejeno število opcij, "prepoznavanje", stalen prikaz.

−: Zasedajo veliko prostora, zamudno iskanje opcij.

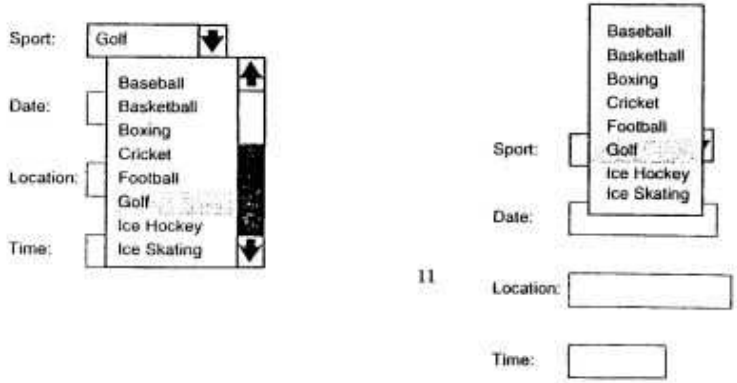


5. Izvlečne in dvižne liste,
8. Kombinirane izvlečne in dvižne liste:

Izbira ene opcije, prikaz na zahtevo, (vnos preko tipkovnice).

+: Neomejeno število opcij, "prepoznavanje", ne zasedajo prostor.

−: Dodatni korak za prikaz liste, zamudno iskanje opcij.



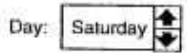
• Navodila (elementi za interakcijo preko zaslona (5))

6. Kombinirana polja za vnos ali izbiro (krožna polja):

Izbira ene opcije, možen vnos preko tipkovnice.

+: Zasedajo malo prostora, fleksibilnost.

−: Težko primerjati opcije, le določena vrsta podatkov.

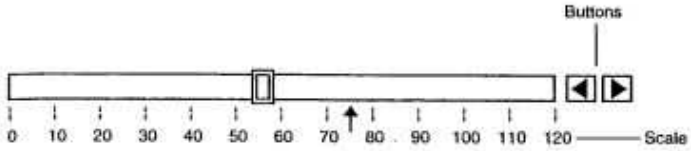


9. Drsniki za parametre aplikacije:

Omogoča zvezno in kvalitativno nastavitev vrednosti parametra.

+: Prostorska predstavitev relativne nastavitve, vizualnost.

−: Manjša preciznost od numerične nastavitve, veliko prostora.

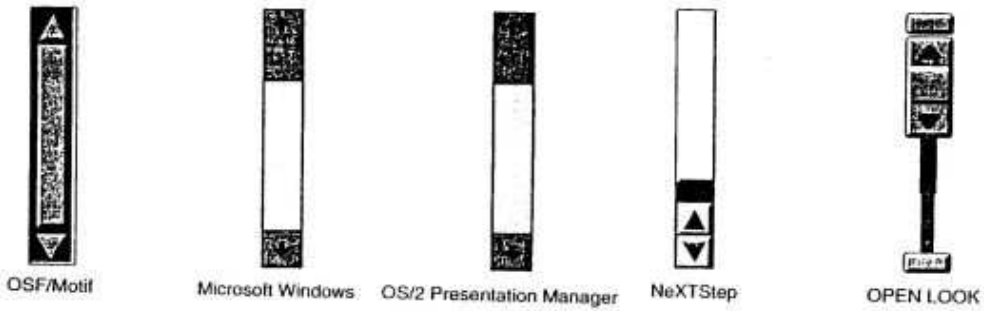


10. Drsniki:

Omogočajo prikaz informacije, ki zaseda več prostora od predvidenega.

+: Prikaz neomejene količine podatkov.

−: Prostor



• Navodila (gumbi ali menuji ?)

Aspekti:

- Število ukazov aplikacije
- Kompleksnost ukazov
- Frekvence (pogostost) ukazov
- Ali je dani ukaz povezan z nekim drugim ukazom ?

Ukazi:	**Izbira:**
7 ali več in obstoja hierarhija	Vrstični menu, menuji
6 ali manj in preprosta aplikacija	Gumbi v oknu
6 ali manj in pogosta uporaba	Gumbi v oknu
So uporabljeni z drugimi ukazi	Gumbi v dialogu
Pogosta uporaba, dve stanji	Stikala

• Navodila (polje za vnos ali polje za izbiro ?)

Ali so vhodni podatki omejeni po količini in vsebini ? NE -> Polje za vnos

↓ DA

Ali so podatki znani in razumljivi ? NE -> Polje za izbiro

↓ DA

Možne tipkarske napake ? DA -> Polje za izbiro

↓ NE

Tipkanje zahteva manj dela kot uporaba miške ? NE -> Polje za izbiro

↓ DA

Polje za vnos

• Navodila (kateri element za interakcijo preko zaslona ? (3))

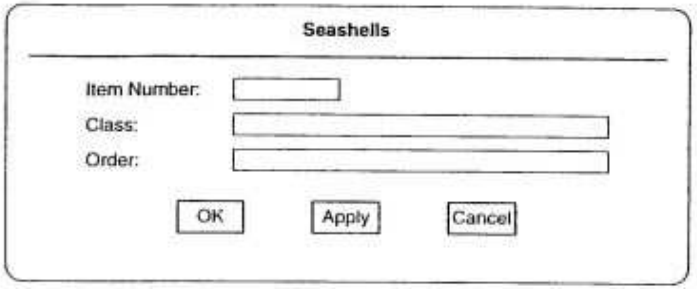
Razmere: **Izbira:**

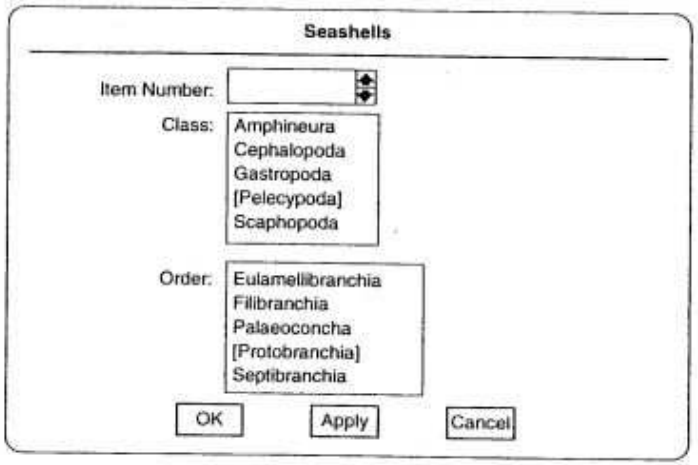
Medsebojno izključevanje alternativ
Diskretni podatki —> Paleta
Željena grafična predstavitev

Medsebojno izključevanje alternativ
Redke spremembe —> Krožno polje
Znani, napovedljivi podatki
Zaporedni podatki
Ni dovolj prostora

Medsebojno izključevanje alternativ
Zvezni podatki —> Drsnik za parameter
omejenega obsega
Napovedljive spremembe

• Navodila (elementi za interakcijo preko zaslona - primer načrtovanja)





• Navodila (aranžiranje elementov za interakcijo (1))

Pravilna predstavitev informacije in gradnikov v oknih je ključnega pomena za hitro in učinkovito interakcijo z aplikacijo.

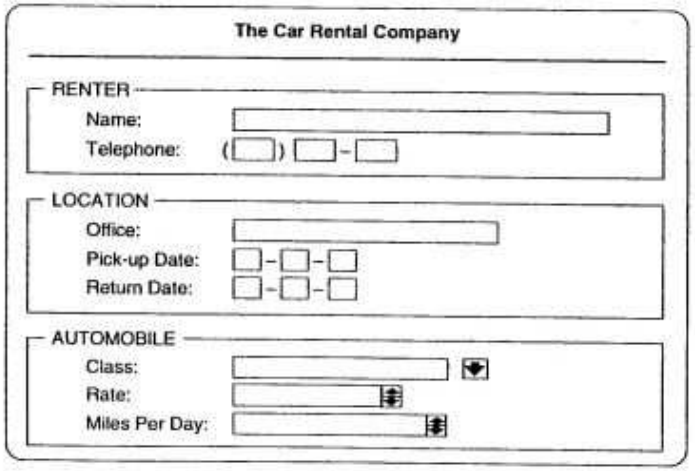
Aspekti aranžiranja elementov za interakcijo:

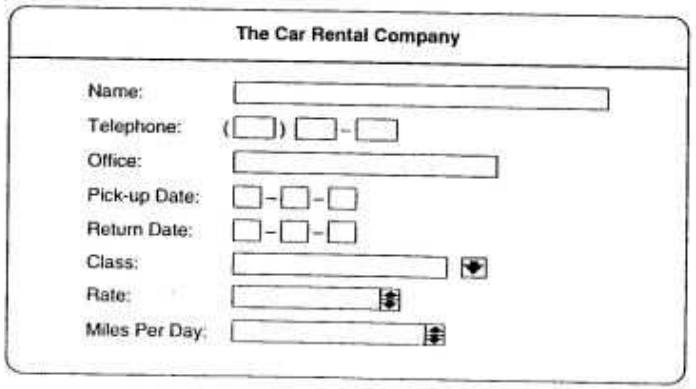
- Količina informacije (perceptualna ločljivost gradnikov, estetska prijetnost, lega glede na pogostost uporabe, navigacija)
- Logična organiziranost (organizacija gradnikov naj odraža organiziranost okolja realnega sveta, ki mu služi aplikacija)
- Velikost okna (eno okno, če je le mogoče)
- Balansiranje (enakomerna distribucija elementov)
- *Grupiranje*
- *Orientacija in poravnavanje*

• Navodila (aranžiranje elementov za interakcijo (2))

Grupiranje:

Glede na tip gradnika, kontekst interakcije ali kontekst podatkov; beli presledki; separatorji; okvirji; glave; pojasnila.

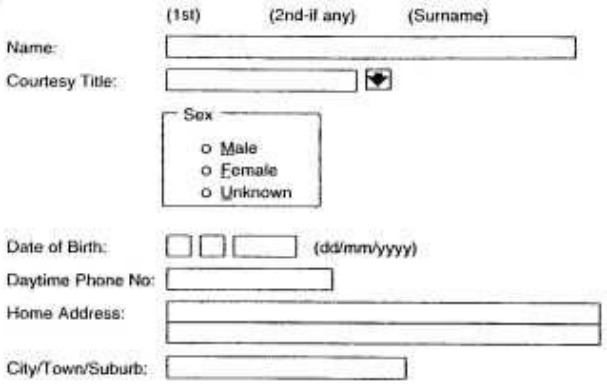




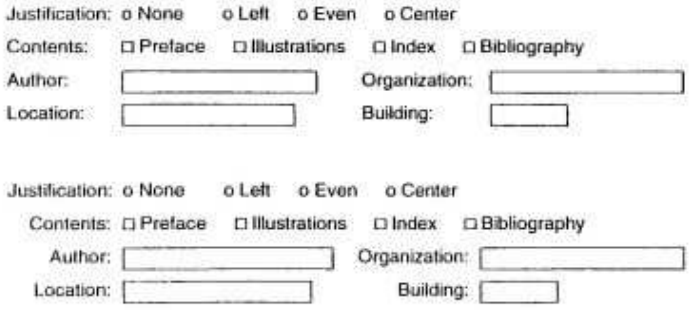
- **Navodila (aranžiranje elementov za interakcijo (3))**

Orientacija in poravnavanje:

- Vertikalna orientacija in vertikalno poravnavanje (polja in opcije levo, pojasnila levo ali desno poravnana)



- Horizontalna orientacija in vertikalno poravnavanje (polja in opcije levo, pojasnila levo ali desno poravnana)



- **Navodila (aranžiranje elementov za interakcijo - primer)**

Aspekti: količina informacije (ločljivost, estetika), logična organiziranost, velikost okna, balansiranje, grupiranje (kontekst podatkov, beli presledki, okvirji, glave, pojasnila), orientacija in poravnavanje.