1996-11-14 Bo Nordin (bosse&cb.uu.se)

Lecture notes, computer graphics (F08) (X)

X overview

Industry standard for *2D* graphics applicationa under UNIX.
Network:

client <----------> server
application communication screen/user
several clients
graphics
complex structures (windows, cursors) -> shared resources

The server runs on a workstation or X terminal.

window manager: a client with increased authority (anyone can write one, X doesn't care)
is in charge of windows
layout policy, rules about size, positions, icons
olwn, mwm, twm, fvwm, 4Dwm (can run on another system than the server)
decides the 'look-and-feel' of the system
can be configured: focus policy, move opaque

X defines a *protoco*l. (Xext, PHIGS) PEX, GL
asynchronously 4 types of messages:

events	server->clients	are queued (input, moved window)
requests	client->server	are buffered (draw)
replies	server->client	
errors	server->client	

round-trip requests (inquiries)

Xlib is the library of X functions that the application uses: most functions correspond to exactly one X protocol request.

Distribution of responsibilities:
server: windows, fonts, cursor, colormaps (*resource*), interfaces to device drivers.
goal: minimise communication, hides hardware differences
window hierarchy, expose, etc.
may have backing store (images)
client: *properties* (client <-> client) app<->wm. hints: wm may ignore them.
window (pixmap, drawable) attributes: size, depth, visual, border colour, event mask, cursor, colourmap, GC

Software architecture:

Application	
UIMS	
Widget Set	Motif, Open Look
Xt Intrinsics	
Xlib	create windows, graphics, resp to events connect to a server

A sample session (page 6 in X11)

Program: connect to server (may be refused (authorization), DISPLAY, xhost +)
create window
set properties ->WM

select input
show the window
event loop

Events (3 types) (X, page 11-12)
from user action: Button Press, MotionNotify
from the system: Expose, ConfigureNotify
from other clients: ClientMessage, PropertyNotify
select events with XSelectInput

Xlib functions are called X...

----drawX-----

X graphics (X, page 7)
Only 2D.
Origin in the *upper* right corner, y increases downwards.
XDrawPoint, etc have few parameters -> reduces network traffic by reducing # parameters,
All graphics attributes are collected in a single data structure (*very* bundled):
can store many GC's in the server.
can change selected portions of the GC
Most components of the GC are attributes but a few are related to graphics write ops:
function, plane_mask, clip_mask.
Line attributes: cap_style, join_style (Miter (??) , Bevel(avfasning))
Area attributes: fill_style
Text attributes: only font (flexible)

Colour:
Visuals determine whether indirect or rgb is used.
TrueColor: RO RGB
DirectColor: 3 subfields
PseudoColor: indexed, allows dynamic alteration
StaticColor: RO (degenerate Pseudo)
StaticGray:
GrayScale

Can allocate colour cells on some visuals. Functions for allocation, defining

Images:

create XImage RGB - 32 bits, 8 bits (colour table)

Toolkits:
Xlib - don't use
Toolkit
Xt (Intrinsics) with functions for creating/using user-interface components called widgets:
menus, buttons, dialogue boxes, scrollbars, etc.
Widgets are used for everything except graphics!
Widget: ADT for UI component
X window + (gadget)
handles expose, reconfigure, etc.
handles input, button press
a UI is a hierarchy of widgets (p 14)

resources (configurability)
widget class
create instances of classe
widget op's: read,write, set resources, create, destroy.

Inheritance: Core -> 3 types:

Display	-> buttons, text fields, etc.	<- Primitive
Containers	-> Dialogue Boxes, Bulletin Board	<- Composite
Shell	-> for WM communication	<- Shell

New widget classes inherit attributes from parent (page 13):
example: Core -> Primitive -> Label -> PushButton
It is possible to write your own widgets (libraries exist)

XmDrawingArea,

Functions in Xt begin with the letters Xt... (explain Va)
Important functions:
XtVaCreateWidget(name, class, parent, ...)
XtDestroy (Widget)
XtManageChild() - hide/display
XtUnmanageChild()
XtRealizeWidget() - only toplevel widget
XtPopup() - for popup dialogues

XtVaAppInitialize(&appcontext, "PGM",)
XtAppMainLoop()

Resources:
Have names like XmNwidth, XmNheight
Can be hardcoded, is also loaded when XtVaAppInitialize is called (X, page 15).
Resources can be read XtVaGetValues() and set XtVaSetValues()

Resources in files:
PGM.widgetname.resourcename: (resourcename with XmN removed)

Callbacks:
Makes it possible to execute application code as a result of a user interface action.
Each widget class defines a number of events when a callback occurs (can be XEvent)
Each callback can have a list of C functions that are called
When a callback occurs these are called by Xt.
XtAddCallback
Callbacks are resources: XmNactivateCallback (is an action) but cannot be set in the resource database.
Data sent to callbacks: widget, client_data, call_data
XmNuserData defined for all widgets.

Actions & translations:
Actions are functions that generate callbacks (you can write your own)
XtAddAppActions()
Translations controls which events trigger an action
----drawMotif.c----

UI Builders and Motif++:
----drawMotif.C-----

Include files & compiler options:

<X11/Xlib.h>
<X11/Intrinsic.h>
<Xm/Xm.h>
<Xm/BulletinB.h>

Compiler options: -lXm -lXt -lX11

Exercise:
Purpose: learn about input / interface design (menu design, construction techniques) & splines

HCI: (if time allows)

When computers first arrived: expensive, human labour cheap. Now it's quite the opposite, so there is a need for well-designed HCI's.
Cf. the automobile: we don't need understand how it works inside to drive it.

Complexityof systems -> potential for poorly designed HCI's is very high

USER (input language) <-> PROTOCOL <-> (output language) COMPUTER

visibility - controls must be visible, good mapping to their effects
affordance - their design should also suggest (afford) their functionality
usability - easy to learn, easy to use

health & productivity (mouse & keyboard should be supported, commands often used together should be close together)

GUI - graphical user interfaces
what's an interface:
input devices, output displays, dialogue structures,
use of colour, icons, commands, graphics, natural language
3D, user support material, multi-media

many disciplines: psychology, ergonomics, AI, linguistics, comp science, (social sciences)

high-level programming languages, toolkits, UIMS (user-interface-management-system), UIDE (user-interface-design-environments)

user-centred
integrate knowledge from different kinds of expertise
highly *iterative*

Coding:
alphanumeric - versatile, takes longer to locate (language-dependent)
colour (4-11) -segmentaiton, standing out, search task (colour pollution)
shape (10-20) (icons) -
size - large/small files

interface metaphors: desktop
scrollbar
menu
window
cut/paste

Consistency important: (Motif style guide)

BIT - basic interaction tasks
positioning - spatial , linguistic
selection - from a set of choices (fixed-size, variable-size)
scrolled lists
varying-sized (large)
a) naming, - wild carrs, don't care, autocompletion (DWIM)
b) pointing - pick
fixed-size (small)
single-level, hierarchical (->) cascading/replace (function keys - mnemonic)
radio buttons <- if a current selection is OK (pop-up with hilite on current)
implict/explicit pop-up menu (RasterOp) or static
(**# of items we can remember 7+= 2 (Miller, 1956) (working memory)**)

*tex*t - not commands
quantify - typing, setting a dial, up-down buttons, scale (linguistic, spatial)
3D - position & select harder (use 2D views) (rotate in AVS)

Should allow keyboard as well as mouse input for all BITs

CIT - composite interaction tasks
dialogue boxes (text attributes, parameters)

Contruction techniques:
rubber-banding (rectangle, circle) - active only while the button is held down
constraints
dynamic: click-and-drag

Window focus (click to focus, mouse focus)
multiple windows (iconification, tiling, overlapping)

Interaction-technique toolkits: Motif et al