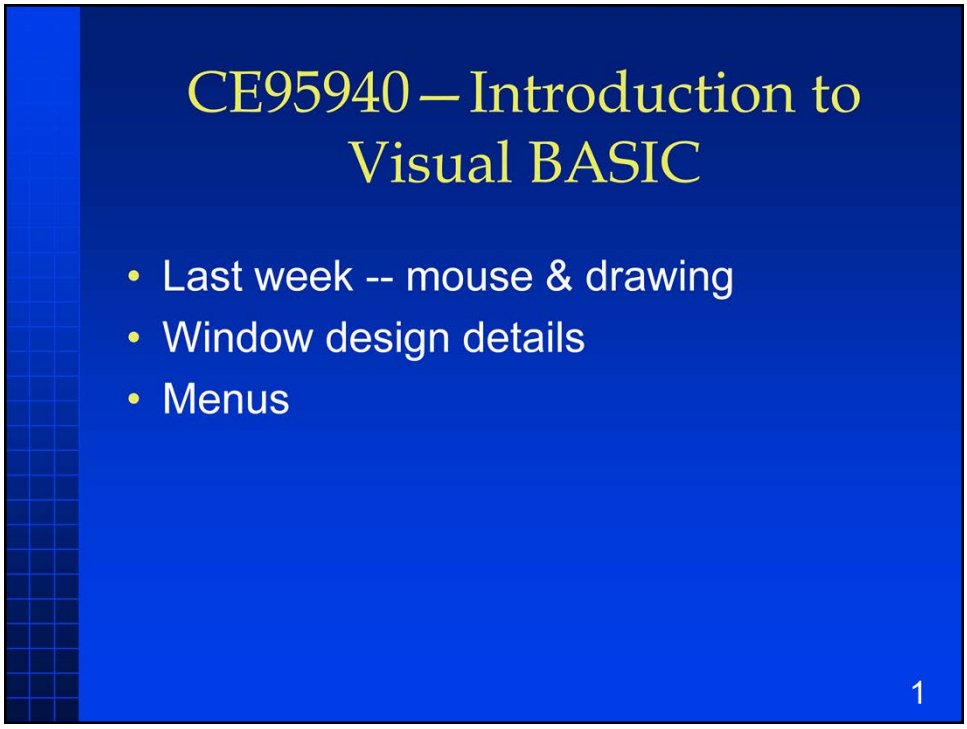
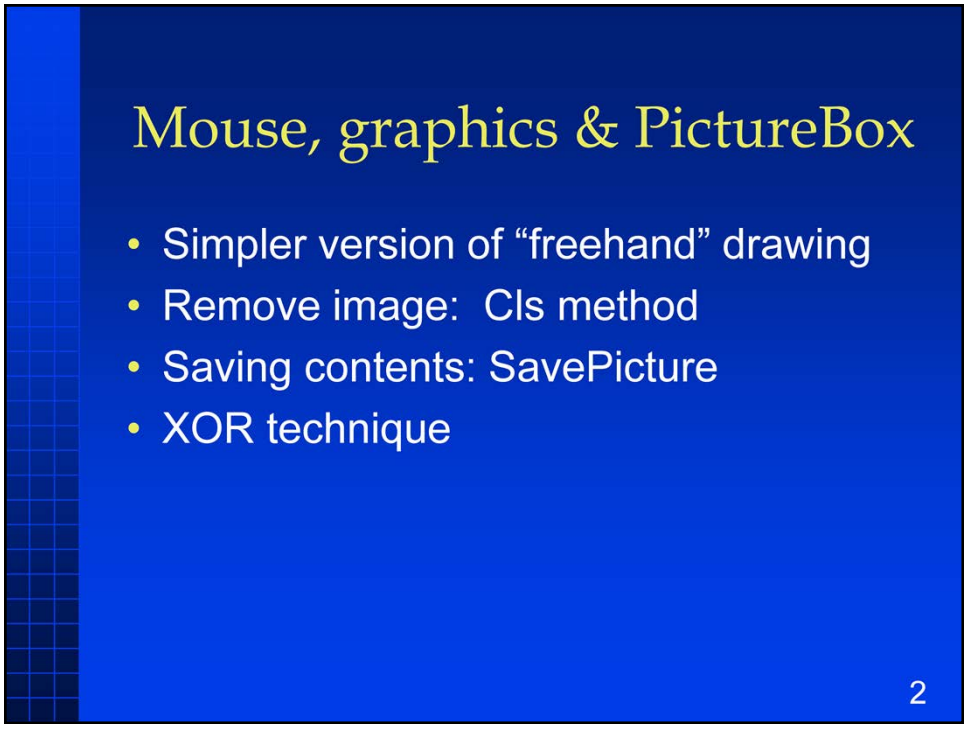


Review common questions and problems from last week



simpler version doesn’t need initial position and “first” switch

cls method clears persistent bitmap (place where drawing occurs)

savepicture statement copies persistent bitmap to file: image property, not picture property (run-time only)

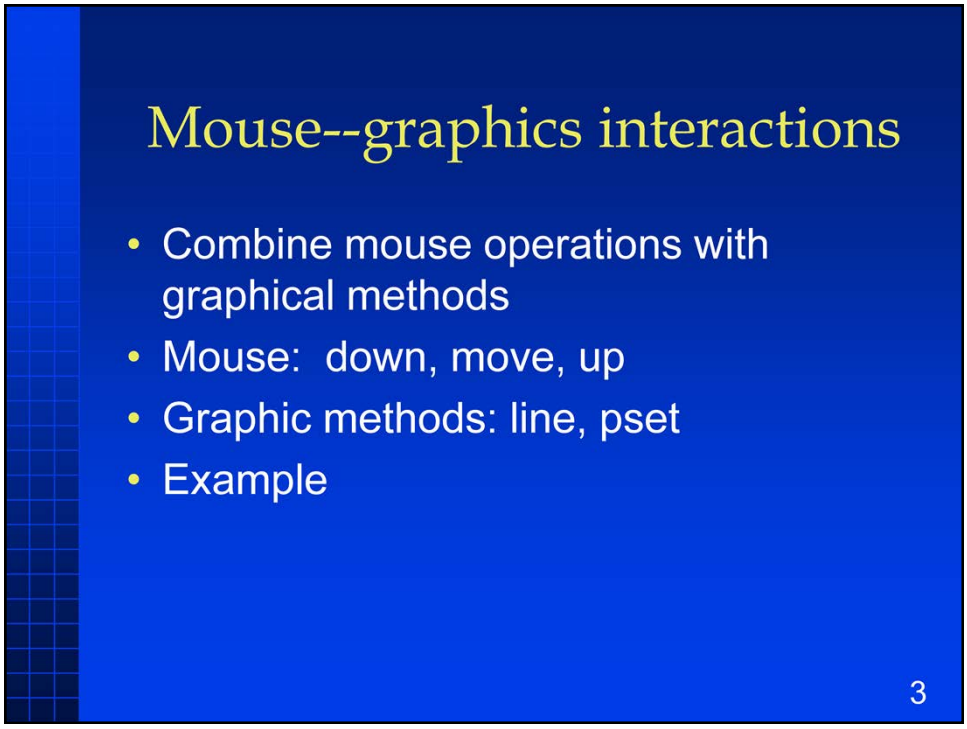
demo program 5.1, look at down, move; clear; save

XOR stuff: basic idea is:

draw a line, next time, erase it and draw a new one. XOR property means that drawing the same line twice (exactly same) restores existing state (background)

more details available

demo programs 2 & 3 cleaned-up versions



Demo progtam 5\1:

show form load, picture1 mousedown, up, move

colours: numbers 0-15, 0-255, 0-65536

fact about binary numbers: i x| C = i’ and then i’ x| C = i

example: 1001 x| 1111 = 0110, 0110 x| 1111 = 1001

1001 x| 0001 = 1000, 1000 x| 0001 = 1001

Drawmode: says how to draw, eg overwrite(copy), or how to blend background with drawing colour.

Usually copy, but xor mode xors bg+fg together and then draws line. repeating same operation restores original colour. Eg:

bg fg drawcolour

1111 0100 1011

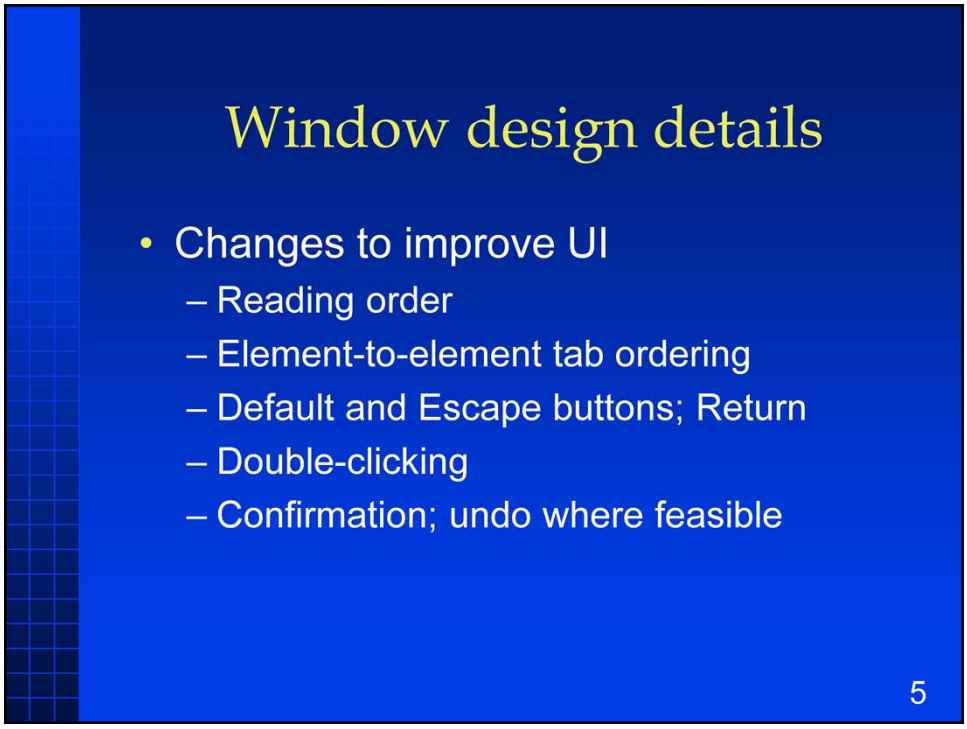
white red ??cyan

1011 0100 1111

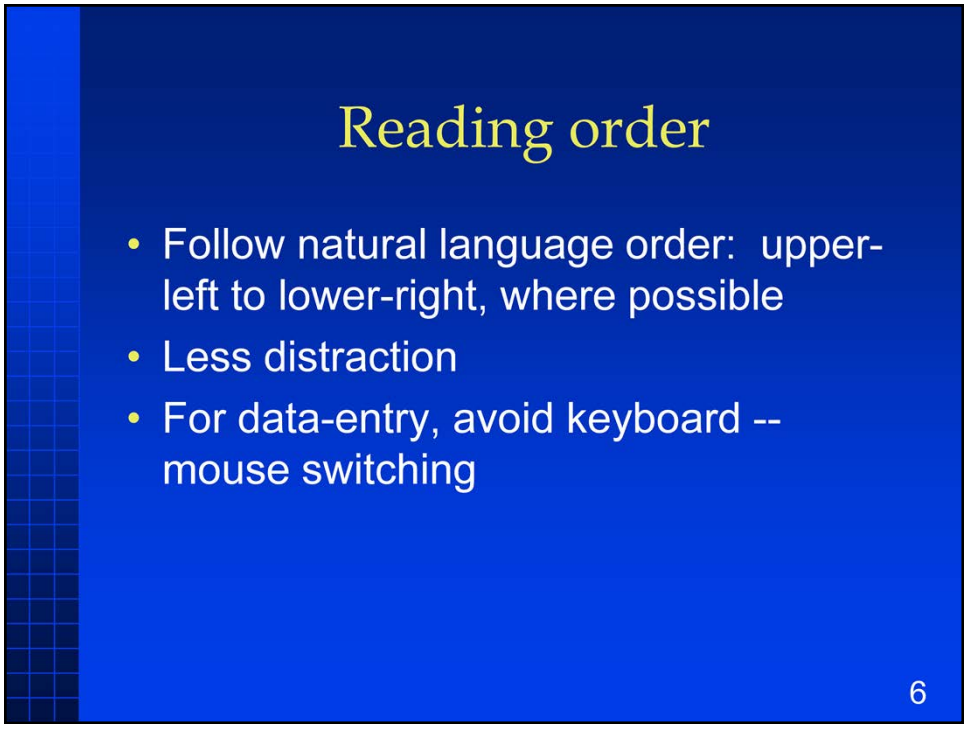
cyan red white

demo program 5\2, line stretching: show load, up, down, move; show consts.bas, try diff drawstyles

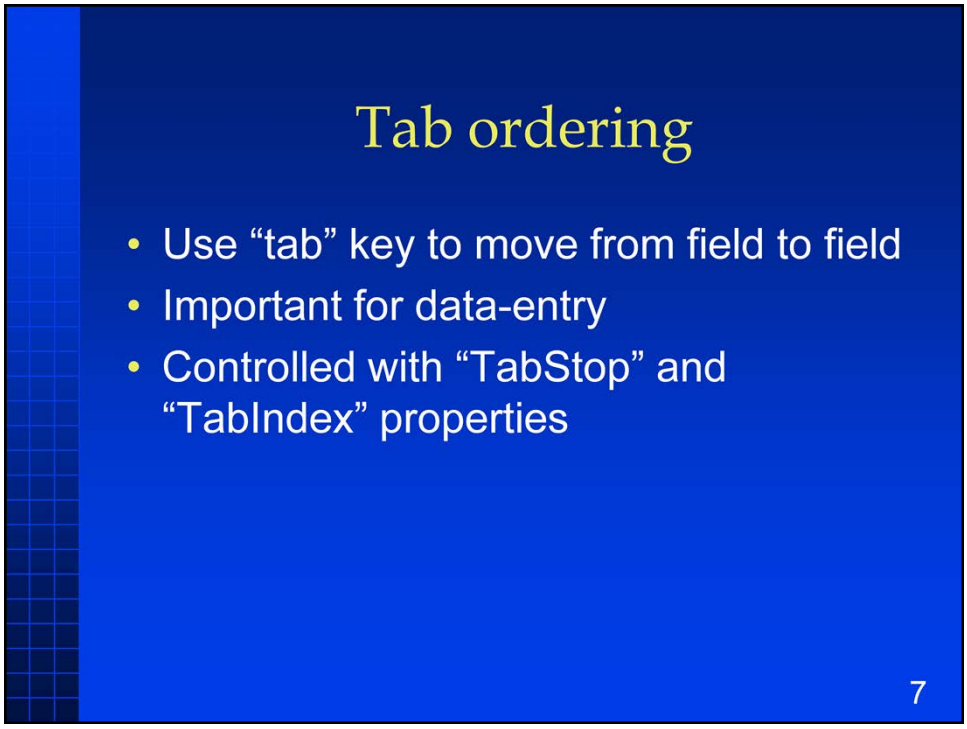
demo program 5\3, boxes



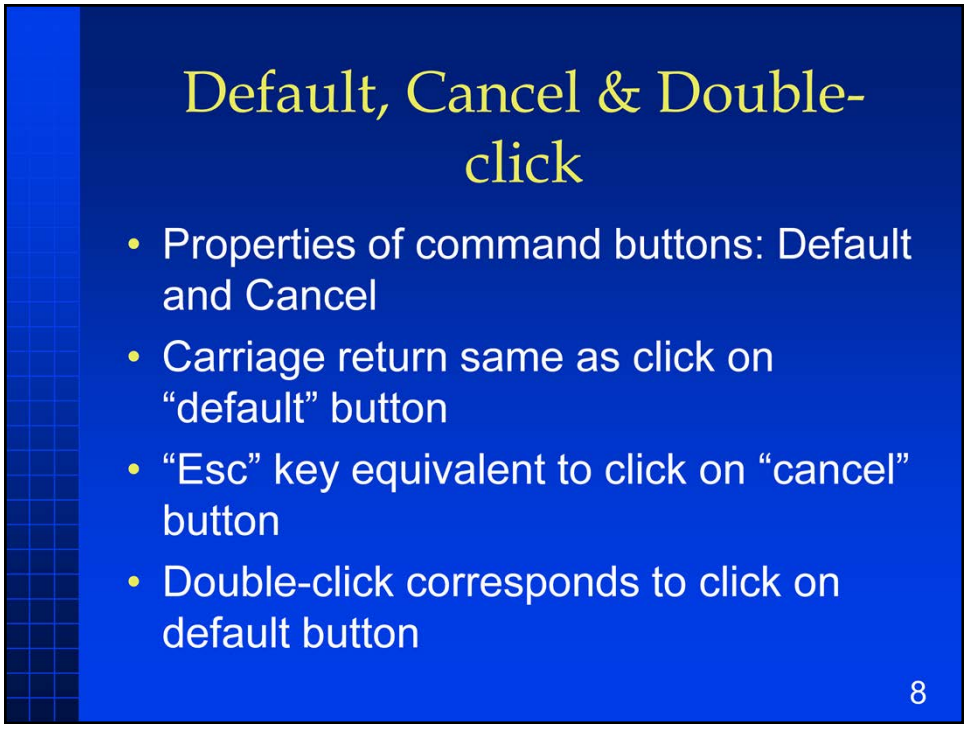
small things that can improve considerably UI effectiveness



where feasible



quickie demo

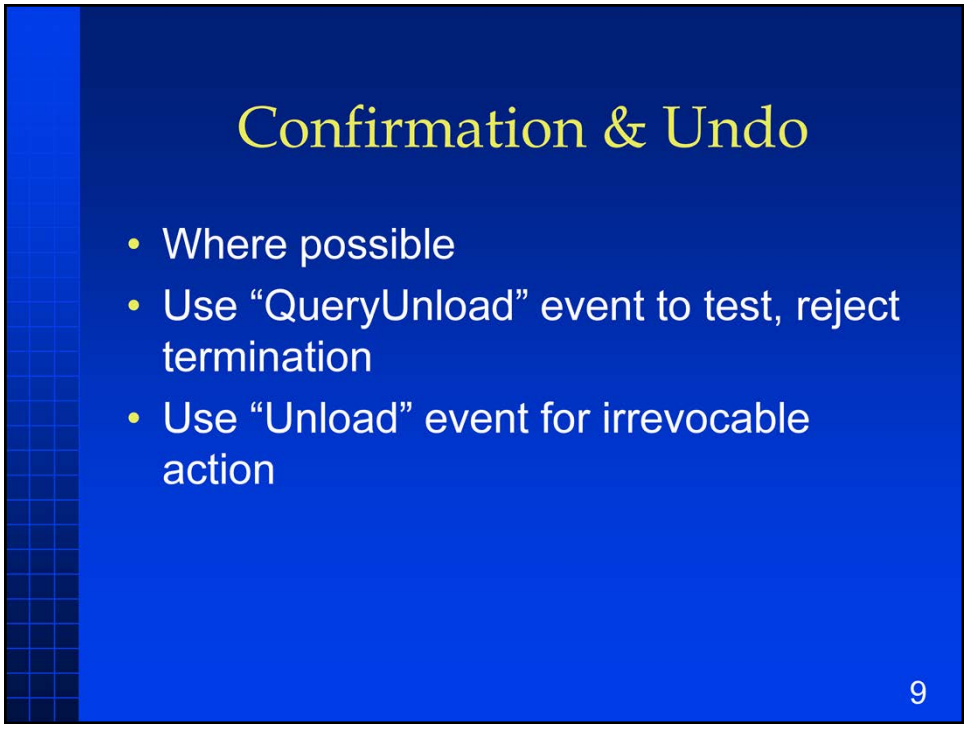


forms with "do-it" buttons, or where clearly-identifiable default

escape key for cancel action

double-click in lists: choose list item a press go (where go is default)

quickie demo; show button highlighting

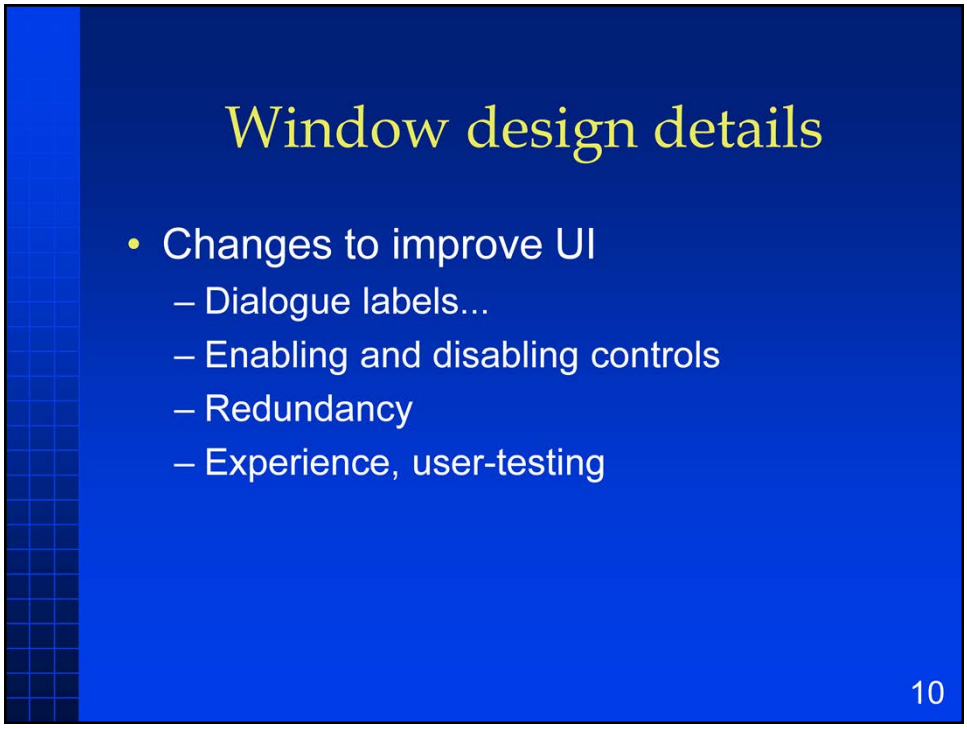


queryunload gives indication of reason for close: “close” button, unload method, Windows ending

can reject close request

use modal dialogues to to data modifications & updates; provide cancel mechanism from those forms

consider opposite: changes in-situ hard to undo unless values captured at appropriate times (hard to know)



"..." important visual clue that another choice -- converse is that no "..." means "do something"

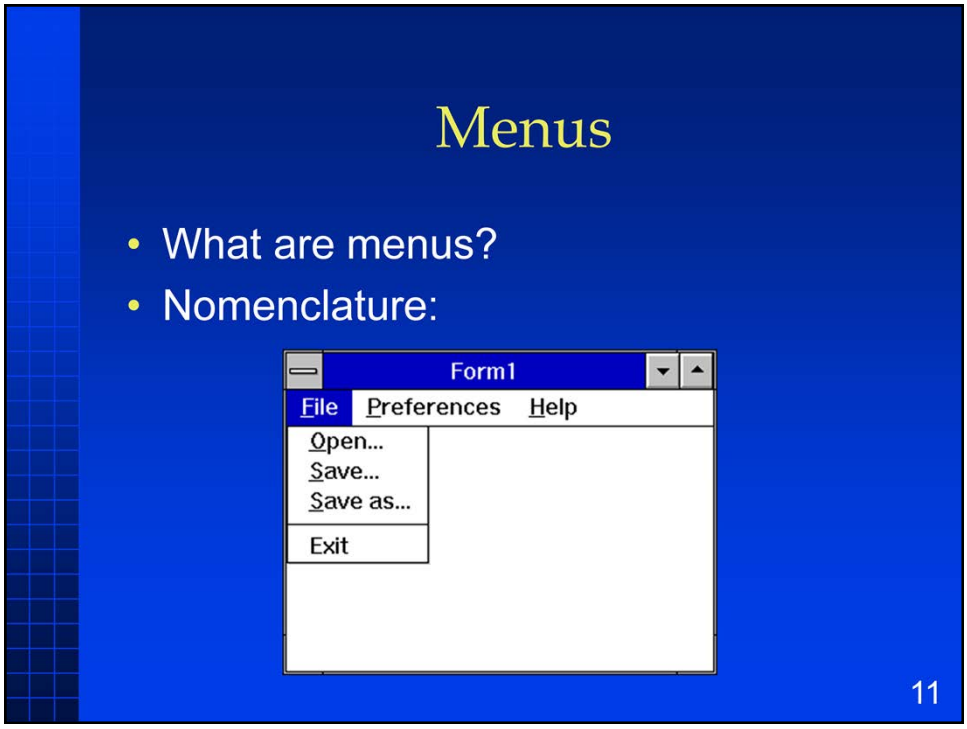
enabling and disabling can simplify user interactions (can't make mistakes if not permitted), can also reduce complexity of code (error prevention instead of error checking)

redundancy can be good and bad: confuses user or accommodates different styles or working:

E.g. no Close/exit button needed, since Close on system menu (unneeded redundancy in this case)

Only way to know for sure is usability testing (keystroke/mouse recorders, video cameras etc)

lots of other stuff: colour usage, # characters per line



Seen lots -- bar across the top of application form; also system menu

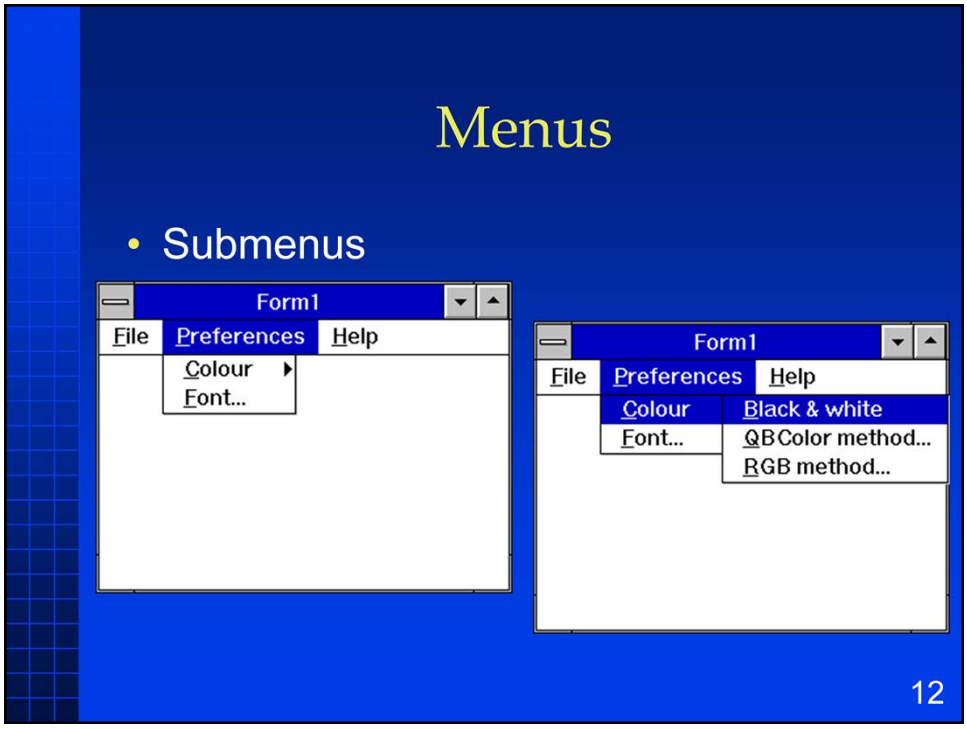
Nothing but command-buttons & checkboxes; can write code to simulate radio-buttons

Only interaction is “click”

Menu-bar consists of menus

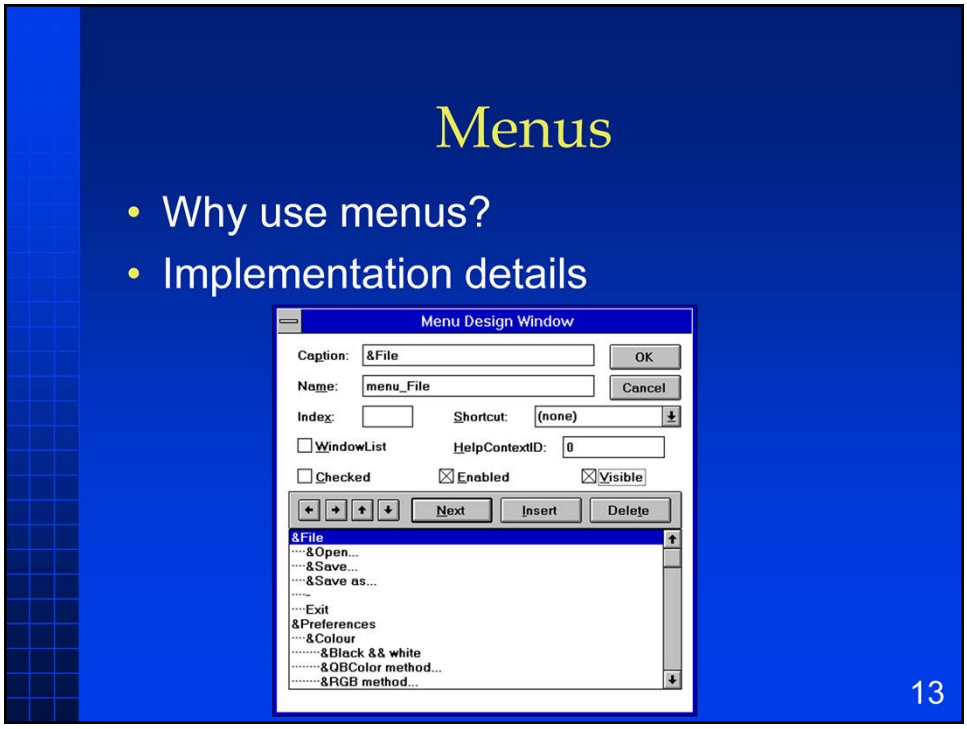
Each menu consists of commands (menu-items), separator bars, submenus

Commands can perform actions or present dialogues (as marked by ...)



Submenu indicator (arrow) automatic

Up to four levels



Lots of buttons -- cluttered

Menus reduce clutter, provide grouping of functionally-related controls

There is nothing that can be done with menus that cannot be done with buttons, and vice versa

Use Menu-design window (from VB Window menu)

caption of menu versus name of control (each item is a control)

& at front of name -- access key (more in a minute)

Visible and Enabled as usual

Checked usually used at run-time

HelpContext & WindowList to be ignored for now

Arrows to change nesting level or move up/down in list; Next/insert/delete as usual

start vb; demo lect5.4

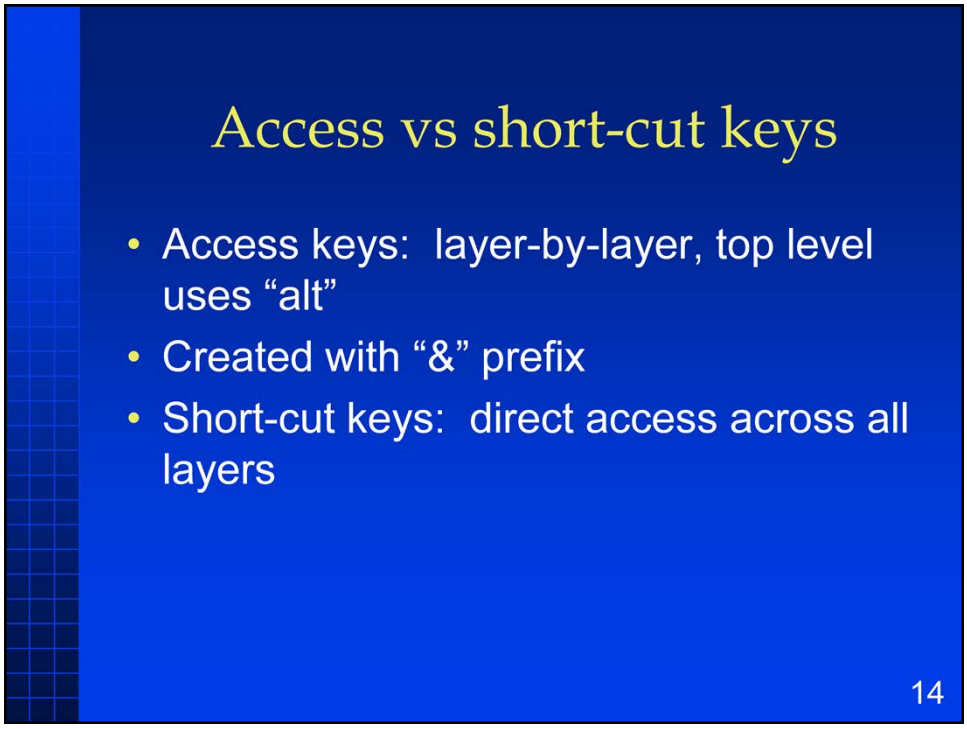
manipulate design-time: short-cuts etc

edit event code via click on menu-bar

show disabled & hidden; add some debug.prints

discuss use of standard dialogues etc

***don’t terminate demo**



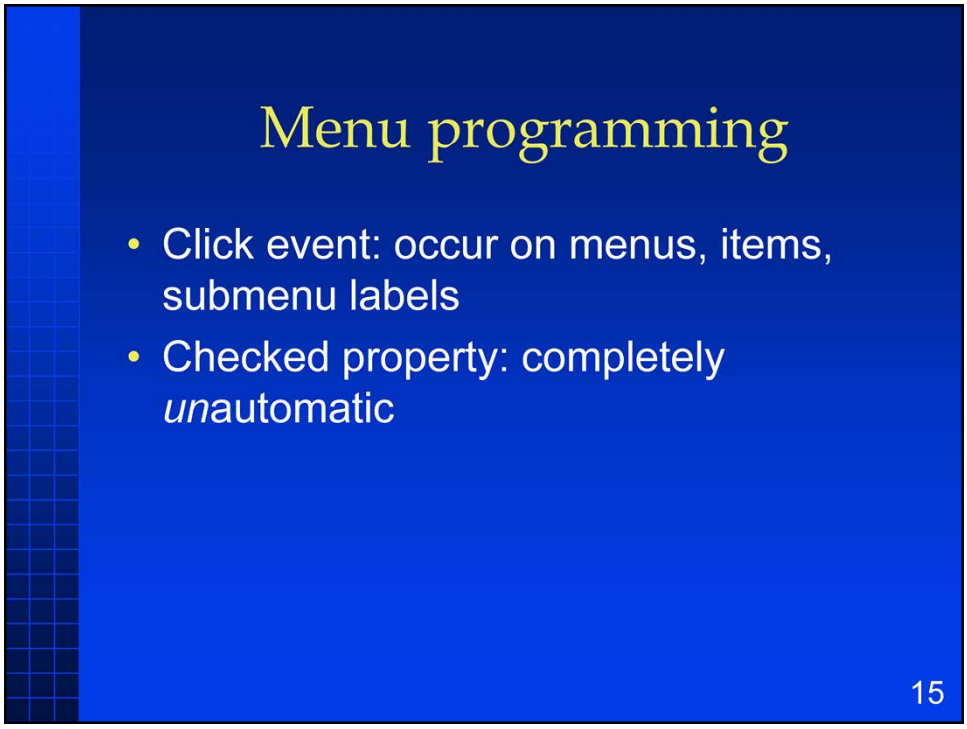
Developer is responsible for ensuring uniqueness

& prefix is displayed as underscored letter

need not be first letter -- can be anywhere

use two & to get one to show up

Shortcut assignments picked from list, appended to item caption automatically

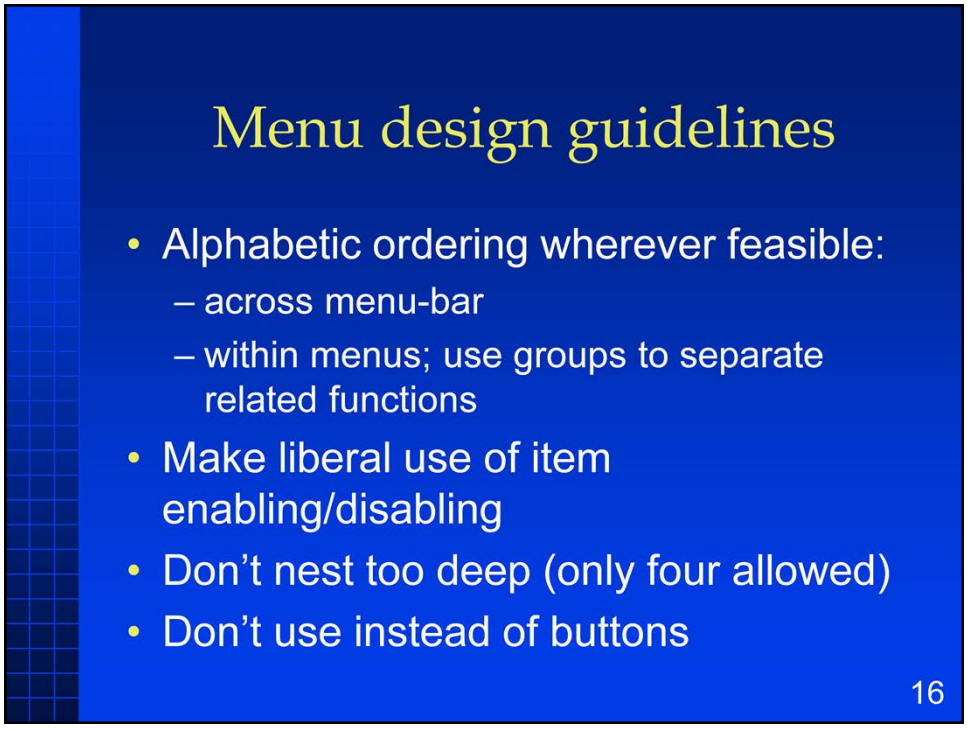


as noted, equivalent to command button, used in the same way

demo 5.4 again: edit top-level menu item via dropdown

use checked property to emulate a checkbox (true/false) eg toolbar item on VB View

can simulate radio-button -- write toggle code for check: (when click occurs: if checked, then uncheck, else find checked, uncheck it, then check original) (will be easier to do this using control arrays, discussed next week)



menubar ordering especially is subject to “everyone else” syndrome

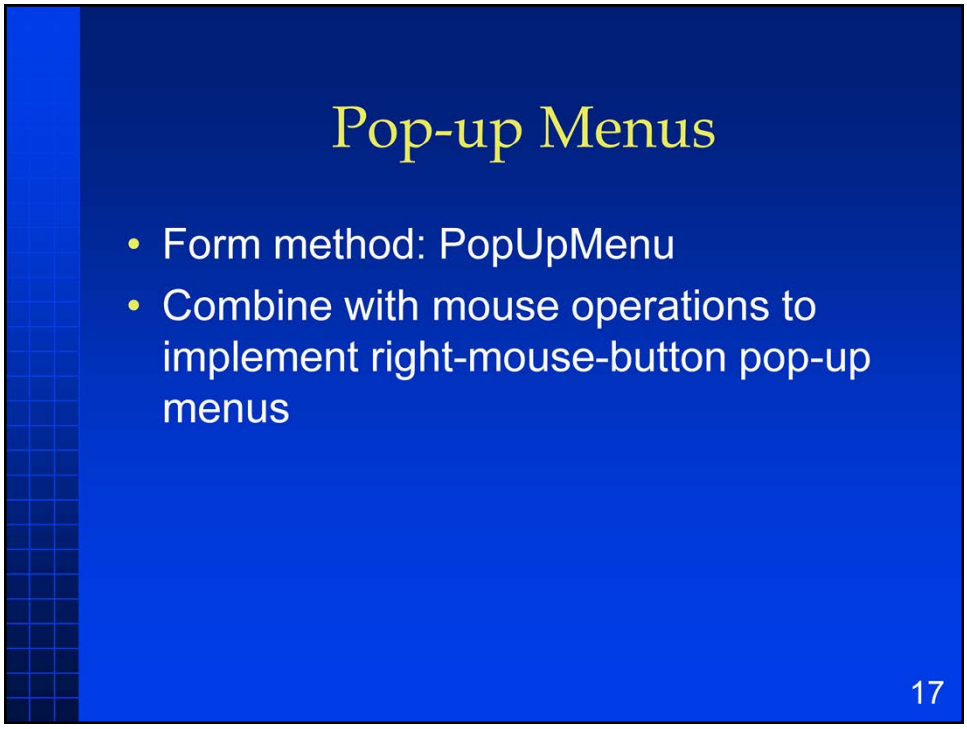
Vendors have their own style (eg File, Edit, View, ..., Window, Help), but since they don’t contain the same things, why bother? My opinion that more important to represent structure of application than adhere to dogmatic rules. Having said that, *if* going to have a menu that contains most of the same items as a MS file menu, call it “file”.

the bigger the menus, the more useful it is to enable/disable valid choices

Hiding entire menus may be more distracting that helpful

important to use “...” to indicate a dialogue (actually converse, items that do actions should be clear.)

“big” actions should use buttons -- don’t bury in a menu



form method, can be called any time, synchronous: once called, everything stops until popup is clicked or cancelled (if click, then menu click event called first, then resume) :: must be careful pgmming this

popupmenu ignores “visible” attribute of (top-level) menus; use this to build sets of menus to be popped up

PopupMenu method has optional args to set x-y position and alignment (centres over cursor, left side etc). default is current cursor x-y, left-aligned: below&right of mouse

By default, only a left click is recognized on the popped-up menu - can allow left or right.

<<simple demo to show syntax>> not

demo lect5.5

discuss mixture of click and button-events (Sequence: down, up, click, dblclick); clicks on & off form, etc

design-time is the same -- except not visible checked

how to edit event code? -- via dropdown (or make visible)

add debug.print code
add larger & smaller code