

1) Operational Goals

Implement the menu item maintenance, menu selection, and menu choice response mechanism which will work for all applications.

2) Fundamental, "key", or cornerstone architectural requirements (POSTMORTEM)

- Parse the menu selection into menu and menu Items.
- Send the parsed selection to the fTarget DEventHandler chain (linked list)
- Have the fTarget chain perform menu Item maintenance for the items they are involved with.
- Respond to command key events correctly.

3) Model of the implementation fulfilling these key requirements (POSTMORTEM)

- In the 1st cracker the Application sends a message to the fTarget chain to enable/disable their menu Items.
- The application then handles the menu select and extracting the menu and menu Item
- The application finally sends the HandleMenuChoice to the fTarget chain.
- The elements of the fTarget chain then needs to handle the choice and pass it on if appropriate.
- The mechanism responds to command key's by going through the above steps (sans the MenuSelect call.) starting from DApplication::KeyDown calling MenuKey.

4) Impact/scope of the implementation on the existing body of code (POSTMORTEM)

- DApplication :: MouseDown has to initiate the set up menu process in the fTarget chain.
- DApplication :: MouseDown has to handle the menu selection and parse out menu and item, then send the HandleMenuChoice message to the fTarget chain.
- The fTarget chain passes the HandleMenuChoice from fTarget to each of the DEventHandlers in the linked list, each passing the message on after its through.
- fTarget has 3 member functions to support Menu handling, SetUpMenus, HandleMenuChoice and EnableMenuItem.
- The command key behavior is implemented in the KeyDown DApplication method.
- The menu ID and menu Item numbers are defined in the bottom of the class declaration files. (the standard menu items are defined at the bottom of the DApplication declaration file)

5) Coding notes (gotchas, warnings, process thoughts, items to revisited later...)

There may well be occasions where one will not want the message to be passed on, and some thought in the order of calling "inherited::SetUpMenues() and inherited::MenuChoice(...)" within the respective method's definitions. The choice is between having the menu related messages go from the fTarget to the end or have the handling go from the end of the list to fTarget.

a) If one wants the messages to go through the chain from fTarget of the end of list (there by giving the last link the opportunity to over ride the desires of the earlier links) call the inherited methods at the bottom of your function definition.

- b) If one wants to have the messages to go through the chain from the end to fTarget call the inherited:: method at the top of your function definition.
- c) To have the passing on stop (perhaps to avoid having latter chain links overriding your wishes) have your class set gPassItOn to false before calling inherited method.

6) Testing notes(bug types, what made a bug hard to fix, what could have been done to catch it sooner....)

7) Process notes (what process did you follow, could it be improved)