Off-Line Documentation template: Activate Event Handling

1) Operational Goals
Just maintain fTarget.
(The window gets brought to the front is response to mouse events.)

2) Fundamental, "key", or cornerstone architectural requirements (POSTMORTEM)
• Need to associate the window getting the activate event with a DEventHandler object (it will turn out to be a DWindow subclass of DEventHandler).
• Need to make that associated DEventHandler the fTarget and send it an HandleActivate message.

3) Model of the implementation fulfilling these key requirements (POSTMORTEM)
a) Put a unique number in the window refCon's for all the windows created by the application (frame work).
b) Use the windowKind to determine if the window passed in from the event manager is a window created by the application (framework), if windowKind >= userKind (=8) then assume that the window is one that the application created.
c) Use the unique refCon to correctly set the fTarget to point to the correct place, and then give the object associated with that refCon a chance to do whatever necessary in response to the activate event.

4) Impact/scope of the implementation on the existing body of code (POSTMORTEM)
a) DWindow::Init sets the window's refCon to point to a reference to the instance of the DWindow object that created the window.
b) DApplication::ActivateEvt accesses the window pointer held in the event message field, it then checks the windowKind field of that window to see if it is >= userKind.
c) If the window is one that was created by the application framework (windowKind >= userKind) then DApplication::ActivateEvt sets fTarget to point to that window's refCon, otherwise it sets fTarget to point to the instance of the DApplication object. If fTarget is set to point to a window's refCon then it sends that object a HandleActivateEvt message to give the activating window a chance to do any maintenance needed when activate events occur.

5) Coding notes (gotchas, warnings, process thoughts, items to revisited later...)
• fTarget is maintained only in the Activate feature.
• Activate dose not bring the window to the front (that usually has already happened in the mouse down support)
• It may be necessary to put in a stronger checking of the window type in order to avoid dereferncing a bogus refCon. Things are working OK now, but as the CommToolBox demo develops this potential problem should get wrung out.
• The CommToolBox demo has not hit any problems associated with this scheme of handling activate events. The reason for this is that the windows the comm tool box creates are of dialogKind = 2 < userKind = 8.

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)