Position Paper
To the Workshop on Windowing Environments and
the Impact of Windowing and Graphics Standards
Copenhagen, November 1988

Jim Michener, mich@apollo.com
Jan Hardenbergh, jch@apollo.com
UUCP: [decwrl!decvax,mit-eddie,attunix)!apollo!{jch,mich)
Apollo Computer Inc.
330 Billerica Road
Chelmsford, MA 01824
22-NOV-88

David Gorgen & Tom Gross agree with the basic ideas behind these positions. Given time Apollo Computer as a company will have a position but the positions stated here are currently put forth by Jim and Jan as individuals.

This paper will attempt to make the case for the following positions.
1) The division of responsibility between Windows and Graphics Standards must be clearly drawn.
2) That the division of responsibilities between the window system and the graphics standard (PHIGS, GKS, CGI) cannot be clearly distinguished unless one also considers the user interface.
3) That primary functional areas addressed by each subsystem should be reasonably distinct and focused, it is too late to banish the overlap of functions but we can encourage a preferred use of of each subsystem.
4) A new set of standards or levels of current standards should be avoided. A minimum number of interface routines may be necessary and extension mechanisms should be encouraged to accomplish special needs beyond the minimum.

One of the most interesting areas of overlap between the Window system and the graphics standard is input handling. Currently, there is much excitement about "look and feel" user interfaces. Many different "look and feel" user interfaces are already and will continue to compete in the marketplace. Standardizing on only one is a bad idea. We believe that the interoperation of window systems and graphics standards should provide for the use of a user interface subsystem that would be the peer of the window system and the graphics system.

There are a couple of utilities that a graphics output system needs to expose to the user interface to replace the graphics input functions. These are coordinate mapping routines and picking routines. Other interactions might include handling refresh/resize events and flushing actions.

The major responsibilities of the subsystems are outlined below.

Window system
1) Manage screen real estate.
2) Capture and distribute device events to appropriate windows.
3) Generate expose/resize events
*) most window systems have a simple bitmap graphics output subsystem.
While this is expedient to provide window client with some graphics we think this should be thought of as a separate subsystem. The user interface system will also rely on this graphics output subsystem.

User Interlace
1) Echo user actions appropriate to given context.
2) Package user desires into application commands.

Geometric Graphics Subsystem (PHIGS, GKS as opposed to bitmap graphics)
1) Manage/Display application display list.
2) Pick or select from the display list.
3) Map between NDC & Window coordinates.

Unassigned responsibilities
There are at least two important functions yet to be assigned. While we feel these are extremely important we do not feel they can be addressed in the scope of our brief position paper.
*) Window management
This has been put in the window system, in the UI { Cognition)
and in a separate application.
*) Interapplication communication (Cut/Paste)

Extensibility
One desire that arises when input is talked about with PHIGS is to tightly couple input and output. Some examples are to hook up a track ball to a matrix or to provide an intelligent pick/echo feedback loop. While this desire widely agreed upon the exact implementation approach is not. Nor is there agreement as to the functions of a sufficient set of capabilities.
The current generation of graphics standards approach to extensions - implementation supplied - is proving inadequate. We would like to see simpler and more flexible graphics standards that would allow this to be done through application supplied extensions. Where application needs differ programmability should be provided.
More mundane issues
The following issues need to be solved more or less arbitrarily.
a) How does a window get associated with a "workstation"?
-create an interface that turns a windowID into a connection/workstation type. We think to fully specify screen real estate you need (type_flag, connectionID, windowID) triplet.
b) What should PHIGS/GKS return as the size of the "workstation"?
-provide a method to return maximum size, current size and exposed rectangles.
c) How should the potential interactions between graphics deferral states and events from the window system/UI be solved?
-UI or application needs to manage this. Define conditions in which information may not be correct or events are lost.
d) Does geometric graphics scale on a resize or just show more or less NDC space?
-should be application dependent.
e) Subsystems should be able to "grab" or lock shared resources. Especially, window sizes and clipping lists.