

**Aleph 87 - 1**  
**Note 87 - 1**

**Graphics Requirements Group :**  
Bowdery, Brandt, Cordier, Dreverman,  
Fernandez, Knobloch, Mermikides,  
Rolandi, Schlatter, Videau

edited by Henri Videau  
February 15, 1987

## **REQUIREMENTS FOR A GRAPHICS AND INTERACTIVE PROGRAMMING ENVIRONMENT**

### OUTLINE

#### INTRODUCTION

The goal of this note  
Where and how to use graphics  
Examples  
Formulation of the graphics question

#### BASIC REQUIREMENTS

Accessibility  
Ease of use  
Structure  
Speed  
Communication

#### FUNCTION REQUIREMENTS

- 1) Interactive programming
  - a) user interface
  - b) interaction with the data structure
  - c) interactive procedures
- 2) Graphics
  - a) how to represent the objects.
  - b) how to define and modify the aspect of the objects.
  - c) how to identify graphically an object.

#### CONCLUSIONS

#### APPENDIX

## INTRODUCTION

Now that we are on our way building the apparatus, making the processing efficient and the access to analysis easy remains a way to improve our chances to win the LEP contest.

Graphics is a natural and efficient way to exchange information between a human being and a computer. Therefore it has to be looked at as part of the computing environment. That is why we elaborate these requirements on the interactive environment trying to show where graphics enters: in the dialogue or in the graphical representation of objects.

### The goal of this note

The main aim of this paper is to describe the functions we would like to find in the interactive graphics environment.

This description should help to define the merits of the different hardware devices we may consider using.

It will show the impact of the choice of the standard graphics software package GKS 3D.

It should constitute a guide for the definition of specifications for the application modules.

### Where and how to use graphics

Graphics has to be discussed in the context of the interactive programming we envisage for monitoring, program development and tuning, scanning and analysis. The requirements for these different tasks are quite different. Even though the basic operations involved are identical the weight on each of them is different:

- Monitoring needs complete standard displays of the monitored part of the apparatus, with a fast throughput quite similar to the scanning of events and a good way of accumulating displayable statistics.
- Program development needs accurate and detailed plots of few events, showing all the intermediate steps of the algorithms.
- Program tuning requires the possibility of building and displaying histograms of sensitive results. Parameters may have to be modified in an interactive way and the algorithms rerun to observe the effect.
- Analysis needs a fast access to selected events, tools for building histograms and selecting from these histograms events to be displayed.
- Publications need high quality plots with titles and comments conveniently set up.

In order to help the understanding of the requirements we start with two examples of what you would wish to do in front of a graphics terminal during a program development or an analysis session.

### Examples

#### 1) Program development.

Looking at delta rays.

(we have a view of the TPC tracks for an event)

- you select a track by picking it on the screen.
- you ask for the display of properties of the track (the momentum and its error for instance) by typing the request or picking the property in a list. This information is displayed in a window on the screen or on a nearby alphanumeric terminal.
- you then want to examine the associated space points and request them to be displayed.
- you zoom on a slightly misaligned point and select it.
- you ask for the display of the pad pulse height versus time (histogram) to appear

- on the screen.
- you dissociate this point from the track.
- you activate a new fit of the track.
- you ask for the display of the fitted track.
- and then
- you ask for the associated EC cluster to be displayed.
- you ask for an extrapolation of the fitted track to the EC.
- you zoom on the cluster looking at the impact point.
- .....

## 2) Analysis.

- you work on a file containing a selected DST with low track multiplicity events ( $\leq 2$ ) and large missing energy ( $\geq E_{\text{beam}}$ ).
- for a first look at  $\gamma \nu \nu$  candidates and the related background you define a first selection by the following conditions: no tracks in TPC, HC energy  $\leq 2$  Gev, more than half the EC energy in one cluster in one endcap. This produces a working set you want to explore.
- you ask for the recording of your actions to follow.
  - you select two standard views where the two EC end caps are shown with the corresponding part of the TPC and ITC together with LC and LT. They appear side by side on the screen.
  - you select the gamma-cluster.
  - you ask for its energy and angle to be displayed.
  - you ask for a scatter plot of these quantities to be started.
  - ...
- you stop the recording of actions.
- you select the next event.
- you replay the set of recorded actions.
- .. you do so for few events then
- ask the process to run through the stored events without the displays.
- ask to display the histogram.
- pick on the histogram a bin you want to look at
- ask for the display of the corresponding events by replaying, on this events, the recorded action.
- ....

### Formulation of the graphics question.

We can distinguish three elements in the use of graphics:

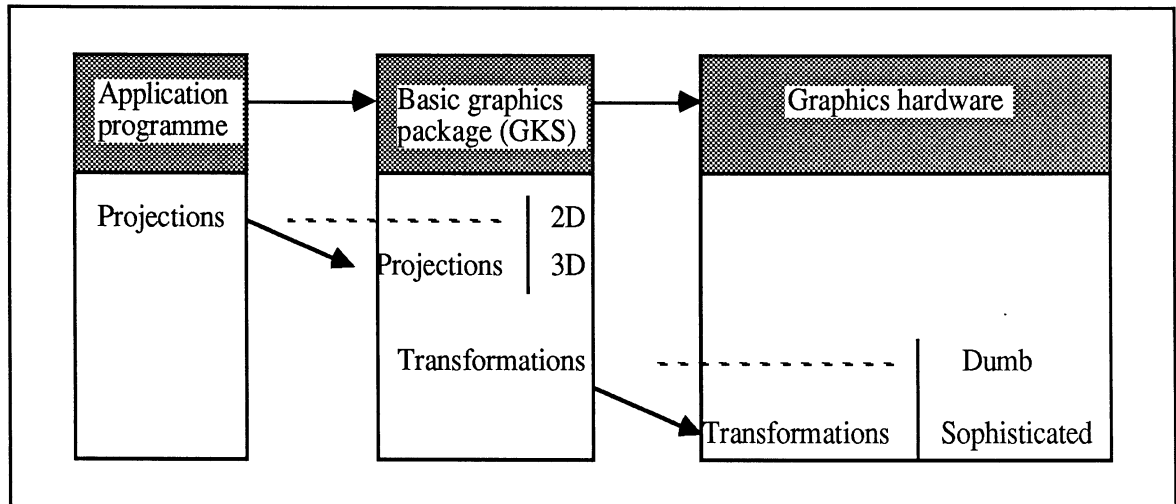
- the graphics hardware: terminals or workstations.
- the basic graphics package which interfaces the hardware.
- the application program which is written by Aleph.

The borders between the three elements are not strictly defined.

The objects we are dealing with in event viewing have 3 dimensions, the screen only two, we have thus to make "projections". If these are just projective transformations they can be done in a general procedure that may appear in the basic graphics package rather than in the application program, it enables also a 3D device to do (part of) it by hardware. Special transformations remain with the application.

Sophisticated devices store locally the graphics objects and some transformations can be handled at the hardware level rather than in the basic graphics package: zooming, scaling, panning for a 2D device, a 3D device is capable, in addition, to perform rotations in space or projective transformations which, depending on the performances, can appear as a quasi continuous motion.

This is illustrated by the following diagram:



To make the best use of the hardware capabilities the basic graphics package has to perform just what is not done by the hardware. It has to provide the adequate interfaces: the drivers which are specific of the device.

For a given level of functionality the application program Aleph has to write must compensate all the weaknesses of the basic graphics package and of the hardware.

Our main constraint is that quite a lot of different hardware devices exist in the Aleph laboratories ranging from ordinary terminals to fancy workstations. We recognize that institutions and countries have their own policies. Therefore a complete standardization on hardware is impossible. We should nevertheless try to reduce the diversity for future acquisitions.

## THE BASIC REQUIREMENTS

### Accessibility.

The analysis framework with its graphics component has to be accessible to the whole collaboration. This means that it will be written in such a way that the 2D devices will be, like it has been in past experiments, adequate for analysis .

The possibility to take advantage of the hardware capabilities has to be preserved.

In consequence we need:

An adequate graphics data structure for 3D objects.

A standard basic graphics package allowing for 3D. Aleph has decided to follow the recommendation from the CCC and use the standard Graphics Kernel System 3D.

Thus the projective transformations will be handled by GKS, the special ones in the application programme.

People having only a 2D terminal but GKS 3D will then have the complete graphics functionality except for the possibility of continuous motions. It is likely to be slower.

For people having only a 2D GKS, a kernel of 2D plots (see Appendix) will be available as an option in the environment.

### Ease of use.

We give a high priority to the ease of learning and using this environment. The accessibility to non experts is considered more important than the sophistication. This implies that the frame of the environment is very robust, having a lot of built in safety against data corruption and user mistakes.

### Structure.

To achieve this goal we need a good level of structuring in the data (see later) and in the software modules. The data we access have to be described in the agreed data model and the operations on the data should be clearly defined along the guidelines of SASD to ensure modularity and correct interface.

On the user side and for the same argument we want standard interface routines providing basic information into fortran arrays to anyone even totally ignorant of the underlying structure in such a way that he can play his own game in analysis.

We should have only one package with one user interface even though different options will exist like running on GKS 3D or GKS 2D (in that case the 3D possibilities should appear as disabled), running on a dumb terminal or on a workstation.

### Speed.

Concerning the speed of the operations we stress the following:

It is for the most common and repeated operations that we need most speed.

One does not want to wait idle. This arises when an operation implies erasing most of the information and creating a new picture, in particular when you redraw a picture or go to a new event, zoom or displace.

By "fast" we mean of the order of one second. The performance will be dramatically dependent on the device; a sophisticated graphics device will be suited for fast transformations, a multiprocess workstation for flexible organisation of the work.

### Communication.

It is very important to have a fast connection between the device you use for display and your computing power.

We consider that there is no clear need for exporting pictures in the way of metafiles but that rather we would like the possibility to replay on another site the same set of operations to get the same picture. For that we need an exportable journal file (see later).

## FUNCTION REQUIREMENTS.

From the examples above we see that we want to be able to perform a lot of different functions. To help the discussion we will class them distinguishing what concerns the interactive programming and what is properly graphics :

### 1) Interactive programming

#### a) user interface

We wish to put a large effort in this part so that we ensure the ease of learning and using this environment:

We require HELP procedures to appear on call or following mistakes, this should be accessible from all the possible modes of operation.

We want also a way to TEACH people by proposing standard operations done by steps and with explanations of their actions .

Each operation should have defaults such that standard actions could be performed straight away.

We should have the possibility to UNDO the last action taken, and to have some resetting points where you can go back if you are lost. By default this reset point would be the start of the play.

It is clear that in the learning phase a graphical mode of dialogue with menus and parameter panels is most commandable, nevertheless for an expert user we want a fast command style. This can be achieved by using a line mode of entering the commands with a possibility of abbreviation, synonyms, short names and compound commands. This last option gives the possibility to play a predefined set of commands with the attached parameters. We call such a set a macro.

These macros would be created either by using an editor or by recording an actual sequence of commands being done (equivalent to a log or journal file.) This sequence could then be reenacted at wish just by calling its name. That is a straightforward way to provide the user, in particular the beginner, with standard procedures. A recorded macro could be edited for modification. Creation of macros should be possible in line mode or in graphical mode. The macros would be executed in one go or by steps enabling to modify each

command before it is executed. Breakpoints allowing for user input should be provided.

b) interaction with the data structure

For the DST and the reconstruction programme (JULIA) the collaboration has chosen to describe the data with a model of the Entity Relationship model class. To present it simply we can say that the objects having the same type of properties are described by the lines of a table where each column is a property. These objects are related and the representation of the relationship done in a well defined way.<sup>(1)</sup>

The physics data and the detector description appear currently in this way.

The model being defined, you want to

- display the types of objects and relationships appearing in the model,
- display the actual objects and their relationships,
- move from one object to another via their relationship,
- modify properties of objects and relationships between them,
- all that interactively.

Designating objects.

You want to be able to designate an object or a collection of objects of the same type. This will be possible by naming them explicitly, conditionally, or relationally and by composing these possibilities<sup>(2)</sup>. Instead of naming it, if the objects are currently displayed on the screen we can directly point at them and pick them.

Showing their properties.

Consider that you are dealing with a set of objects, you want to know the properties which are attached to this type of object, then display the actual values of these properties corresponding to that particular set or make a selection on these properties (conditionnal designation)<sup>(3)</sup>.

Showing the related objects.

You want to know what are the types of objects related to the one you are considering, and what actual objects are related. You want to select objects by selecting the ones they are related to.

To continue the exploration of the structure we move the focus to a related object and do the same operations<sup>(4)</sup>. We call navigation the way of passing from object to object via their relationships.

We want to be able to change the value of some properties. This is clear for graphical aspect properties like colour of the object but has to be possible for others.

We want also to modify the relationships between objects.(eg: detach a space point from a track or attach one).

These modifications can be done one by one by hand or by means of interactives procedures.

(1) It will be clear with an example:

In the TPC space points (objects) are characterized by their position and deposited energy (properties).

These points will group to form tracks (objects) with a momentum (property).

There is a relationship between tracks and space points:

tracks (object) contain (relationship) space points (object).

Vertices (objects) are related to tracks, electromagnetic clusters (objects) are related to tracks, etc...

(2) You can say: track # 3, tracks with momentum > 4.56 GeV or tracks belonging to vertex #2.

(3) For example to the object type "track" corresponds the property "momentum", to the track#3 correspond the actual momentum components.

(4) Move from a track to the corresponding space points or to the vertex and get their coordinates.

c) interactive procedures

The possibility of activating a procedure on a selected set of data is one of the most powerful aspects of an interactive graphics system.

Such procedures will modify properties, create objects as well as relationships of known types, described previously in the model <sup>(5)</sup>. It will not be possible to create interactively new types of structures.

The procedures will commonly be procedures coming from the event reconstruction; the interactive frame will enable them to be run easily with different sets of parameters. Some will be new versions of procedures using different algorithms. This will be very helpful at the start of the experiment for program tuning .

There will be standard procedures for the analysis of the data but most will be user procedures introducing specific algorithms.

A list of standard procedures should be put together <sup>(6)</sup>.

Quite different types of procedures have to be considered, those relative to a set instead of one event. It concerns the browsing of log files, the examination of histograms. This has not been looked at carefully enough for the time being.

Even though we recognise its difficulty we would appreciate the possibility of relating a bin in an histogram to the histogrammed events.

2) Graphicsa) How to represent the objects.

The following concerns the detector description, the raw data, the objects appearing in the DST and in the (simulation and?) event reconstruction.

All the geometrical information concerning the objects described in our model should have standard graphical translation in agreement with the level of detail expected in the view (eg: a calorimeter storey may be shown as a dot or as a truncated pyramid ).

The detail level is related to the scale of the drawing. It is impossible to represent the full detail at the most general level.

The graphical translation of an object is generally not unique, it is chosen by the user.

There will be options to superimpose on the representation of the data the following:

- the contour of the corresponding detector piece or some important piece of material (like the TPC legs in front of the end-cap calorimeter),
- the grid formed by the read-out units ( calorimeters),
- the system of axis,
- the rotation centre or axis, or the translation vector when such operations are performed.

The information will be presented in independent windows which can be changed in size and position.

We want the possibility to draw on the same screen more than one picture, each one in a window.

---

(5) As an example consider the fitting of a set of TPC coordinates to a track. The points may be selected individually or be taken from a previously recognized track, using the relationship of tracks to hits. The routine will access the coordinates as the properties of the selected points, will perform the fit and return the resulting vector as a track property which can then be used and displayed. The user may store the track in the data structure as a new object or replace the previous one for propagation to other levels during the interactive session. This last possibility has to be used very carefully because the new object may have different relationships!

(6) A typical list for the central tracking detectors would include:  
 Compute coordinate for TPC pad cluster  
 Perform helix fit to set of TPC space coordinates  
 Perform circle fit to set of ITC coordinates  
 Perform combined fit to selected ITC and TPC tracks  
 Extrapolate TPC tracks to any region: vertex, calorimeters, etc..

This can be either different views of the same object <sup>(7)</sup> or different objects <sup>(8)</sup>, in the first case the aspect (in particular colour) should be the same for the different views to make evident the correspondance between the objects.

We remark that many of the plots of interest are not straight projective views of the objects for example R/phi view or shape of a TPC island.., or need special distortions for clarity .

It is likely that diverse possibilities will be developed freely. Nevertheless we have to ensure the existence of a standard core:

set of standard projections for common work (see the appendix),  
set of defaults for possible views, in particular standard viewing parameters for 3D objects.

#### b) How to define and modify the "aspect" of the objects.

An object has a certain "aspect" defined at the time of the representation definition. By "aspect" we mean the nature of the text symbols, their size, the colour chosen for the object, its brightness or the fact that it is currently visible or not. We want to be able to modify the aspect of a particular object or of a class of objects selected conditionally or relationally <sup>(9)</sup>.

We act also on the appearance of the objects by transformations done directly at the graphical level. According to the device used this will be done by the hardware or not.

We distinguish the following transformations: panning, zooming, scaling, rotating the object or changing the point of sight. More elaborate transformations will be done in the application.

Panning: This is a displacement of the object parallel to the screen. We can define it either by giving the translation vector, by defining the new centre of the picture or by moving the object with the mouse.

Zooming: The zooming in or out needs a reduction factor and a centre. A convenient way to define a zooming in is by delimiting the region to be blown up by a rectangle. We would like a level of detail in the display adapted to the zooming factor.

Scaling: The difference with the previous transformation is that we have now two factors corresponding to the change of scale along the two axes. The method of definition with a rectangle is particularly suited.

Rotating: To define the rotation we specify an axis in 3D, a centre in 2D and the angle of rotation, the angle is replaced in the case of a continuous rotation by the angular speed.

Changing the viewpoint: This needs the position of the viewpoint and the direction of sight.

To formulate the transformation we need to give vectors, centres, factors, axes, windows. This would be done in three different manners: by typing values (coordinates), by picking points on the screen or by identifying points or directions belonging to objects (see next paragraph), for example a given digitization, a corner of a piece of detector, the direction of a track <sup>(10)</sup>.

Note that when we have different views of the same objects we want to be able to apply the transformations simultaneously or not simultaneously to the different views.

When a view has been built we want the possibility to add titles, frames, comments etc.. needed to generate a convenient picture for a talk or a publication.

We do not consider that some fancy features like surfacing, shadowing etc.. concern us really.

---

(7) eg: a general view and a blow up of the vertex region, or two orthogonal views

(8) eg: the two end-caps, or a view of the TPC and the histogram of the dE/dx of a track.

(9) "Tracks with momentum > 4 GeV in blue" or "tracks belonging to the vertex # 2 in green".

(10) We can put the eye at the interaction point and look along the track.



### c) How to identify graphically an object

We want a way to identify a graphical object on the screen and to get access to the corresponding object in the data structure. This operation is called PICK: We define a point on the screen by an arrow, a cursor, cross.... and an algorithm looks among the displayed objects for one close enough to be designated by the point or asks for complementary information in case of ambiguity.

To pick a collection of objects we can either pick them one after the other or envelope them in a closed domain, usually a rectangle.

Such a selection of objects has to be acknowledged by highlighting ( or blinking..) the selected objects.

We have at that time a correspondance to the object in the data structure and can change the coding of non geometrical information.

The data selection can be facilitated by graphical segmentation which reflects the underlying data structure. This segmentation can be done naturally along the structure of the detector or along the physical structure of the events. We would like to have a dynamical segmentation according to the level of detail we are considering.

## CONCLUSIONS

We can now try to draw some general conclusions about the implications on the desirable hardware and software features.

A good resolution on a large screen is necessary to support the density of information we may want ( multiwindow) even though fast zooming can palliate a bit the poor resolution if it comes only from the screen.

We have recognized the power of colour. It appears more when colour is used as a marker to identify objects and correlate views than as a mean to translate a continuous variable.

Multiwindowing is essential. We require the application to provide this feature for the devices which do not have it by themselves.

The gain of speed you get from devices with local graphics storage and processing is clear but we have seen that often we need speed when moving in the data structure more than in the graphics representation. Therefore powerful local intelligence or fast links are more important than dedicated graphics hardware. The possibility of multiprocessing from one screen is a clear advantage.

The environment will incorporate specific modules, modules from the reconstruction and free style user modules.

We have stressed the uniqueness of this frame and in particular of the user interface. Nevertheless the possibility of casting in the same mould versions for dumb terminals and multiprocess workstations is not clear and need practical evaluation.

A necessary condition for accessing easily, at the analysis level, all the information is that our data, detector description, constants are correctly described in terms of the entity-relationship model Aleph has chosen.

---

## APPENDIX

Definition of a set of bidimensional representations of the Data considered as a minimum to ensure a first graphics tool for development of programs.

For all the following plots we want:

\_ to be able to select indepently each subcomponent of the detector i.e:

Vertex detector, ITC, TPC [end cap A, end cap B], EC [end cap A, barrel, endcap B], HC [end cap A, barrel, end cap B], LT [A, B], LC [A, B].

In the case of the calorimeters the different stacks should have the possibility to be shown independently or added.

\_ to be able to visualise any of them together at their respective place .

As we will see these views are often somehow artificial and cannot be always straightly constructed from projective views.

List of the projections.

In the following  $x, y, z, \rho, \theta, \phi, R$  are the standard coordinates defined in Aleph note 84 / 131.

\_\_\_  $x, y$  projection seen from  $z+$  or  $z-$

Note that if a parallel projection is adequate for the TPC ,the projective character of the calorimeters asks for a conical projection of the end caps on their entry face.

\_\_\_  $x, z$  or  $y, z$  projection seen from  $y+$  or  $y-$  and from  $x+$  or  $x-$

Here too, we have some problems with the representation of the calorimeters: for the barrel do we represent the part which is on the side of the view point? What projection for the end caps? A refinement is to define freely the projection plane passing through the  $z$  axis.

\_\_\_  $\rho, z$

The  $\rho$  is the distance to the beam axis but in the case of the calorimeters to have a correct view we need to take into account the dodecagonal structure and follow it.

\_\_\_  $\theta, \phi$ .

This can be a normal  $\theta, \phi$  plot, fine for the track chambers but where the dodecagonal structure of the calorimeters gives rise to undulations (not important if only the touched towers are shown) or a  $\theta', \phi$  plot where  $\theta'$  is the angle corresponding to the middle of a calorimeter module. This last projection is adequate for the calorimeter related information when the first one fits better the track chambers.

\_\_\_  $z, \phi$ .

\_\_\_  $\rho, \phi$ .

\_\_\_  $\rho, \theta$ .

\_\_\_  $R, \phi$ .

\_\_\_  $\theta, z$ .

\_\_\_  $R, \theta$ .

\_\_\_  $R, z$ .

---