# Basic Libraries Report

Bob Payne, Mark Stupar
18 March 1992

*Basic Libraries* consist of the lowest level classes, sufficiently general in scope to have utility for everyone. It includes portability (wrappers for platform specific ideosyncracies), number crunching (*e.g.*, math oriented matrix classes), containers, low level miscellaneous classes (*e.g.*, strings and complex), and exception handling. Added to this is the task of recommending an existing graphics package containing (at a minimum) both graphics primitives and drivers for both `X` windows and `PostScript`, with `CIC` specifically mentioned as a candidate.

## 1 Past Work

Our approach has been to build on existing freeware and shareware rather than just sitting down and coding everything from scratch.

Consequently, the bulk of our time to date has been spent in prowling around anonymous ftp servers for source code, porting potentially useful candidates to Charlottesville, and examining them in some detail. Packages considered for adoption have also been compiled and tested (though not extensively exercised).

In this manner, we've discarded most candidates, for a variety of reasons: requires use of a commercial package such as `Motif`; doesn't compile by following the instructions, or with a managable amount of hacking; doesn't look useful; looks like it will create as many problems as it solves; etc.

We've also used this opportunity to compare the available compilers and environments, and to obtain needed technical information (*e.g.*: assessing the feasibility of compilers; determining the required setup and mechanics of integrating FORTRAN subroutines into C++ classes; comparing the implementation of various packages to determine good and bad ways of doing things; accumulating classes which can profitably be used in implementing `AIPS++` classes, so as to avoid re-inventing wheels; the considerations and proper implementation of shared libraries; incorporating documentation in the code for use on-line).

This having been completed, we're up to speed and ready to proceed to the next phase of the project: implementing the classes within the responsibility of *Basic Libraries*.

The remainder of this report (1) makes specific recommendations regarding the import of pre-existing graphics into `AIPS++`, (2) briefly discusses what classes will implemented by us in general terms, and (3) discusses problems that may or will give us trouble. War stories, regurgitations of experiences, and techno-minutiae are intentionally omitted.

## 2 Graphics

We recommend that `InterViews` be adopted. Indeed, it is the only C++ package we have found that is feasible within the criteria specified, and both `InterViews` and the applications library distributed with it offer immediate benefits to the *User Interface* group. `InterViews` is immediately usable by this project upon its adoption.

We have found nothing that offers similar benefits to the *Image Handling* group. Of the packages we have examined, most offer some image handling capability, but all are ancilliary to the purposes of the packages of which they are a part, and are tailored to the needs of those packages. The best course we can suggest is to pick and choose among the available application packages, extracting those parts that are of interest, with the intent of modifying and building upon the code that is so borrowed. Feasible packages from which code can be borrowed are `CIC` and UNC's `COOL`, as well as portions of `MIRIAD`'s `MVX` (*e.g.*, palette manipulation, spreadsheeting, slice and dice, as well as sundry other features).

We also suggest that `PEX` (PHIGS+ Extended for X) be considered for use for the sake of its 3D capabilities. QUALIFICATION: as it comes with the `X` distribution, it is intended that `PEX` be installed under the X11 tree, and any given site might have chosen not to install that portion of `X`. Use of `PEX`, other than the simple borrowing of code, requires the installation of `PEX` at the site. While 3D capability is not a criterion for `AIPS++`, it should be advantageous to have that capability for the future, and we have found no other package that we can use towards this end. While its protocol is not fully implemented, its shortcomings are all in 3D areas that do not impair its utility to this project.

One graphics related issue seems to be in no specific group's area of responsibility. We note the availability of `WIP`, an existing package for the generation of production quality image hardcopy. While it is a part of `MIRIAD`, it is completely standalone, aside from its use of `PGPLOT`. It currently reads `MIRIAD` and `FITS` formats, but others can be added. The author (Jim Morgan, Maryland) is a member of BIMA, and attended the OOD/C++ class in Charlottesville this past January. The adoption of `WIP`, modified to read `AIPS++` data format, gives this project an immediately available means of compositing and annotating images.

Caveat: note that the as-is adoption of packages such as `InterViews`, which does not have a class naming convention assuring uniquenesss across different packages, precludes the use of like-named classes elsewhere within `AIPS++`.

## 3 Basic Library Classes

This is the area in which we expect to spend the remainder of our time on this project. Work estimates will be no better than faked numbers at present, but we

are confident that we can implement the *Basic Libraries* in the time remaining.

No package that we've found adequately serves the needs of this project, and we are resigned to writing and tailoring classes for the particular needs of `AIPS++`. To this end, we will implement the classes with `AIPS++` functionality that are within the responsibility of this group, relying on other groups to specify their needs to us. We've assembled a number of packages from which code can be borrowed and modified, and will use these to the extent possible.

The packages we expect to be most used by us are `GNU libg++` (particularly rich in functionality), `TI COOL` (excellent docs and it both complements and slightly overlaps `libg++`), and `RWVector2.2` (interfaces to numerical FORTRAN subroutines). A number of other packages are available for ad hoc borrowing (eg, from NIH) on an anecdotal basis.

This in no way constrains the other groups of the project to read or become familiar with any existing package, nor do we expect them to do anything more than to tell us what they want (*i.e.*, the software packages mentioned above are intended for use by *Basic Libraries* as a means of avoiding the re-invention of wheels).

We'll provide lists and documents on what is available (to serve the other groups as idea sources) and will begin work immediately on those fundamental classes whose need is "intuitively obvious".

# 4 Current Concerns and Problems

We don't have the upgrade to Sun's C++ that allows parameterized templates. The importance of the features available in this upgrade cannot be overemphasized.

Practical exception handling as a generality is not available except for trivially conceived circumstances (eg, divide by zero); its design would be ad hoc and highly local in scope.