#239: Inside Object Pascal

Written by: Keith Rollin June 1989

This Technical Note briefly explains why Object Pascal and MacApp should only be used to write applications and MPW tools.

Although Pascal can be used to write desk accessories, drivers, XCMDs and other types of stand–alone code, and Object Pascal is an extension of Pascal, Object Pascal cannot be used to write anything other than an application. This limitation is due to the fact that Object Pascal method dispatching relies on a valid `A5` pointing to a jump table. Because MacApp is written in Object Pascal, this limitation applies to it as well.

Once Over Lightly

Object methods cannot always be called directly. To explain why this is so, let's take a case from MacApp. Part of the way MacApp works includes defining `TView` objects that can draw themselves. Whenever an update event occurs, MacApp traverses the list of `TView` objects that are installed in a window and calls the `Draw` method for each one. However, how does Pascal know which `Draw` method to call? Does it call `TYourView.Draw`? Does it call `TView.Draw`? There is no way to know, at compile time, what `TView` objects and descendants of `TView` will be passed to the MacApp update routine. Therefore, there is no way to determine the appropriate `Draw` routine at compile time and generate a direct call to it.

Object Pascal solves this problem by maintaining data structures called Class Info Tables for each Object Class defined. These Class Info Tables not only contain information about the correct procedure to call whenever a message is sent to an object, but they also contain information used to create a new instance of that object.

The mechanism for this dispatching is quite complex and not described here. However, the main point is that the mechanism absolutely relies on special jump table entries. These jump table entries are used to dynamically map method calls to the correct procedure, using the information found in the Class Info Tables. Since desk accessories, drivers, and XCMDs, by their very nature, cannot have a jump table, you cannot use Object Pascal to create them.

Object Pascal **can** be used to write MPW tools, and, in fact, was used to create the MABuild and PostRez tools that come with MacApp 2.0.

Conclusion

For more information on how Object Pascal works, I highly recommend the article by Ken Doyle, “Introduction to Object Pascal,” anthologized in *The Complete MacTutor*, Volume 2. However, keep in mind that this information is already slightly out of date, and should not be counted on to be completely accurate at this time. In general, however, it is a good description of what is actually happening when a method call is made.

Further Reference:

- *Inside Macintosh*, Volume II-53, The Segment Loader
- *The Complete MacTutor*, Volume 2, “Introduction to Object Pascal”, p. 336
- Macintosh Technical Note #105, MPW Object Pascal Without MacApp
- Macintosh Technical Note #110, MPW: Writing Stand-Alone Code
- Macintosh Technical Note #220, Segment Loader Limitations