

---



---

## PLOT OBJECTS

---



---

By: Jan Pedersen (pedersen.PA @ Xerox.com)

Uses: PLOT and TWODGRAPHICS

Plot objects are the primitive quantities of the **PLOT** module. A plot object is abstracted as an instance of datatype PLOBJECT. A point plot object is an instance of PLOBJECT whose data component describes a point. That is, a point plot object is a subtype of PLOBJECT; all plot objects satisfy (`type? PLOBJECT FOO`), but only a point plot object satisfies in addition (`PLOBJECTSUBTYPE? POINT FOO`).

A PLOBJECT is both a datatype and a collection of functions that implements a set of generic operations on that plot object. A plot object must know how to draw itself, erase itself, highlight itself, etc. The PLOT module then deals only with generic operations, and allows the plot objects to implement them as is appropriate.

PLOBJECT	[Datatype]
OBJECTFNS	[Field]
Must be an instance of PLOTFNS	
OBJECTSUBTYPE	[Field]
Describes the plot objects subtype	
OBJECTUSERDATA	[Field]
Space for a property list	
OBJECTMENU	[Field]
The object's MENU	
OBJECTLABEL	[Field]
Something to print	
OBJECTDATA	[Field]
Space for a datatype that describes the subtype of this PLOBJECT	
The field OBJECTFNS must be an instance of PLOTFNS, essentially a vector of functions which implements the generic operations.	
PLOTFNS	[Datatype]
DRAWFN	[Field]
Implements the DRAWOBJECT generic operation	
ERASEFN	[Field]
etc.	

HIGHLIGHTFN	[Field]
LOWLIGHTFN	[Field]
LABELFN	[Field]
MOVEFN	[Field]
EXTENTFN	[Field]
DISTANCEFN	[Field]
COPYFN	[Field]
PUTFN	[Field]
GETFN	[Field]

The generic operations are:

(DRAWPLOT OBJECT *object viewport plot*) [Function]

Draw the object within viewport. A VIEWPORT may be thought of as a sub imagestream. It will usually be associated with the plot's PLOTWINDOW, but might also be associated with some other image stream. Typically this generic operation will make use of functions from TWODGRAPHICS and the position of the object in world coordinates. The plot is also passed as an argument, so that the draw operation may make use of information cached on the property list of plot.

The only operation that is expected to draw on streams other than the PLOTWINDOW is drawobject, so the drawfn may have to behave differently depending on the imagestreamtype of the stream. All other generic operations are assumed to operate on the PLOTWINDOW. The idea here is that plot's may be drawn on any stream, but may be interacted with only through the PLOTWINDOW. It is also guaranteed that an object will be drawn before it is erased, highlighted, etc.

(ERASEPLOT OBJECT *object viewport plot*) [Function]

Erase the object from the viewport. The inverse of DRAWOBJECT. It is guaranteed that the viewport will be on the PLOTWINDOW

(HIGHLIGHTPLOT OBJECT *object plot*) [Function]

Highlight the object. Used in selection.

(LOWLIGHTPLOT OBJECT *object plot*) [Function]

The inverse of HIGHLIGHTOBJECT. With XOR drawing the HIGHLIGHTFN and the LOWLIGHTFN can often be the same.

(MOVEPLOT OBJECT *object dx dy plot*) [Function]

Destructively alter the object's OBJECTDATA, so that its position is moved dx, dy units (in world coordinates).

(LABELPLOT OBJECT *object plot*) [Function]

If it is desired to label the object, the LABELFN will be called. Often the function LABELGENERIC will do the trick.

(EXTENTOFPLOT OBJECT *object plot*) [Function]

Should return an EXTENT, which expresses the range of the object in world coordinates.

EXTENT	[Datatype]
MINX	[Field]
Minimun extent in the X (horizontal) direction	
MAXX	[Field]
Maximun extent in the X (horizontal) direction	
MINY	[Field]
Minimun extent in the Y (vertical) direction	
MAXY	[Field]
Maximun extent in the Y (vertical) direction	
All fields are type floating.	

(DISTANCETO PLOT OBJECT *object streamposition plot*) [Function]

Should return a number (more efficient if it returns a SMALLP), which is some measure of the distance from the REPRESENTATION of the object to the POSITION streamposition. Note that distance is calculated in stream coordinates, NOT world coordinates. This is done for efficiency and logical consistency. Selection makes most sense as an activity in stream coordinates.

A plot object will typically cache its stream coordinates when it is drawn. Although not strictly necessary (it is always possible to backsolve to stream coordinates from world coordinates), this improves efficiency many fold by avoiding generation of floating point boxes.

The following functions are provided to allow the plot object to customize how it is copied, printed on file, etc. The generic defaults will usually be satisfactory.

(COPY PLOT OBJECT *object plot*) [Function]

Returns a copy of object. COPY OBJECT will create a new instance of PLOT OBJECT and copy over all the fields of object except for OBJECT DATA. The object's COPY FN is evoked with the arguments object and plot and is expected to return a new instance of OBJECT DATUM. The objects property list is handled as follows: If object has a prop COPY FN (which may be a function or list of functions), for each property it is called with the arguments newobject, oldobject, plot, propname. If the returned value is non-nil it is used as the value for that property on newobject; else the prop value is HCOPY ALL'ed. If the value of COPY FN is a list of functions, they are invoked in order head to tail, and the first non-NIL value is used as the new value.

(PRINT PLOT OBJECT *object plot stream*) [Function]

Writes out to stream an HREADable symbolic representation of object. As in COPY OBJECT, PRINT OBJECT takes care of all PLOT OBJECT fields except of OBJECT DATUM. The objects PUT FN will be invoked with the arguments object plot stream and is expected to write out a representation of OBJECT DATUM which is HREADable. This will usually be in prop list format.

Again the prop list of object requires special handling. The special object prop PUT FN may be a function or list of functions. For each property it will be invoked with the arguments object plot propname and stream and if it returns a non-NIL value, it is assumed that property has been written out in a HREADable format. Again, if the the PUT FN prop is a list of fns then if any one of them returns non-NIL then the property is assumed written out. If there is no PUT FN then the property is (HPRINT prop stream NIL T) 'ed.

PUTFNS may put out special lists of the form ((FUNCTION *fname*) *arg*) in which case *fname* will be invoked at HREAD time with *args* object *plot* *propname* *arg* and *fname* will be expected to return the *propvalue* of *propname*.

(READPLOT OBJECT *stream*) [Function]

Reads in the product of PRINT OBJECT. Calls the objects GETFN to read in the OBJECTDATA field.

An instance of PLOT FNS may be created by the function:

(CREATEPLOT FNS *drawfn erasefn extentfn distancefn highlightfn*  
*lowlightfn labelfn movefn copyfn putfn getfn borrowfrom*) [Function]

Returns an instance of PLOT FNS. *Drawfn*, *erasefn*, and *extentfn* are required. If a *distancefn* is supplied then so must be a *highlightfn*. *Lowlightfn* defaults to *highlightfn*, *labelfn* defaults to LABELGENERIC. The other arguments also default to some safe, if not too efficient *genericfn*.

A primitive inheritance scheme is implemented via the optional argument *borrowfrom*. If supplied, *borrowfrom* must be an instance of PLOT FNS. Before creating the new instance of PLOT FNS, the NIL arguments passed are filled in from the fields of *borrowfrom*, with the following exception; *lowlightfn* is only inherited if *highlightfn* is also NIL.

The OBJECTDATA field will typically be a datatype which holds the data characterizing the PLOT OBJECT. For example a point plot object will have an OBJECTDATA field whose value is an instance of the datatype POINTDATA (has fields *position*, *symbol*, etc). So, a point PLOT OBJECT is a specialization of PLOT OBJECT. The field OBJECTSUBTYPE is supplied to make the subtype explicit. The following macro is provided to facilitate testing for plot object subtypes.

(PLOT OBJECTSUBTYPE? *subtype plotobject*) [Macro]

Essentially tests if (EQ *subtype* (fetch OBJECTSUBTYPE of *plotobject*))

(PLOT OBJECTSUBTYPE *plotobject*) [Function]

Returns the value of the OBJECTSUBTYPE field.

PLOT OBJECTS may be created via the function:

(CREATEPLOT OBJECT *objectfns objectlabel objectmenu objectdata*) [Function]

Returns an instance of PLOT OBJECT. Coerces *objectmenu* into a MENU if it is an item list.

The following subtypes of PLOT OBJECT are currently implemented.

pointPLOT OBJECT, curvePLOT OBJECT, polygonPLOT OBJECT, linePLOT OBJECT,  
graphPLOT OBJECT, texttPLOT OBJECT, filledrectanglePLOT OBJECT, compound PLOT OBJECT

The functions CREATEPOINT, etc. return an instance of PLOT OBJECT, with the appropriate OBJECTFNS and OBJECTDATA. In order for this to work, some initializations must be done at load time.

The function PLOT.SETUP performs the initializations at LOAD time.

(PLOT.SETUP *opstable*) [Function]

*Opstable* must be a list of lists of the form:

(

```
(subtype1 (opname1 function1) (opname2 function2) ....  
(subtype2 (opname1 function1) (opname2 function2) ....  
.....  
(subtypenamen (opname1 function1)(opname2 function2) ....  
)
```

Creates one instance of PLOTFNS for each subtype.

In summary, to add a new plot object you need to:

- Determine the data required to describe the new subtype. This may involve declaring a new datatype.
- Write functions similar to CREATEPOINT and PLOTPOINT for the new subtype.
- Write (or borrow) the functions which implement the generic ops described above.
- Invoke MAKEPLOTFNS to create an instance of PLOTFNS for the new plot object subtype, which all objects of that subtype will refer to.
- If continued use of the new plot object is contemplated, PLOT.SETUP should be evoked at load time to effect the proper initializations.

Look at the code for existing plot objects for more details. The point plot object is the simplest example.