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 PLOTOBJECT. A point plot object is an instance of PLOTOBJECT whose data component describes a point. That is, a point plot object is a subtype of PLOTOBJECT; all plot objects satisfy `(type? PLOTOBJECT FOO)`, but only a point plot object satisfies in addition `(PLOTOBJECTSUBTYPE? POINT FOO)`.

A PLOTOBJECT 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.

PLOTOBJECT [Datatype]

OBJECTFNS [Field]

Must be an instance of PLOTFNS

OBJECTSUBTYPE [Field]

Describes the plot objects subtype

OBJECTUSERDATA [Field]

Space for a propery list

OBJECTMENU [Field]

The object's MENU

OBJECTLABEL [Field]

Something to print

OBJECTDATA [Field]

Space for a datatype that describes the subtype of this PLOTOBJECT

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:

(DRAWPLOTOBJECT *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 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.

(ERASEPLOTOBJECT *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

(HIGHLIGHTPLOTOBJECT *object plot*) [Function]

Highlight the object. Used in selection.

(LOWLIGHTPLOTOBJET *object plot*) [Function]

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

(MOVEPLOTOBJECT *object dx dy plot*) [Function]

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

(LABELPLOTOBJECT *object plot*) [Function]

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

(EXTENTOFPLOTOBJECT *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.

(DISTANCETOPLOTOBJECT *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.

(COPYPLOTOBJECT *object plot*) [Function]

Returns a copy of object. COPYOBJECT will create a new instance of PLOTOBJECT and copyover all the fields of object except for OBJECTDATA. The object's COPYFN is evoked with the agruments object and plot and is expected to return a new instance of OBJECTDATUM. The objects property list is handled as follows: If object has a prop COPYFN (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 HCOPYALL'ed. If the value of COPYFN 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.

(PRINTPLOTOBJECT *object plot stream*) [Function]

Writes out to stream an HREADable symbolic representation of object. As in COPYOBJECT, PRINTOBJECT takes care of all PLOTOBJECT fields except of OBJECTDATUM. The objects PUTFN will be invoked with the arguments object plot stream and is expected to write out a representation of OBJECTDATUM which is HREADable. This will usually be in prop list format.

Again the prop list of object requires special handling. The special object prop PUTFN 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 PUTFN 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 PUTFN then the property is (HPRINT prop stream NIL T) 'ed.

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

(READPLOTOBJECT *stream*) [Function]

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

An instance of PLOTFNS may be created by the function:

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

Returns an instance of PLOTFNS. 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 PLOTFNS. Before creating the new instance of PLOTFNS, 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 PLOTOBJECT. 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 PLOTOBJECT is a specialization of PLOTOBJECT. The field OBJECTSUBTYPE is supplied to make the subtype explicit. The following macro is provided to facilitate testing for plot object subtypes.

(PLOTOBJECTSUBTYPE? *subtype plotobject*) [Macro]

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

(PLOTOBJECTSUBTYPE *plotobject*) [Function]

Returns the value of the OBJECTSUBTYPE field.

PLOTOBJECTS may be created via the function:

(CREATEPLOTOBJECT *objectfns objectlabel objectmenu objectdata*) [Function]

Returns an instance of PLOTOBJECT. Coerces objectmenu into a MENU if it is an item list.

The following subtypes of PLOTOBJECT are currently implemented.

pointPLOTOBJECT, curvePLOTOBJECT, polygonPLOTOBJECT, linePLOTOBJECT,
graphPLOTOBJECT, texttPLOTOBJECT, filledrectanglePLOTOBJECT, compound PLOTOBJECT

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

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

(PLOT.SETUP *opstable*) [Function]

Opstable must be a list of lists of the form:

(

(subtypename1 (opname1 function1) (opname2 function2) ....

(subtypename2 (opname1 function1) (opname2 function2) ....

.....

(subtypenamen (opname1 function1)(opname2 function2) ....

)

Creates one instance of PLOTFNS for each subtypename.

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.