Visualization Control System:
Python Command Line and Application Programming Interface

PCMDI Staff)
cdat-developers@lists.sourceforge.net

May 21, 2001

Program for Climate Model Diagnosis and Intercomparison (PCMDI) Lawrence Livermore National Laboratory, Livermore California 94550

Legal Notice

Table of Contents

CHAPTER 1 Introduction

The PCMDI Visualization Control System (VCS) is expressly designed to meet the needs of climate scientists. Because of the breadth of its capabilities, VCS can be a useful tool for other scientific applications as well. VCS allows wide-ranging changes to be made to the data display, provides for hardcopy output, and includes a means for recovery of a previous display.

Basic Concepts of VCS

In the VCS model, the data display is defined by a trio of named object sets, designated the "primary objects" (or "primary elements"). These include:

- the data, which define what is to be displayed and is ingested into the system via other PCMDI software components or via the Numeric module;
- the graphics method, which specifies the display technique; and
- the picture template, which determines the appearance of each segment of the display. Tables for manipulating these primary objects are stored in VCS for later recall and possible use.

In addition, detailed specification of the primary objects' attributes is provided by eight "secondary objects" (or secondary elements"):

1. *colormap:* specification of combinations of 256 available colors
2. *fill area:* style, style index, and color index
3. *format:* specifications for converting numbers to display strings
4. *line:* line type, width and color index
5. *list:* a sequence of pairs of numerical and character values
6. *marker:* marker type, size, and color index
7. *text:* text font type, character spacing, expansion and color index
8. *text orientation:* character height, angle, path, and horizontal/vertical alignment

By combining primary and secondary objects in various ways (either at the command line or in a program), the VCS user can comprehensively diagnose and intercompare climate model simulations. VCS provides capabilities to:

- View, select and modify attributes of data variables and of their dimensions
- Create and modify existing template attributes and graphics methods
- Save the state-of-the-system as a script to be run interactively or in a program
- Save a display as a Computer Graphics Metafile (CGM), GIF, Postscript, Sun Raster, or Encapsulated Postscript file
- Perform grid transformations and compute new data variables
- Create and modify color maps
- zoom into a specified portion of a display
- Change the orientation (portrait vs. landscape) or size (partial vs. full-screen) of a display
- Animate a single data variable or more than one data variable simultaneously
- Display different map projections

Purpose of this document

This document will focus primarily on the VCS software commands necessary to operate VCS with minimal knowledge. The knowledge of VCS will gradually be increased allowing the user to construct more complex visualization operations that are vital to their scientific research. The material contained in this document will walk you through simple VCS operations and use CDMS module to ingest data sets and to manipulate the data before it is displayed. Because the best way to learn a

new tool is by examples, this document is heavy on examples and provides an extensive command reference guide.

TABLE 1. Guide to This Document

Chapter Title and Location	Purpose
“VCS Installation and Setup” on page 8	This chapter explains how to install and test VCS (either from the source code or from the distributed shared library).
“Selecting or Creating Data” on pa ge11	This chapter explains the type of data needed for the VCS module and how to read data into the module.
“Overview” on page 12	Overview chapter explaining VCS’s Python Application Interface.
“VCS Command Reference Guide” on page 28	If you need to know the full extent of a command and all of its parameters, then this is where you want to be.
“VCS Primary Objects” on pag e122	VCS primary object list and their members.
“VCS Secondary Objects” on pa ge145	VCS secondary object list and their members.
“VCS Examples” on page148	This chapter has examples showing how to use VCS in the Python environment. If you just want to get started, then we suggest you start with this chapter in conjunction with Chapter 5.
“Quick Reference Guides” on page198 9. Includes a one-page “cheat sheet” on page “VCS Cheat Sheet” on pag e204.	If you need a command in a hurry or want to review a command quickly, then this is the place to be.
“Fonts, Lines, Markers, and Patterns” on page 205	Font, line, marker, and pattern symbols.

CHAPTER 2

VCS Installation and Setup

These steps below should be followed before running VCS, and are necessary in order to produce plots on the screen and in background mode. The steps include putting in place various input files and setting up fonts for XGKS.

Mandatory Input Files

You must have the XGKS fonts directory set. XGKS is an implementation of the ANSI Graphical Kernel System in C, the programming language used to develop VCS. XGKS fonts pertain to those used for graphical displays on the VCS Canvas.

Before running VCS, it is necessary to set the environment variable XGKSFontDir. That is, enter:

setenv XGKSFontDir /the_absolute_path/fontdb

where */the_absolute_path* denotes the absolute path location for the */fontdb* directory.

It is best if this *setenv* statement is included in the .login or .cshrc file located in the user's home directory (/$HOME).

Note: if VCS aborts with the message:

"XGKS: can't load font 1 from /the_absolute_path./fontdb - aborting"

then the *XGKSFontDir* variable is improperly set.

Nine XGKS font styles are supported at this time (24 additional fonts are in the works):

- SanSerif Roman
- Serif Roman
- Sanserif Bold Roman
- Serif Bold Roman
- Sanserif Italic Roman
- Serif Italic Roman
- Sanserif Script
- Serif Script
- Gothic

Recommended Input Files

It is strongly recommended that two input files be put in place before attempting to run VCS, but it is not necessary to have these files in place in order to run VCS. These include a file for specifying initial attributes called, *"initial.attributes"* and a file for printing hard copy output called, *"HARD_COPY*".

File for Specifying Initial Attributes

At start-up, VCS reads a script file named *initial.attributes* that defines the initial settings of the VCS tables. Although not required to run VCS, this *initial.attributes* file contains many predefined settings to aid the beginning user of VCS. The path to the file must be:

/$HOME/PCMDI_GRAPHICS/initial.attributes

where */$HOME* denotes the user's home directory. (Note, when VCS is executed for the first time, a */PCMDI_GRAPHICS* subdirectory will be created automatically if it has not already been created.)

Changing the *initial.attributes* File

The user can customize the contents of the *initial.attributes* file. This is most easily accomplished by changing the contents of a VCS object saving the state of the system with the use of the *saveVCSinitialattribute()* function. This action will place a new *initial.attributes* file with the desired setting(s) in the user's

/$HOME/PCMDI_GRAPHICS

directory. For recovery purposes, the old *initial.attributes* file is copied to file *initial.attributes%* in the same directory.

File for Printing

VCS graphical displays can be printed only if the user customizes a HARD_COPY file (included with the VCS software) for the home system. The path to the HARD_COPY file must be:

/$HOME/PCMDI_GRAPHICS/HARD_COPY

where */$HOME* denotes the user's home directory. The HARD_COPY file contains the following necessary information for printing at the user's home site:

- A list of the available home Postscript printing devices.
- The absolute path on the home system to the *gplot* executable (provided with the VCS software) that converts files in the Computer Graphics Metafile (CGM) format to Postscript files.
- Instructions for setting the environment aliases `landscape' and `portrait' that are used to generate Postscript files outside VCS.

The setting for the environment variable `PRINTER'. (When `PRINTER' is set to `printer', VCS assumes that the printer manager `*lpr*' is in use; when `PRINTER' is unset, VCS assumes that the printer manager is `*lp*'. Incidences of the message:

"Error - In sending CGM file to printer" are an indication of an incorrect setting for the `PRINTER' environment variable.)"

CHAPTER 3

Selecting or Creating Data

This chapter explains the type of data needed for the VCS module and how to read data into the VCS module.

The VCS module can except data from the CDMS module, the CU module, or the Numeric module. For use on how to use either of the mentioned modules, see their respective documentation. For examples on the direct use of these modules, see the “VCS API Examples” chapter and the examples located throughout this texts.

CHAPTER 4 Overview

VCS Model Overview

The VCS model is defined by a trio of named attribute sets, designated the "Primary Objects" (also known as "Primary Elements"). These include: the data, which specifies what is to be displayed and are obtained from the "cdms", "cu", or Numeric modules; the graphics method, which specifies the display technique; and the picture template, which determines the appearance of each segment of the display.

VCS Primary Objects (or Primary Elements)

A description of each primary object is warranted before showing their use and usefulness in VCS. See descriptions below.

Graphics Method Objects

A graphics method simply defines how data is to be displayed on the screen. Currently, there are eleven different graphics methods with more on the way. Each graphics method has its own unique set of attributes (or members) and functions. They also have a set of core attributes that are common in all graphics methods. The descriptions of the current set of graphics methods are as follows:

- boxfillobject - The boxfill graphics method draws color grid cells to represent the data on the VCS Canvas. Its class symbol or alias is “Gfb”.
- continentsobject - The continents graphics method draws a predefined, generic set of continental outlines in a longitude by latitude space. To draw continental outlines, no external data set is required. Its class symbol or alias is “Gcon”.

- isofillobject - The isofill graphics method fills the area between selected isolevels (levels of constant value) of a two-dimensional array with a user-specified color. Its class symbol or alias is “Gfi”.
- isolineobject - The isoline graphics method draws lines of constant value at specified levels in order to graphically represent a two-dimensional array. It also labels the values of these isolines on the VCS Canvas. Its class symbol or alias is “Gi”.
- outfillobject - The outfill graphics method fills a set of integer values in any data array. Its primary purpose is to display continents by filling their area as defined by a surface type array that indicates land, ocean, and sea-ice points. Its class symbol or alias is “Gfo”.
- outlineobject - The Outline graphics method outlines a set of integer values in any data array. Its primary purpose is to display continental outlines as defined by a surface type array that indicates land, ocean, and sea-ice points. Its class symbol or alias is “Go”.
- scatterobject - The scatter graphics method displays a scatter plot of two 4-dimensional data arrays, e.g. A(x,y,z,t) and B(x,y,z,t). Its class symbol or alias is “GSp”.
- vectorobject - The Vector graphics method displays a vector plot of a 2D vector field. Vectors are located at the coordinate locations and point in the direction of the data vector field. Vector magnitudes are the product of data vector field lengths and a scaling factor. Its class symbol or alias is “Gv”.
- xvsyobject - The XvsY graphics method displays a line plot from two 1D data arrays, that is X(t) and Y(t), where ‘t’ represents the 1D coordinate values. Its class symbol or alias is “GXY”.
- xyvsyobject - The Xyvsy graphics method displays a line plot from a 1D data array, i.e. a plot of X(y) where ‘y’ represents the 1D coordinate values. Its class symbol or alias is “GXy”.
- Yxvsxobject - The Yxvsx graphics method displays a line plot from a 1D data array, i.e. a plot of Y(x) where ‘x’ represents the 1D coordinate values. Its class symbol or alias is “GYx”.

Picture Template Object

A picture template determines the location of each picture segment, the space to be allocated to it, and related properties relevant to its display. The description of the picture template is as follows:

- templateobject - Picture Template attributes describe where and how segments of a picture will be displayed. The segments are graphical representations of: textual identification of the data formatted values of single-valued dimensions and mean, maximum, and minimum data values axes, tick marks, labels, boxes, lines, and a legend that is graphics-method specific the data. Picture templates describe where to display all segments including the data. Its class symbol or alias is “P”.

Data Objects

Array data attribute sets and their associated dimensions are to be modified outside of VCS. See the CDMS, CU, and Numeric module documentation for data extraction, creation and manipulation.

VCS Secondary Objects (or Secondary Elements)

A description of each secondary object is warranted before showing their use and usefulness in VCS. It is these secondary objects that defines the detailed specification of the primary objects’ attributes. Currently, there are five secondary objects with more to follow.

Colormap Object

The colormap object is used to specify, create, and modify colormaps. There are 256 colors and color indices, but only the first 240 color indices can be modified (indices 240 through 255 are reserved for VCS internal use). The description of the colormap object is as follows:

- colormapobject - A colormap contains 240 user-definable colors that are used for graphical displays. The color mixtures are defined in terms of percentages of red, green, and blue colors (0 to 100% for each). The resulting color depends on the specified mixtures of red, green, and blue. Its class symbol or alias is “Cp”.

Note: VCS colormaps are objects, but they are not referenced like other secondary objects.

Fillarea Object

The fillarea objects allows the user to edit fillarea attributes, including fillarea interior style, style index, and color index. The description of the fillarea object is as follows:

- fillareaobject - The fill area attributes are used to display regions defined by closed polygons, which can be filled with a uniform color, a pattern, or a hatch style. Attributes specify the style, color, position, and dimensions of the fill area. Its class symbol or alias is “Tf”.

Line Object

The line object allows the editing of line type, width, and color index. The description of the line object is as follows:

- lineobject – The line attributes specify the type, width, and color of the line to be drawn for a graphical display. Its class symbol or alias is “Tl”.

Marker Object

The marker object allows the editing of the marker type, width, and color index. The description of the marker object is as follows:

- markerobject – The marker attribute specifies graphical symbols, symbol sizes, and colors used in appropriate graphics methods. Its class symbol or alias is “Tm”.

Text Objects

Graphical displays often contain textual inscriptions, which provide further information. The text-table object attributes allow the generation of character strings on the VCS Canvas by defining the character font, precision, expansion, spacing, and color. The text-orientation object attributes allow the appearance of text character strings to be changed by defining the character height, up-angle, path, and horizontal and vertical alignment. The text-combined object is a combination of both text-table and text-orientation objects. The description of the text objects are as follows:

- textcombinedobject - The text-combined attributes combine the text-table attributes and a text-orientation attributes together. From combining the two

classes, the user is able to set attributes for both classes at once (i.e., define the font, spacing, expansion, color index, height, angle, path, vertical alignment, and horizontal alignment). Its class symbol or alias is “Tc”.

- textorientationobject - The text-orientation attributes set names that define the height, angel, path, horizontal alignment and vertical alignment. Its class symbol or alias is “To”.
- texttableobject - The text-table attributes set names that define the font, spacing, expansion, and color index. Its class symbol or alias is “Tt”.

Initializing VCS

Importing VCS

In Python, before one can start using a module they must first load it. To load the VCS module, like all other Python modules, either type:

“from vcs import *”; or
“import vcs”

If you use “import vcs”, then you must prepend “vcs” to certain calls (e.g., “vcs.help()”). If you use “from vcs import *”, then you must be aware of possible name clashes. That is, if two packages are imported using the form “from *name* import *” and both have a “*help*” function, then Python doesn’t know which “*help*” function to call. For such cases, and indeed as an unspoken rule, it is best to use “import *name*” to avoid name clashing between packages.

VCS Initialize

To construct a VCS Canvas object type the following:

```
a=vcs.init()
```

There can only be at most 8 VCS Canvas objects initialized at any given time. When a VCS Canvas object is initialized, the current template and graphics method will be displayed. For example:

“‘Template' is currently set to P_default.

Graphics method 'Boxfill' is currently set to Gfb_default.'"

VCS Functions

Plotting in VCS

There are several different ways to display data on the VCS Canvas. The most basic way is to use the plot() function. The simple plot() function command: plot(array1,[array2], [template object], [graphics_method object]). The examples below are showing how to plot a simple array using default values for everything else.

4.0.0.1 Plotting a CDMS Persistent Array

```
import vcs                      # import the VCS module
import cdms                     # import CDMS for ingesting data
cdms.setAutoReshapeMode('on')  # needed by CDMS module if
                                # data is reshaped
f=cdms.openDataset('example.nc')# open file via the cdms module
psl=f.variables['clt']          # get the "psl" variable
data=psl[…]                     # get the "psl" variable data (Note, data is
                                # now a Numeric array)
v=vcs.init()                    # "v" is an instance of the VCS class
                                # object (constructor)
v.plot(data, variable=psl)      # call "plot" function to display the CDMS
                                # Persistent Array on the VCS Canvas
                                # using default settings
```

4.0.0.2 Plotting a CU Slab Array

```
import vcs                      # import the VCS module
import cu                       # import CU for ingesting data
f=cu.open('example.nc')         # open file via the cu module
s=f.getslab('clt')              # get the "psl" variable slab
v=vcs.init()                    # "v" is an instance of the VCS
                              # class object (constructor)
v.plot(s)                       # call "plot" function to display the CU "slab"
                          # array on the VCS Canvas using default settings
```

4.0.0.3 Plotting a Numeric Array

```
import vcs                      # import the VCS module
import Numeric                  # import Numeric for generating data
a=Numeric.array([[1,2,3],[4,5,6],[7,8,9]])  # create a simple
                                    # multidimensional array
v=vcs.init()                    # “v” is an instance of the VCS
                                # class object (constructor)
v.plot(a)                       # call “plot” function to display the Numeric
                                # “array” data on the VCS Canvas using
                                # default settings
```

4.0.0.4 Plotting Using Keyword Arguments

The plot function has many overriding keyword arguments that control textural and graphical output of the display. As shown above, the arguments necessary to plot data can be very simple. Below is a more complex plot() function showing the uses of array objects, template object, graphics method object, and key word arguments. Objects placed in brackets “[ ]” indicate optional entries into the plot function:

plot(array1, [array2 [, template [, graphics method [,key=value [, key=value [, …]]]]]]),

where array1 and array2 are Numeric arrays or CU slabs; template represents a template object; graphics method represents a graphics method object (such as, boxfill or isofill); and key=value represents one variable attributes used to display textual information or to modify the plot’s output. If no template is specified, then the default template will be used. If no graphics method is specified, then the default boxfill graphics method is used.

Variable attribute keys are:

```
comment1    = string            # Comment plotted above file_comment
comment2    = string            # Comment plotted above comment1
comment3    = string            # Comment plotted above comment2
comment4    = string            # Comment plotted above comment3
file_comment= string            # Comment (defaults to file.comment)
hms         = string (hh:mm:ss) # Hour, minute, second
long_name   = string                # Descriptive variable name
missing_value= (same type as array) # Missing data value
                                        #(defaults to var.getMissing())
name        = string                # Variable name (defaults to var.id)
```

```
    time          = cdtime              # instance (relative or absolute),
                                        # cdtime, reltime or abstime value
    units         = string               # Variable units
    ymd           = string (yy/mm/dd)  # Year, month, day

  Dimension attribute keys (dimension length=n) are:
    [x|y|z|t|w]array1   = NumPy array of length n# x or y Dimension values
    [x|y|z|t|w]array2   = NumPy array of length n   # x or y Dimension

values:
    [x|y]bounds         = NumPy array of shape (n,2)# x or y Dimension

boundaries:
    [x|y|z|t|w]name     = string                  # x or y Dimension name
    [x|y|z|t|w]units    = string                  # x or y Dimension units
    [x|y]weights        = NumPy array of length n # x or y Dimension
                                                  # weights (used # to
                                                 # calculate area-weighted
                                                 # mean)

  CDMS object attributes are:
    [x|y|z|t|w]axis= CDMS axis object          # x or y Axis
    grid           = CDMS grid object          # Grid object (e.g.
                                               # grid=var.getGrid()
    variable       = CDMS variable object     # Variable object

  Miscellaneous attributes are:
    [x|y]rev       = 0|1                       # if ==1, reverse the direction of
                                               # the x or y axis
  continents       = 0,1,2,3,4,5,6,7,8,9,10,11   # if >=1, plot continental
                                        # outlines (default: plot if x-axis is
                                        # longitude, y-axis is latitude -or-
                                        # xname is 'longitude' and yname is
                                        # 'latitude'
                                        # The continents-type values are integers
                                        # ranging from 0 to 11, where:
                                        #    0 signifies "No Continents"
                                        #    1 signifies "Fine Continents"
                                        #    2 signifies "Coarse Continents"
                                        #    3 signifies "United States"
                                        #    4 signifies "Political Borders"
```

```
                                    #    5 signifies "Rivers"

                                    # Values 6 through 11 signify the line
                                    # type defined by the files
                                    # data_continent_other7
                                    # through data_continent_other12.

Graphics Output in Background Mode:
  bg            = 0|1                   # if==1, create images in the
                                         # background (Don't display the
                                         # VCS Canvas)
```

Note: More specific attributes take precedence over general attributes. In particular, specific attributes override variable object attributes; dimension attributes and arrays override axis objects, which override grid objects, which override variable objects.

For example, if both 'file_comment' and 'variable' keywords are specified, the value of 'file_comment' is used instead of the file comment in the parent of variable. Similarly, if both 'xaxis' and 'grid' keywords are specified, the value of 'xaxis' takes precedence over the x-axis of grid.

4.0.0.5 Example Plotting Using Keyword Arguments:

When using the plot() function, keep in mind that all keyword arguments must be last. Note that the order of the objects is not restrictive, just as long as they are before any keyword argument. See the two plot() function examples below.

```
import vcs                          # import the VCS module
import cu                           # import CU for ingesting data
f=cu.open(‘example.nc’)             # open file via the cu module
s=f.getslab(‘psl’)                  # get the “psl” variable slab
v=vcs.init()                        # “v” is an instance of the VCS
                                    # class object (constructor)
t=x.createtemplate('new')           # create a new template from the default
                                    # template
iso=x.createisofill(‘new’)          # create a new isofill graphics method from
                                    # the default isofill
x.plot(s, t, iso, continents=0)     # call “plot” function to display the CU
                                    # slab “s” using
                                        # newly created template “t” and isofill
```

```
                                    # graphics method “iso” and turn
                                    # continents off
x.clear()                               # clear the VCS Canvas of all plots
x.plot(t,iso,s,continents=0)        # shows that the order of the objects are
irrelevant, but
                                        # not that all keyword arguments must be last.
```

Other Plotting functions in VCS

There are other ways to plot data in VCS. These additional plotting routines utilizes the same parameter format as the plot() function. What makes these plotting functions unique are their direct association with the graphics methods. That is, each graphics method has its own plot function. For example, if the user wishes to plot data using the isofill graphics method, then the function isofill() can be used instead of the plot() function. If the isofill object is not specified then the default isofill graphics method will be used. The user can also pass down the name of the graphics method to be used. In some ways, the graphics method plot functions can be thought of as short cuts to plotting data.

Note, if a different graphics method object is specified and passed down to one of these alternate plot functions, then the alternate plot function will behave as the plot() function and plot the data in the specified graphics method format.

See table below for additional plot functions.

Table 4.6

Plot Function	Description
boxfill()	plot data using the boxfill graphics method
continents()	plot to the screen continental graphics method
isofill()	plot data using the isofill graphics method
isoline()	plot data using the isoline graphics method
outfill()	plot data using the outfill graphics method
outline()	plot data using the outline graphics method
scatter()	plot data using the scatter graphics method
vector()	plot data using the vector graphics method
xvsy()	plot data using the xvsy graphics method

Table 4.6

Plot Function	Description
xyvsy()	plot data using the xyvsy graphics method
yxvsy()	plot data using the yxvsy graphics method

Checking VCS Objects

In some cases, there my be a need to check or verify an object in VCS. For these cases, you may want to use the query functions below.

Check for VCS graphics method objects:

Table 4.7

Query Function	Description
isboxfill()	verifies a boxfill graphics method object
iscontinents()	verifies a continents graphics method object
isisofill()	verifies an isofill graphics method object
isisoline()	verifies an isoline graphics method object
isoutfill()	verifies an outfill graphics method object
isoutline()	verifies an outline graphics method object
isscatter()	verifies a scatter graphics method object
isvector()	verifies a vector graphics method object
isxvsy()	verifies a xvsy graphics method object
isxyvsy()	verifies a xyvsy graphics method object
isyxvsx()	verifies a yxvsx graphics method object
isgraphicsmethod()	verifies if an object is one of the graphics methods: boxfill, isofill, isoline, outfill, outline, continents, scatter, vector, xvsy, xyvsy, or yxvsx.
graphicsmethod-name()	Returns the name of the graphics methods object. Returns either: ‘boxfill’, ‘isofill’, ‘isoline’, ‘outfill’, ‘outline’, ‘continents’, ‘scatter’, ‘vector’, ‘xvsy’, ‘xyvsy’, or ‘yxvsx’.

Check for VCS secondary objects:

Table 4.8

Query Function	Description
isfillarea()	verifies a fillarea secondary object
isline()	verifies a line secondary object
ismarker()	verifies a marker secondary object
istextcombined()	verifies a text-combined secondary object
istextorientation()	verifies a text-orientation secondary object
istexttable()	verifies a text-table secondary object
issecondaryobject()	verifies if an object is one of the secondary objects: fillarea, line, marker, textcombined, textorientation, or texttable

Check for VCS template objects:

Table 4.9

Query Function	Description
istemplate()	verifies a template object

Creating VCS Objects

The create functions allow the user to create VCS objects which can be modified directly to produce the desired results. Since the VCS “default” objects do allow modifications, it is best to either create a new VCS object or get an existing one. When a VCS object is created, it is stored in an internal table for later use and/or recall.

Create the following VCS objects:

Table 4.10

Create	Description
createboxfill()	creates a new boxfill graphics method object
createcontinents()	creates a new continents graphics method object

Table 4.10

Create	Description
createfillarea()	creates a new fillarea secondary object
createisofill()	creates a new isofill graphics method object
createisoline()	creates a new isoline graphics method object
createline()	creates a new line secondary object
createmarker()	creates a new marker secondary object
createoutfill()	creates a new outfill graphics method object
createoutline()	creates a new outline graphics method object
createscatter()	creates a new scatter graphics method object
createtextcombined()	creates a new text-combined secondary object
createtextorientation()	creates a new text-orientation secondary object
createtexttable()	creates a new text-table secondary object
createvector()	creates a new vector graphics method object
createxvsy()	creates a new xvsy graphics method object
createxyvsy()	creates a new xyvsy graphics method object
createyxvsx()	creates a new xyvsy graphics method object

Get Existing VCS Objects

The get functions are used to obtain VCS objects that exist in the object memory tables. The get function directly manipulates the object's attributes in memory. If the object is used to display data on a plot and is manipulated by the user, then the plot will be automatically updated.

Get the following VCS objects:

Table 4.11

Get	Description
getboxfill()	get specified boxfill graphics method and create boxfill object
getcontinents()	get specified continents graphics method and create continents object

Table 4.11

Get	Description
getfillarea()	get specified fillarea secondary object and create fillarea object
getisofill()	get specified isofill graphics method and create fillarea object
getisoline()	get specified isoline graphics method and create isoline object
getline()	get specified line secondary object and create line object
getmarker()	get specified marker secondary object and create marker object
getoutfill()	get specified outfill graphics method and create outfill object
getoutline()	get specifed outline graphics method and create outline object
getscatter()	get specified scatter graphics method and create scatter object
gettextcombined()	get specified text-combined secondary object and create text-combined object
gettextorientation()	get specified text-orientation secondary object and create text-orientation object
gettexttable()	get specified text-table secondary object and create text-table object
getvector()	get specified vector graphics method and create vector object
getxvsy()	get specified xvsy graphics method and create xvsy object
getxyvsy()	get specified xyvsy graphics method and create xyvsy object
getyxvsx()	get specified yxvsx graphics method and create yxvsx

Removing VCS Objects

Unwanted VCS objects can be removed from internal memory with the use of the remove function. The remove function will identify the VCS object type and remove it from the appropriate object table.

Remove VCS objects:

Table 4.12

Remove	Description
removeobject()	allows the user to remove objects from the appropriate object list

Show VCS Object List

The show function is handy to list VCS objects tables.

The show function is used to list the VCS objects in memory:

Table 4.13

Show	Description
show()	list VCS primary and secondary class objects in memory

Saving VCS Object in a Script File

Script commands define the actions that are necessary to preserve an interactive session as a script and to mimic that session in a non-interactive replay of the script. Many attributes are needed to create a graphical representation of a variable, e.g. attributes to identify the variable and to label the plotting axes. By use of VCS and Python scripts, most of these attributes can be manipulated to create the desired visual effect, and the resulting attributes can be saved for later use. VCS and Python scripts also allow the user to save a sequence of interactive operations for replay, and to recover from a system failure.

After creating and/or modifying a VCS object, the user can save the object in the initial.attributes file. The initial.attributes file contains many predefined attribute settings to aid the beginning user of VCS. The path to the file must be:

/$HOME/PCMDI_GRAPHICS/initial.attributes

where /$HOME denotes the user's home directory. (Note, when VCS is executed for the first time, a /PCMDI_GRAPHICS subdirectory will be created automatically if one has not already been created.) The user also can customize the initial.attributes file directly.

To re-save the initial.attributes file, use the function:

Table 4.14

Save Initial Script	Description
saveinitialfile()	saves current VCS objects in the initial.attributes file, which is read initially at start-up

To save VCS objects has Python scripts or VCS scripts, use the function:

Table 4.15

Save VCS Objects	Description
scriptobject()	save individual VCS objects as Python or VCS script file

To save the state of the system, use the function:

Table 4.16

Save System State	Description
scriptstate()	saves a sequence of interactive operations for replay; or to recover from a system failure

To run a VCS script file, use the function:

Table 4.17

Run Script	Description
scriptrun()	run a previously saved VCS script file

CHAPTER 5

VCS Command Reference Guide

If you want the full description of a command, then you've made it to the right place.

Note, in the "Options" column, any item(s) surrounded by "[]" are optional to the function.

Command	Description	Options	Examples
Initializing			

init	**Function:** init
	# Initialize, Construct a VCS Canvas Object
	
	Description of Function:
	Construct the VCS Canas object. There can only be at most 8 VCS Canvases open at any given time.
			Example of Use:
			import vcs,cu
			
			file=cu.open('filename.nc')
			slab=file.getslab('variable')
			a=vcs.init()
			# This examples constructs 4 VCS Canvas
			a.plot(slab)
			# Plot slab using default settings
			b=vcs.init()
			# Construct VCS object
			template=b.gettemplate('AMIP')
			# Get 'example' template object
			b.plot(slab,template)
			# Plot slab using template 'AMIP'
			c=vcs.init()
			# Construct new VCS object
			isofill=c.getisofill('quick')
			# Get 'quick' isofill graphics method
			c.plot(slab,template,isofill)
			# Plot slab using template and isofill objects
			d=vcs.init()
			# Construct new VCS object
			isoline=c.getisoline('quick')
			
			# Get 'quick' isoline graphics method
			c.plot(isoline,slab,template)
			# Plot slab using isoline and template objects

Command	Description	Options	Examples
Help Commands			
help	**Function:** help # On-Line HELP!!! **Description of Function:** Gives insight to other VCS functions by providing a description and at least one example.		**Example of Use:** import vcs vcs.help() vcs.help('init') vcs.help('plot')

objecthelp	**Function:** objecthelp
	# Print out the object's doc string
	
	Description of Function:
	Print out information on VCS objects. See example on its use.
			Example of Use:
			import vcs
			a=vcs.init()
			ln=a.getline('red')
			# Get a VCS line object
			a.objecthelp(ln)
			# This will print out information on how to use ln

Command	Description	Options	Examples
		Canvas	

mode	**Function:** mode
	# Update the VCS Canvas.
	
	Description of Function:
	Updating of the graphical displays on the VCS Canvas can be deferred until a later time. This is helpful when generating templates or displaying numerous plots. If a series of commands are given to VCS and the Canvas Mode is set to manual (i.e., 0), then no updating of the VCS Canvas occurs until the 'update' function is executed.
	Note, by default the VCS Canvas Mode is set to 'Automatic', which means VCS will update the VCS Canvas as necessary without prompting from the user.
		1|0
		
		1 = automatic
		0=manual
			Example of Use:
			import vcs
			...
			a=vcs.init()
			a.mode=0
			# Set updating to manual mode
			a.plot(array,'default','boxfill','quick')
			box=x.getboxfill('quick')
			box.color_1=100
			box.xticlabels('lon30','lon30')
			box.xticlabels('','')
			box.datawc(1e20,1e20,1e20,1e20)
			box.datawc(-45.0, 45.0, -90.0, 90.0)
			...
			a.update()
			# Update the changes manually

Command	Description	Options	Examples

update	**Function:** update
	
	Description of Function:
	If a series of commands are given to VCS and the Canvas Mode is set to manual, then use this function to update the VCS Canvas manually.
			Example of Use:
			import vcs
			...
			a=vcs.init()
			a.mode=0
			# Go to manual mode
			a.plot(s,'default','boxfill','quick')
			box=x.getboxfill('quick')
			box.color_1=100
			box.xticlabels('lon30','lon30')
			box.xticlabels('','')
			box.datawc(1e20,1e20,1e20,1e20)
			box.datawc(-45.0, 45.0, -90.0, 90.0)
			a.update()
			# Update the changes manually

open	**Function:** open **Description of Function:** Open VCS Canvas object. This routine really just manages the VCS canvas. It will popup the VCS Canvas for viewing. It can be used to display the VCS Canvas.		**Example of Use:** import vcs a=vcs.init() a.open()
close	**Function:** close **Description of Function:** Close the VCS Canvas. It will remove the VCS Canvas object from the screen, but not deallocate it.		**Example of Use:** import vcs a=vcs.init() a.plot(array,'default','isofill','quick') a.close()

Command	Description	Options	Examples
portrait	**Function:** portrait **Description of Function:** Change the VCS Canvas orientation to Portrait.		**Example of Use:** a=vcs.init() a.plot(array) a.portrait() # Change the VCS Canvas orientation and set object flag to portrait
landscape	**Function:** landscape **Description of Function:** Change the VCS Canvas orientation to Landscape.		**Example of Use:** a=vcs.init() a.plot(array) a.landscape() # Change the VCS Canvas orientation and set object flag to landscape
page	**Function:** page **Description of Function:** Change the VCS Canvas orientation to either 'portrait' or 'landscape'. The orientation of the VCS Canvas and of cgm and raster images is controlled by the PAGE command. Only portrait (y > x) or landscape (x > y) orientations are permitted.		**Example of Use:** a=vcs.init() a.plot(array) a.page() # Change the VCS Canvas orientation and set object flag to portrait
geometry	**Function:** geometry **Description of Function:** The geometry command is used to set the size and position of the VCS canvas.	(w,h,x,y), where w=width, h=height, x=x_position, y=y_position	**Example of Use:** a=vcs.init() a.plot(array,'default','isofill','quick') a.geometry(450, 337,100, 100)

Command	Description	Options	Examples
Printing and Saving Graphics			

printer	**Function: printer**
	# Send plots to the printer
	
	Description of Function:
	This function creates a temporary cgm file and then sends it to the specified printer. Once the printer received the information, then the temporary cgm file is deleted. The temporary cgm file is created in the user's PCMDI_GRAPHICS directory.
	The PRINTER command is used to send the VCS Canvas plot(s) directly to the printer.
	Note: VCS graphical displays can be printed only if the user customizes a HARD_COPY file (included with the VCS software) for the home system. The path to the HARD_COPY file must be:
	/$HOME/ PCMDI_GRAPHICS/ HARD_COPY
	where /$HOME denotes the user's home directory.
	For more information on the HARD_COPY file, see URL:
	
	http://www-pcmdi.llnl.gov/software/vcs/vcs_guidetoc.html #1.Setup
		printer's name
		
		l|p -
		Orientation can be either: 'l' = landscape, or 'p' = portrait.
			Example of Use:
			import vcs
			a=vcs.init()
			a.plot(array)
			a.printer('printer_name')
			# Send plot(s) to postscript printer
			a.printer('printer_name','p')
			# Send plot(s) to the printer in portrait mode

Command	Description	Options	Examples

gif	**Function:** gif
	# Save plot(s) as gif image
	
	Description of Function:
	In some cases, the user may want to save the plot out as a gif image. This routine allows the user to save the VCS canvas output as a gif file.
	This file can be converted to other gif formats with the aid of xv and other such imaging tools found freely on the web.
	If no path/file name is given and no previously created gif file has been
	designated, then file
	/$HOME/ PCMDI_GRAPHICS/ default.gif
	will be used for storing gif images. However, if a previously created gif file is designated, that file will be used for gif output.
	By default, the page orientation is in Landscape mode (l). To translate the page orientation to portrait mode (p), enter 'p' as the second parameter.
	The GIF command is used to create or append to a gif file. There are two modes for saving a gif file: 'Append' mode (a) appends gif output to an existing gif file(i.e., making it an **animated** gif); 'Replace' (r) mode overwrites an existing gif file with new gif output.
		gif file name
		
		'a'=will append (or merge) image to an existing file, making it an **animated** gif
		'r'=will replace file with new image
		'l'=landscape mode
		'p'=portrait mode
			Example of Use:
			import vcs
			a=vcs.init()
			a.plot(array)
			# Note, if you don't specify the extension '.gif' at the end of file name, then the extension '.gif' will be put on for you.
			a.gif('example')
			# merge gif image into existing gif file
			a.gif('example','r')
			# over write existing gif file
			a.gif('example','a')
			# merge gif image into existing gif file
			a.gif('example','a','p')
			# merge gif image into existing gif file with portrait orientation
			a.gif('example.gif','r','p')
			# over write gif image file with new portrait orientation gif

Command	Description	Options	Examples

postscript	**Function:** postscript
	# Save plot(s) to a postscript file
	
	Description of Function:
	Postscript output is another form of vector graphics. It is larger than its CGM output counter part, because it is not stored out in ASCII format. To save out a postscript file, VCS will first create a cgm file in the user's PCMDI_GRAPHICS directory. Then it will use gplot to convert the cgm file to a postscript file in the location the user has chosen.
		postscript file name
		
		'a'=will append post-script to an existing file
		'r'=will replace postscript file with new apostscript file
			Example of Use:
			import vcs
			a=vcs.init()
			a.plot(array)
			# Note, if you don't specify the extension '.ps' at the end of file name, then the extension '.ps' will be put on for you.
			a.postscript('example')
			# Creates a landscape postscript file a.postscript('example','p')
			# Creates a portrait postscript file

cgm	**Function:** cgm
	
	Description of Function:
	To save a graphics plot in VCS the user can call CGM along with the name of the output. This routine will save the displayed image on the VCS canvas as a binary vector graphics that can be imported into MSWord or Framemaker. CGM files are in ISO standards output format.
	The CGM command is used to create or append to a cgm file. There are two modes for saving a cgm file: 'Append' mode (a) appends cgm output to an existing cgm file; 'Replace' (r) mode overwrites an existing cgm file with new cgm output.
		cgm file name
		
		'a'=will append cgm to an existing file
		'r'=will replace cgm file with a new cgm file
			Example of Use:
			a=vcs.init()
			a.plot(array,'default','isofill','quick')
			# Note, if you don't specify the extension '.cgm' at the end of file name, then the extension '.cgm' will be put on for you.
			a.cgm(o)
			a.cgm('example')
			# by default a cgm file will be appended it an existing file
			a.cgm('example','a')
			# 'a' will instruct cgm to append to an existing file
			a.cgm('example','r')
			# 'r' will instruct cgm to over write an existing file

Command	Description	Options	Examples

raster	**Function:** raster
	
	Description of Function:
	In some cases, the user may want to save the plot out as an raster image. This routine allows the user to save the VCS canvas output as a SUN raster file.
	This file can be converted to other raster formats with the aid of xv and other such imaging tools found freely on the web.
	If no path/file name is given and no previously created raster file has beendesignated, then file
	/$HOME/ PCMDI_GRAPHICS/ default.ras
	will be used for storing raster images. However, if a previously created raster file is designated, that file will be used for raster output.
		raster file name
		
		'a'=will append raster image to an existing raster file
		'r'=will replace a raster file with a new raster file
			Example of Use:
			import vcs
			a=vcs.init()
			a.plot(array)
			# Note, if you don't specify the extension '.ras' at the end of file name, then the extension '.ras' will be put on for you.
			a.raster('example','a')
			# append raster image to existing file
			a.raster('example','r')
			# over write existing raster file

pstogif	**Function:** pstogif **Description of Function:** This function allows the user to convert a postscript file to a gif file.	postscript file name ['l'=landscape 'p'=portrait]	**Example of Use:** import vcs a=vcs.init() a.plot(array) a.pstogif('filename.ps') # convert to landscape gif file a.pstogif('filename.ps','l') # convert to landscape gif file a.pstogif('filename.ps','p') # convert to portrait gif file

Command	Description	Options	Examples
Plot and Clear Commands			

plot	**Function:** plot
	
	Description of Function:
	Plot an array(s) of data given a template and graphics method. The VCS template is used to define where the data and variable attributes will be displayed on the VCS Canvas. The VCS graphics method is used to define how the array(s) will be shown on the VCS Canvas.
		The form of the call is:
		
		plot(array1=None, array2=None, template_name=None, graphics_method=None,graphics_name=None, [key=value [, key=value [, ...]]])
		where array1 and array2 are NumPy arrays, such that 2<=rank(ar)<=5.
		
		See section 4.5.1 for a detail listing of possible plot options.
			Example of Use:
			
			import vcs
			x=vcs.init()
			# x is an instance of the VCS class object (constructor)
			x.plot(array)
			# this call will use default settings for template and boxfill
			x.plot(array, 'AMIP', 'isofill','AMIP_psl')
			# this is specifying the template and graphics method
			t=x.gettemplate('AMIP')
			# get a predefined the template 'AMIP'
			vec=x.getvector('quick')
			# get a predefined the vector graphics method 'quick'
			x.plot(array1, array2, t, vec)
			# plot the data as a vector using the 'AMIP' template
			x.clear()
			# clear the VCS Canvas of all plots
			box=x.createboxfill('new')
			# create boxfill graphics method 'new'
			x.plot(box,t,array)
			# plot array data using box 'new' and template 't'

Command	Description	Options	Examples

boxfill	**Function:** boxfill
	# Generate a boxfill plot
	
	Description of Function:
	Generate a boxfill plot given the data, boxfill graphics method, and template. If no boxfill class object is given, then the 'default' boxfill graphics method is used. Similarly, if no template class object is given, then the 'default' template is used.
		See plot command for options.	**Example of Use:**
			import vcs
			a=vcs.init()
			a.show('boxfill')
			# Show all the existing boxfill graphics methods
			box=a.getboxfill('quick')
			# Create instance of 'quick'
			a.boxfill(array,box)
			# Plot array using specified box and default template
			templt=a.gettemplate('AMIP')
			# Create an instance of template 'AMIP'
			a.clear()
			# Clear VCS canvas
			a.boxfill(array,box,template)
			# Plot array using specified box and template
			a.boxfill(box,array,template)
			# Plot array using specified box and template
			a.boxfill(template,array,box)
			# Plot array using specified box and template
			a.boxfill(template,array,box)
			# Plot array using specified box and template
			a.boxfill(array,'AMIP','quick')
			# Use 'AMIP' template and 'quick' boxfill
			a.boxfill('AMIP',array,'quick')
			# Use 'AMIP' template and 'quick' boxfill
			a.boxfill('AMIP','quick',array)
			# Use 'AMIP' template and 'quick' boxfill

Command	Description	Options	Examples

continents	**Function:** continents
	# Generate a continents plot
	
	Description of Function:
	Generate a continents plot given the continents graphics method, and template. If no continents class object is given, then the 'default' continents graphics method is used. Similarly, if no template class object is given, then the 'default' template is used.
		See plot command for options.	**Example of Use:**
			import vcs
			a=vcs.init()
			a.show('continents')
			# Show all the existing continents graphics methods
			con=a.getcontinents('quick')
			# Create instance of 'quick'
			a.continents(array,con)
			# Plot array using specified con and default template
			a.clear()
			# Clear VCS canvas
			a.continents(array,con,template)
			# Plot array using specified con and template

isofill	**Function:** isofill
	# Generate an isofill plot
	
	Description of Function:
	Generate a isofill plot given the data, isofill graphics method, and template. If no isofill class object is given, then the 'default' isofill graphics method is used. Similarly, if no template class object is given, then the 'default' template is used.
		See plot command for options.	**Example of Use:**
			import vcs
			a=vcs.init()
			a.show('isofill')
			# Show all the existing isofill graphics methods
			iso=a.getisofill('quick')
			# Create instance of 'quick'
			a.isofill(array,iso)
			# Plot array using specified iso and defaul template
			a.clear()
			# Clear VCS canvas
			a.isofill(array,iso,template)
			# Plot array using specified iso and template

Command	Description	Options	Examples

isoline	**Function:** isoline
	# Generate an isoline plot
	
	Description of Function:
	Generate a isoline plot given the data, isoline graphics method, and template. If no isoline class object is given, then the 'default' isoline graphics method is used. Similarly, if no template class object is given, then the 'default' template is used.
		See plot command for options.	**Example of Use:**
			a=vcs.init()
			a.show('isoline')
			# Show all the existing isoline graphics methods
			iso=a.getisoline('quick')
			# Create instance of 'quick'
			a.isoline(array,iso)
			# Plot array using specified iso and default template
			a.clear()
			# Clear VCS canvas a.isoline(array,iso,template)
			# Plot array using specified iso and template

outfill	**Function:** outfill
	# Generate an outfill plot
	
	Description of Function:
	Generate a outfill plot given the data, outfill graphics method, and template. If no outfill class object is given, then the 'default' outfill graphics method is used. Similarly, if no template class object is given, then the 'default' template is used.
		See plot command for options.	**Example of Use:**
			a=vcs.init()
			a.show('outfill')
			# Show all the existing outfill graphics methods
			out=a.getoutfill('quick')
			# Create instance of 'quick'
			a.outfill(array,out)
			# Plot array using specified out and default template
			a.clear()
			# Clear VCS canvas
			a.outfill(array,out,template)
			# Plot array using specified out and template

Command	Description	Options	Examples

outline	**Function:** outline
	# Generate an outline plot
	
	Description of Function:
	Generate a outline plot given the data, outline graphics method, and template. If no outline class object is given, then the 'default' outline graphics method is used. Similarly, if no template class object is given, then the 'default' template is used.
		See plot command for options.	**Example of Use:**
			import vcs
			a=vcs.init()
			a.show('outline')
			# Show all the existing outline graphics methods
			out=a.getoutline('quick')
			# Create instance of 'quick'
			a.outline(array,out)
			# Plot array using specified out and default template
			a.clear()
			# Clear VCS canvas a.outline(array,out,template)
			# Plot array using specified out and template

scatter	**Function:** scatter
	# Generate a scatter plot
	
	Description of Function:
	Generate a scatter plot given the data, scatter graphics method, and template. If no scatter class object is given, then the 'default' scatter graphics method is used. Similarly, if no template class object is given, then the 'default' template is used.
		See plot command for options.	**Example of Use:**
			a=vcs.init()
			a.show('scatter')
			# Show all the existing scatter graphics methods
			sct=a.getscatter('quick')
			# Create instance of 'quick'
			a.scatter(array1,array2,sct)
			# Plot array using specified sct and default template
			a.clear()
			# Clear VCS canvas
			a.scatter(array1,array2,sct,template)
			# Plot array using specified sct and template

Command	Description	Options	Examples

vector	**Function:** vector
	# Generate a vector plot
	
	Description of Function:
	Generate a vector plot given the data, vector graphics method, andtemplate. If no vector class object is given, then the 'default' vectorgraphics method is used. Similarly, if no template class object is given,then the 'default' template is used.
		See plot command for options.	**Example of Use:**
			import vcs
			a=vcs.init()
			a.show('vector')
			# Show all the existing vector graphics methods
			vec=a.getvector('quick')
			# Create instance of 'quick'
			a.vector(array1,array2,vec)
			# Plot array using specified vec and default template
			a.clear()
			# Clear VCS canvas
			a.vector(array1,array2,vec,template)
			# Plot array using specified vec and template

xvsy	**Function:** xvsy
	# Generate a Xvsy plot
	
	Description of Function:
	Generate a XvsY plot given the data, XvsY graphics method, and template. If no XvsY class object is given, then the 'default' XvsY graphics method is used. Similarly, if no template class object is given, then the 'default' template is used.
		See plot command for options.	**Example of Use:**
			import vcs
			a=vcs.init()
			a.show('xvsy')
			# Show all the existing XvsY graphics methods
			xy=a.getxvsy('quick')
			# Create instance of 'quick'
			a.xvsy(array1,array2,xy)
			# Plot array using specified xy and default template
			a.clear()
			# Clear VCS canvas
			a.xvsy(array1,array2,xy,template)
			# Plot array using specified xy and template

Command	Description	Options	Examples

xyvsy	**Function:** xyvsy
	# Generate a Xyvsy plot
	
	Description of Function:
	Generate a Xyvsy plot given the data, Xyvsy graphics method, and template. If no Xyvsy class object is given, then the 'default' Xyvsy graphics method is used. Simerly, if no template class object is given, then the 'default' template is used.
		See plot command for options.	**Example of Use:**
			import vcs
			a=vcs.init()
			a.show('xyvsy')
			# Show all the existing Xyvsy graphics methods
			xyy=a.getxyvsy('quick')
			# Create instance of 'quick'
			a.xyvsy(array,xyy)
			# Plot array using specified xyy and default template
			a.clear()
			# Clear VCS canvas
			a.xyvsy(array,xyy,template)
			# Plot array using specified xyy and template

yxvsx	**Function:** yxvsx
	# Generate a Yxvsx plot
	
	Description of Function:
	Generate a Yxvsx plot given the data, Yxvsx graphics method, and template. If no Yxvsx class object is given, then the 'default' Yxvsx graphics method is used. Simerly, if no template class object is given, then the 'default' template is used.
		See plot command for options.	**Example of Use:**
			import vcs
			a=vcs.init()
			a.show('yxvsx')
			# Show all the existing Yxvsx graphics methods
			yxx=a.getyxvsx('quick')
			# Create instance of 'quick'
			a.yxvsx(array,yxx)
			# Plot array using specified yxx and default template
			a.clear()
			# Clear VCS canvas
			a.yxvsx(array,yxx,template)
			# Plot array using specified yxx and template

Command	Description	Options	Examples
clear	**Function:** clear # Generate a Yxvsx plot **Description of Function:** At some point it will become necessary to clear all the plots from the VCS Canvas. This routine will remove all plots on the VCS Canvas.		**Example of Use:** import vcs a=vcs.init() a.plot(array,'default','isofill','quick') a.clear()
Query Functions			

graphics-method-name	**Function:** graphicsmethod-name
	
	Description of Function:
	Returns the grapics method's type string: boxfill, isofill, isoline, outfill, outline, continents, scatter, vector, xvsy, xyvsy, or yxvsx.
			Example of Use:
			import vcs
			a=vcs.init()
			box=a.getboxfill() # Get an default boxfill
			iso=a.getisofill() # Get default isofill
			ln=a.getline() # Get default line
			print a.graphicsmethodname(box) # Will
			# print 'boxfill'
			print a.graphicsmethodname(iso) # Will
			# print 'isofill'
			print a.graphicsmethodname(ln) # Will return
			# None, because ln is not a
			# graphics method

getconti-nentstype	**Function:** getcontinentstype **Description of Function:** Return continents type from VCS. Remember the value can only be between 0 and 11.		**Example of Use:** import vcs a=vcs.init() cont_type = a.getcontinentstype() # Get the # continents type

Command	Description	Options	Examples
isgraphics-method	**Function:** isgraphicsmethod **Description of Function:** Indicates if the entered argument is one of the following graphics methods: boxfill, isofill, isoline, outfill, outline, continents, scatter, vector, xvsy, xyvsy, yxvsx. Returns a 1, which indicates true, if the argment is one of the above. Otherwise, it will return a 0, indicating false.		**Example of Use:** import vcs a=vcs.init() box=a.getboxfill('quick') # To Modify an existing boxfill use: ... if a.isgraphicsmethod(box): box.list()
isboxfill	**Function:** isboxfill **Description of Function:** Check to see if an object is a VCS primary boxfill graphics method object. Returns a 1 if the argment true. Otherwise, it will return a 0, indicating false.		**Example of Use:** import vcs a=vcs.init() box=a.getboxfill("quick") # To Modify an existing boxfill object ... if a.isboxfill(box): box.list()
iscontinents	**Function:** iscontinents **Description of Function:** Check to see if an object is a VCS primary continents graphics method. Returns a 1 if the argment true. Otherwise, it will return a 0, indicating false.		**Example of Use:** import vcs a=vcs.init() con=a.getcontinents("quick") # To Modify an existing continents object ... if a.iscontinents(con): con.list()

Command	Description	Options	Examples
isisofill	**Function:** isisofill **Description of Function:** Check to see if an object is a VCS primary isofill graphics method. Returns a 1 if the argment true. Otherwise, it will return a 0, indicating false.		**Example of Use:** import vcs a=vcs.init() iso=a.getisofill("quick") # To Modify an existing isofill object ... if a.isisofill(iso): iso.list()
isisoline	**Function:** isisoline **Description of Function:** Check to see if an object is a VCS primary isoline graphics method. Returns a 1 if the argment true. Otherwise, it will return a 0, indicating false.		**Example of Use:** import vcs a=vcs.init() iso=a.getisoline("quick") # To Modify an existing isoline object ... if a.isisoline(iso): iso.list()
isoutfill	**Function:** isoutfill **Description of Function:** Check to see if this object is a VCS primary outfill graphics method. Returns a 1 if the argment true. Otherwise, it will return a 0, indicating false.		**Example of Use:** import vcs a=vcs.init() out=a.getoutfill("quick") # To Modify an existing outfill object ... if a.isoutfill(out): out.list()

Command	Description	Options	Examples
isoutline	**Function:** isoutline **Description of Function:** Check to see if an object is a VCS primary outline graphics method. Returns a 1 if the argment true. Otherwise, it will return a 0, indicating false.		**Example of Use:** import vcs a=vcs.init() out=a.getoutline("quick") # To Modify an existing outline object ... if a.isoutline(out): out.list()
isvector	**Function:** isvector **Description of Function:** Check to see if an object is a VCS primary vector graphics method. Returns a 1 if the argment true. Otherwise, it will return a 0, indicating false.		**Example of Use:** import vcs a=vcs.init() vec=a.getvector("quick") # To Modify an existing vector object ... if a.isvector(vec): vec.list()
isxvsy	**Function:** isxvsy **Description of Function:** Check to see if an object is a VCS primary xvsy graphics method. Returns a 1 if the argment true. Otherwise, it will return a 0, indicating false.		**Example of Use:** import vcs a=vcs.init() xy=a.getxvsy("quick") # To Modify an existing xvsy object ... if a.isxvsy(xy): xy.list()

Command	Description	Options	Examples
isxyvsy	**Function:** isxyvsy **Description of Function:** Check to see if an object is a VCS primary Xyvsy graphics method. Returns a 1 if the argment true. Otherwise, it will return a 0, indicating false.		**Example of Use:** import vcs a=vcs.init() xyy=a.getxyvsy("quick") # To Modify an existing Xyvsy object ... if a.isxyvsy(xyy): xyy.list()
isyxvsx	**Function:** isyxvsx **Description of Function:** Check to see if an object is a VCS primary yxvsx graphics method. Returns a 1 if the argment true. Otherwise, it will return a 0, indicating false.		**Example of Use:** a=vcs.init() yxx=a.getyxvsx("quick") # To Modify an existing yxvsx object ... if a.isyxvsx(yxx): yxx.list()
istemplate	**Function:** istemplate **Description of Function:** Indicates if the entered argument a template. Returns a 1 if the argment true. Otherwise, it will return a 0, indicating false.		**Example of Use:** a=vcs.init() templt=a.gettemplate('quick') # Modify an existing template named 'quick' ... if a.istemplate(templt): templt.list() # If it is a template then list its members

Command	Description	Options	Examples
issecond-aryobject	**Function:** issecondaryobject **Description of Function:** In addition, detailed specification of the primary elements' (or primary class elements'), attributes is provided by eight secondary elements or (secondary class elements): 1.) colormap: specification of combinations of 256 available colors 2.) fill area: style, style index, and color index 3.) format: specifications for converting numbers to display strings 4.) line: line type, width, and color index 5.) list: a sequence of pairs of numerical and character values 6.) marker: marker type, size, and color index 7.) text table: text font type, character spacing, expansion, and color index 8.) text orientation: character height, angle, path, and horizontal/vertical alignment		**Example of Use:** a=vcs.init() line=a.getline('red') # To Modify an existing line object ... if a.issecondaryobject(line): box.list()
isfillarea	**Function:** isfillarea **Description of Function:** Check to see if an object is a VCS secondary fillarea.		**Example of Use:** a=vcs.init() fa=a.getfillarea("def37") # To Modify an existing fillarea object ... if a.isfillarea(fa): fa.list()

Command	Description	Options	Examples
isline	**Function:** isline **Description of Function:** Check to see if this object is a VCS secondary line.		**Example of Use:** a=vcs.init() ln=a.getline("red") # To Modify an existing line object ... if a.isline(ln): ln.list()
ismarker	**Function:** ismarker **Description of Function:** Check to see if an object is a VCS secondary marker.		**Example of Use:** a=vcs.init() mk=a.getmarker("red") # To Modify an existing marker object ... if a.ismarker(mk): mk.list()

istextcombined	**Function:** istextcombined
	
	Description of Function:
	Check to see if an object is a VCS secondary text combined.
			Example of Use:
			a=vcs.init()
			tc=a.gettextcombined("std", "7left")
			# To Modify existing text table and orientation objects
			...
			if a.istextcombined(tc):
			tc.list()
			if a.istexttable(tc):
			tc.list()
			if a.istextorientation(tc):
			tc.list()

istextorientation	**Function:** istextorientation **Description of Function:** Check to see if an object is a VCS secondary text orientation.		**Example of Use:** a=vcs.init() to=a.gettextorientation("7left") # To Modify an existing text orientation object ... if a.istextorientation(to): to.list()

Command	Description	Options	Examples
istexttable	**Function:** istexttable **Description of Function:** Check to see if an object is a VCS secondary text table.		**Example of Use:** a=vcs.init() tt=a.gettexttable("std") # To Modify an existing text table object ... if a.istexttable(tt): tt.list()
List Available Templates, Graphics Methods and Secondary Objects			
listele-ments	**Function:** listelements **Description of Function:** Returns a Python list of all the VCS class objects.		**Example of Use:** import vcs a=vcs.init() a.listelements()
show	**Function:** show **Description of Function:** Show the list of VCS primary and secondary class objects.		**Example of Use:** import vcs a=vcs.init() a.show('boxfill') a.show('isofill') a.show('template') a.show('line') a.show('marker')

Command	Description	Options	Examples
Graphics Method Objects			
Boxfill			
createbox-fill	**Function:** createboxfill **Description of Function:** Create a new boxfill graphics method given the name and the existingboxfill graphics method to copy the attributes from. If no existing boxfill graphics method name is given, then the default boxfill graphics method will be used as the graphics method to which the attributes will be copied from. If the name provided already exists, then a error will be returned. Graphics method names must be unique.	new boxfill name [name of box-fill to copy attributes from]	**Example of Use:** import vcs a=vcs.init() a.show('boxfill') box=a.createboxfill('example1') a.show('boxfill') box=a.createboxfill('example2','quick') a.show('boxfill')

getboxfill	**Function:** getboxfill
	
	Description of Function:
	VCS contains a list of graphics methods. This function will create a boxfill class object from an existing VCS boxfill graphics method. If no boxfill name is given, then boxfill 'default' will be used.
	Note, VCS does not allow the modification of 'default' attribute sets. However, a 'default' attribute set that has been copied under a different name can be modified. (See the createboxfill function.)
		[boxfill name]	**Example of Use:**
			import vcs
			a=vcs.init()
			a.show('boxfill')
			# Show all the existing boxfill graphics methods
			box=a.getboxfill()
			# box instance of 'default' boxfill graphics method
			box2=a.getboxfill('quick')
			# box2 instance of existing 'quick' boxfill graphics method

Command	Description	Options	Examples

	Object: boxfillobject
	
	Description of Function:
	The boxfill graphics method (Gfb) displays a two-dimensional data array by surrounding each data value by a colored grid box.
	This class is used to define a boxfill table entry used in VCS, or it can be used to change some or all of the attributes in an existing boxfill table entry.
	Other Useful Functions:
	a=vcs.init()
	# Constructor
	a.show('boxfill')
	# Show predefined boxfill graphics methods
	a.setcolormap("AMIP")
	# Change the VCS color map
	a.boxfill(s,b,'default')
	# Plot data 's' with boxfill 'b' and 'default' template
	a.update()
	# Updates the VCS Canvas at user's request
	a.mode=1, or 0
	# If 1, then automatic update, else if 0, then use the update function to update the VCS Canvas.
	See Chapter 6 for additional information.
		name
		projection
		xticlabels1
		xticlabels2
		xmtics1
		xmtics2
		yticlabels1
		yticlabels2
		ymtics1
		ymtics2
		datawc_x1
		datawc_y1
		datawc_x2
		datawc_y2
		xaxisconvert
		yaxisconvert
		level_1
		level_2
		color_1
		color_2
		legend_type
		legend
		ext_1
		ext_2
		missing
			Example of Use:
			import vcs
			a=vcs.init()
			
			# To Create a new instance of boxfill use:
			box=a.createboxfill('new','quick')
			# Copies content of 'quick' to 'new'
			box=a.createboxfill('new')
			# Copies content of 'default' to 'new'
			
			# To Modify an existing boxfill use:
			box=a.getboxfill('AMIP_psl') box.list()
			# Will list all the boxfill attribute values
			box.projection='linear'
			lon30={-180:'180W',-150:'150W',0:'Eq'}
			box.xticlabels1=lon30
			box.xticlabels2=lon30
			box.xticlabels(lon30, lon30)
			# Will set them both
			box.xmtics1=''
			box.xmtics2=''
			box.xmtics(lon30, lon30)
			# Will set them both
			box.yticlabels1=lat10
			box.yticlabels2=lat10
			box.yticlabels(lat10, lat10)
			# Will set them both
			box.ymtics1='' box.ymtics2=''
			box.ymtics(lat10, lat10)
			# Will set them both
			box.datawc_y1=-90.0 box.datawc_y2=90.0
			box.datawc_x1=-180.0
			box.datawc_x2=180.0 box.datawc(-90,90,-180,180)
			box.exts('n', 'y')
			# Will set them both

Command	Description	Options	Examples

			# Will set them all
			xaxisconvert='linear'
			yaxisconvert='linear'
			box.xyscale('linear', 'area_wt')
			# Will set them both
			level_1=1e20
			level_2=1e20
			box.levels(10, 90)
			# Will set them both
			color_1=16
			color_2=239
			box.colors(16, 239)
			# Will set them both
			legend_type='VCS'
			legend=''
			ext_1='n'
			ext_2='y'
			missing=241
			# Color index value range 0 to 255

Command	Description	Options	Examples
	Continents		
createcontinents	**Function:** createcontinents **Description of Function:** Create a new continents graphics method given the the name and the existing continents graphics method to copy the attributes from. If no existing continents graphics method name is given, then the default continents graphics method will be used as the graphics method to which the attributes will be copied from. If the name provided already exists, then a error will be returned. Graphics method names must be unique.	new continents name [name of continents to copy attributes from]	**Example of Use:** import vcs a=vcs.init() a.show('continents') con=a.createcontinents('example1',) a.show('continents') con=a.createcontinents('example2','quick') a.show('continents')

getcontinents	**Function:** getcontinents
	
	Description of Function:
	VCS contains a list of graphics methods. This function will create a continents class object from an existing VCS continents graphics method. If no continents name is given, then continents 'default' will be used.
	Note, VCS does not allow the modification of 'default' attribute sets. However, a 'default' attribute set that has been copied under a different name can be modified. (See the createcontinents function.)
		[continents name]	**Example of Use:**
			import vcs
			a=vcs.init()
			a.show('continents')
			# Show all the existing continents graphics methods
			con=a.getcontinents()
			# con instance of 'default' continents graphics method
			con2=a.getcontinents('quick')
			# con2 instance of existing 'quick' continents graphics method

Command	Description	Options	Examples

	Object: continentsobject
	
	Description of Function:
	The Continents graphics method draws a predefined, generic set of continental outlines in a longitude by latitude space. (To draw continental outlines, no external data set is required.)
	This class is used to define an continents table entry used in VCS, or it can be used to change some or all of the continents attributes in an existing continents table entry.
	Other Useful Functions:
	a=vcs.init()
	# Constructor
	a.show('continents')
	# Show predefined boxfill graphics methods
	a.show('line')
	# Show predefined line class objects
	a.setcolormap("AMIP")
	# Change the VCS color map
	a.continents(c,'default')
	# Plot continents, where 'c' is the defined continents object
	a.update()
	# Updates the VCS Canvas at user's request a.mode=1, or 0. If 1, then automatic update, else if 0, then use update function to update the VCS Canvas.
		name
		projection
		xticlabels1
		xticlabels2
		xmtics1
		xmtics2
		yticlabels1
		yticlabels2
		ymtics1
		ymtics2
		datawc_x1
		datawc_y1
		datawc_x2
		datawc_y2
		line
		
		linecolor
		type
			Example of Use:
			import vcs
			a=vcs.init()
			
			# To Create a new instance of continents use:
			con=a.createcontinents('new','quick')
			#copies content of 'quick' to 'new'
			con=a.createcontinents('new')
			# copies content of 'default' to 'new'
			
			# To Modify an existing continents use:
			con=a.getcontinents('AMIP_psl')
			con.list()
			# Will list all the continents attribute values
			con.projection='linear'
			lon30={-180:'180W',-150:'150W',0:'Eq'}
			con.xticlabels1=lon30
			con.xticlabels2=lon30
			con.xticlabels(lon30, lon30)
			# Will set them both
			con.xmtics1=''
			con.xmtics2=''
			con.xmtics(lon30, lon30)
			# Will set them both
			con.yticlabels1=lat10
			con.yticlabels2=lat10
			con.yticlabels(lat10, lat10)
			# Will set them both
			con.ymtics1=''
			con.ymtics2=''
			con.ymtics(lat10, lat10)

Command	Description	Options	Examples

	See Chapter 6 for additional information.		# Will set them both
			con.datawc_y1=-90.0
			con.datawc_y2=90.0
			con.datawc_x1=-180.0
			con.datawc_x2=180.0
			con.datawc(-90, 90, -180, 180) # Will set them all
			# Specify the continents line style (or type):
			con.line=0 # Same as con.line='solid'
			con.line=1 # Same as con.line='dash'
			con.line=2 # Same as con.line='dot'
			con.line=3 # Same as con.line='dash-dot'
			con.line=4 # Same as con.line='long-dash'
			# There are three possibilities for setting the line #color indices (Ex):
			con.linecolor=22 # Same as con.line-color=(22)
			con.linecolor=([22])
			# Will set the continents to a specific color index
			con.linecolor=None # Turns off the line color index, defaults to Black

Command	Description	Options	Examples
	Isofill		
createiso-fill	**Function:** createisofill **Description of Function:** Create a new isofill graphics method given the the name and the existing isofill graphics method to copy the attributes from. If no existing isofill graphics method name is given, then the default isofill graphics method will be used as the graphics method to which the attributes will be copied from. If the name provided already exists, then a error will be returned. Graphics method names must be unique.	new isofill name [name of iso-fill to copy attributes from]	**Example of Use:** import vcs a=vcs.init() a.show('isofill') iso=a.createisofill('example1',) a.show('isofill') iso=a.createisofill('example2','quick') a.show('isofill')

getisofill	**Function:** getisofill
	
	Description of Function:
	VCS contains a list of graphics methods. This function will create a isofill class object from an existing VCS isofill graphics method. If no isofill name is given, then isofill 'default' will be used.
	Note, VCS does not allow the modification of 'default' attribute sets. However, a 'default' attribute set that has been copied under a different name can be modified. (See the createisofill function.)
		[continents name]	**Example of Use:**
			import vcs
			a=vcs.init()
			a.show('isofill')
			# Show all the existing isofill graphics methods
			iso=a.getisofill()
			# iso instance of 'default' isofill graphics method
			iso2=a.getisofill('quick')
			# iso2 instance of existing 'quick' isofill graphics method

Command	Description	Options	Examples

	Object: isofillobject
	
	Description of Function:
	The Isofill graphics method fills the area between selected isolevels (levels of constant value) of a two-dimensional array with a user-specified color. The example below shows how to display an isofill plot on the VCS Canvas and how to create and remove isofill isolevels.
	This class is used to define an isofill table entry used in VCS, or it can be used to change some or all of the isofill attributes in anexisting isofill table entry.
	Other Useful Functions:
	a=vcs.init()
	# Constructor
	a.show('isofill')
	# Show predefined isofill graphics methods
	a.show('fillarea')
	# Show predefined fillarea objects
	a.show('template')
	# Show predefined fillarea objects
	a.setcolormap("AMIP")
	# Change the VCS color map
	a.createtemplate('test')
	# Create a template
		name
		projection
		xticlabels1
		xticlabels2
		xmtics1
		xmtics2
		yticlabels1
		yticlabels2
		ymtics1
		ymtics2
		datawc_x1
		datawc_y1
		datawc_x2
		datawc_y2
		xaxisconvert
		yaxisconvert
		missing
		ext_1
		ext_2
		fillareaindices
		fillareastyle
		fillareacolors
		levels
			Example of Use:
			import vcs
			a=vcs.init()
			
			# To Create a new instance of isofill use:
			iso=a.createisofill('new','quick')
			# Copies content of 'quick' to 'new'
			iso=a.createisofill('new')
			# Copies content of 'default' to 'new'
			
			# To Modify an existing isofill use:
			iso=a.getisofill('AMIP_psl')
			
			iso.list()
			# Will list all the isofill attribute values
			iso.projection='linear'
			lon30={-180:'180W',-150:'150W',0:'Eq'}
			iso.xticlabels1=lon30
			iso.xticlabels2=lon30
			iso.xticlabels(lon30, lon30)
			# Will set them both
			iso.xmtics1=''
			iso.xmtics2=''
			iso.xmtics(lon30, lon30)
			# Will set them both
			iso.yticlabels1=lat10
			iso.yticlabels2=lat10
			iso.yticlabels(lat10, lat10)
			# Will set them both
			iso.ymtics1=''
			iso.ymtics2=''
			iso.ymtics(lat10, lat10)

Command	Description	Options	Examples

	a.createfillarea('fill') # Create a fillarea
	a.gettemplate('AMIP') # Get an existing template
	a.getfillarea('def37') # Get an existing fillarea
	a.isofill(s,i,t) # Plot array 's' with isofill 'i' and template 't'
	a.update() # Updates the VCS Canvas at user's request a.mode=1, or 0. If 1, then automatic update,
	else if 0, then use update function to update the VCS Canvas.
	See Chapter 6 for additional information.
			# Will set them both
			iso.datawc_y1=-90.0
			iso.datawc_y2=90.0
			iso.datawc_x1=-180.0
			iso.datawc_x2=180.0
			iso.datawc(-90, 90, -180, 180) # Will set them all
			xaxisconvert='linear'
			yaxisconvert='linear'
			iso.xyscale('linear', 'area_wt') # Will set them both
			missing=241 # Color index value range 0 to 255
			# There are two possibilities for setting the isofill levels:
			# A) Levels are all contiguous (Examples):
			iso.levels=([0,20,25,30,35,40],)
			iso.levels=([0,20,25,30,35,40,45,50])
			iso.levels=[0,20,25,30,35,40]
			iso.levels=(0.0,20.0,25.0,30.0,35.0,40.0,50.0)
			# B) Levels are not contiguous (Examples):
			iso.levels=([0,20],[30,40],[50,60])
			iso.levels=([0,20,25,30,35,40],[30,40],[50,60])
			iso.fillareaindices=(7,fill,4,9,fill,15) # Set index using fillarea
			fill.list() # list fillarea attributes
			fill.style='hatch' # change style
			fill.color=241 # change color
			fill.index=3 # change style index
			ext_1='n'

Command	Description	Options	Examples

			ext_2='y'
			iso.exts('n', 'y') # Will set them both
			
			There are three possibilities for setting the fillarea color indices (Ex):
			iso.fillareacol-ors=([22,33,44,55,66,77])
			iso.fillareacolors=(16,19,33,44)
			iso.fillareacolors=None
			
			There are three possibilities for setting the fillarea style (Ex):
			iso.fillareastyle = 'solid'
			iso.fillareastyle = 'hatch'
			iso.fillareastyle = 'pattern'
			There are two ways to set the fillarea hatch or pattern indices (Ex):
			iso.fillareaindices=([1,3,5,6,9,20])
			iso.fillareaindices=(7,1,4,9,6,15)
			See using fillarea objects below!
			
			Using the fillarea secondary object (Ex):
			f=createfillarea('fill1')
			To Create a new instance of fillarea use:
			fill=a.createisofill('new','quick') # Copies 'quick' to 'new'
			fill=a.createisofill('new') # Cop-ies 'default' to 'new'
			
			To Modify an existing isofill use:
			fill=a.getisofill('def37')

Command	Description	Options	Examples
		Isoline	
createiso-line	**Function:** createisoline **Description of Function:** Create a new isoline graphics method given the the name and the existing isoline graphics method to copy the attributes from. If no existing isoline graphics method name is given, then the default isoline graphics method will be used as the graphics method to which the attributes will be copied from. If the name provided already exists, then a error will be returned. Graphicsmethod names must be unique.	new isoline name [name of iso-line to copy attributes from]	**Example of Use:** import vcs a=vcs.init() a.show('isoline') iso=a.createisoline('example1',) a.show('isoline') iso=a.createisoline('example2','quick') a.show('isoline')

getisoline	**Function:** getisoline
	
	Description of Function:
	VCS contains a list of graphics methods. This function will create a isoline class object from an existing VCS isoline graphics method. If no isoline name is given, then isoline 'default' will be used.
	Note, VCS does not allow the modification of 'default' attribute sets. However, a 'default' attribute set that has been copied under a different name can be modified. (See the createisoline function.)
		[isoline name]	**Example of Use:**
			import vcs
			a=vcs.init()
			a.show('isoline')
			# Show all the existing isoline graphics methods
			iso=a.getisoline()
			# iso instance of 'default' isoline graphics method
			iso2=a.getisoline('quick')
			# iso2 instance of existing 'quick' isoline graphics method

Command	Description	Options	Examples

	Object: isolineobject
	
	Description of Function:
	The Isoline graphics method draws lines of constant value at specified levels in order to graphically represent a two-dimensional array. It also labels the values of these isolines on the VCS Canvas. The example below shows how to plot isolines of different types at specified levels and how to create isoline labels having user-specified text and line type and color.
	This class is used to define an isoline table entry used in VCS, or it can be used to change some or all of the isoline attributes in an existing isoline table entry.
	Other Useful Functions:
	a=vcs.init()
	# Constructor
	a.show('isoline')
	# Show predefined isoline graphics methods
	a.show('line')
	# Show predefined VCS line objects
		name
		projection
		xticlabels1
		xticlabels2
		xmtics1
		xmtics2
		yticlabels1
		yticlabels2
		ymtics1
		ymtics2
		datawc_x1
		datawc_y1
		datawc_x2
		datawc_y2
		xaxisconvert
		yaxisconvert
		label
		line
		linecolors
		text
		textcolors
		level
			Example of Use:
			import vcs
			a=vcs.init()
			
			# To Create a new instance of isoline use:
			iso=a.createisoline('new','quick')
			# Copies content of 'quick' to 'new'
			iso=a.createisoline('new')
			# Copies content of 'default' to 'new'
			
			# To Modify an existing isoline use:
			iso=a.getisoline('AMIP_psl')
			iso.list()
			# Will list all the isoline attribute values
			iso.projection='linear'
			lon30={-180:'180W',-150:'150W',0:'Eq'}
			iso.xticlabels1=lon30
			iso.xticlabels2=lon30
			iso.xticlabels(lon30, lon30)
			# Will set them both
			iso.xmtics1=''
			iso.xmtics2=''
			iso.xmtics(lon30, lon30)
			# Will set them both
			iso.yticlabels1=lat10
			iso.yticlabels2=lat10
			iso.yticlabels(lat10, lat10)

Command	Description	Options	Examples

	a.update()
	# Updates the VCS Canvas at user's request
	a.mode=1, or 0
	# If 1, then automatic update, else if 0, then use update function.
	a.setcolormap("AMIP")
	# Change the VCS color map
	a.isoline(s,a,'default')
	# Plot data 's' with isoline 'i' and 'default' template
	tion to update the VCS Canvas.
	See Chapter 6 for additional information.
			# Will set them both
			iso.datawc_y1=-90.0
			iso.datawc_y2=90.0
			iso.datawc_x1=-180.0
			iso.datawc_x2=180.0
			iso.datawc(-90, 90, -180, 180) # Will set them all
			xaxisconvert='linear'
			yaxisconvert='linear'
			iso.xyscale('linear', 'area_wt') # Will set them both
			# There are many possibilities ways to set the isoline values:
			# A) As a list of tuples (Examples):
			iso.level=[(23,32,45,50,76),]
			iso.level=[(22,33,44,55,66)]
			iso.level=[(20,0.0),(30,0),(50,0)]
			iso.level=[(23,32,45,50,76), (35, 45, 55)]
			# B) As a tuple of lists (Examples):
			iso.level=([23,32,45,50,76],)
			iso.level=([22,33,44,55,66])
			iso.level=([23,32,45,50,76],)
			iso.level=([0,20,25,30,35,40],[30,40],[50,60])
			# C) As a list of lists (Examples):
			iso.level=[[20,0.0],[30,0],[50,0]]
			# D) As a tuple of tuples (Examples):
			iso.level=((20,0.0),(30,0),(50,0),(60,0),(70,0))

Command	Description	Options	Examples

			# Note: a combination of a pair (i.e., (30,0) or [30,0]) represents
			the isoline value plus it increment value. Thus, to let VCS
			generate "default" isolines enter the following:
			iso.level=[[0,1e20]]
			# Same as iso.level=((0,1e20),)
			
			Displaying isoline labels:
			iso.label='y'
			# Same as iso.label=1, will display isoline labels
			iso.label='n'
			# Same as iso.label=0, will turn isoline labels off
			# color index
			iso.linecolors=None # Turns off the line color index
			
			There are three ways to specify the text or font number:
			iso.text=(1,2,3,4,5,6,7,8,9) # Font numbers are between 1 and 9
			iso.text=[9,8,7,6,5,4,3,2,1]
			iso.text=([1,3,5,6,9,2])
			iso.text=None # Removes the text settings
			There are three possibilities for setting the text color indices (Ex.):
			iso.textcolors=([22,33,44,55,66,77])
			iso.textcolors=(16,19,33,44)
			iso.textcolors=None # Turns off the text color index

Command	Description	Options	Examples
	Outfill		
createoutfill	**Function:** createoutfill **Description of Function:** Create a new outfill graphics method given the the name and the existing outfill graphics method to copy the attributes from. If no existing outfill graphics method name is given, then the default outfill graphics method will be used as the graphics method to which the attributes will be copied from. If the name provided already exists, then a error will be returned. Graphics method names must be unique.	new outfill name [name of outfill to copy attributes from]	**Example of Use:** import vcs a=vcs.init() a.show('outfill') out=a.createoutfill('example1',) a.show('outfill') out=a.createoutfill('example2','quick') a.show('outfill')

getoutfill	**Function:** getoutfill
	
	Description of Function:
	VCS contains a list of graphics methods. This function will create a outfill class object from an existing VCS outfill graphics method. If no outfill name is given, then outfill 'default' will be used.
	Note, VCS does not allow the modification of 'default' attribute sets. However, a 'default' attribute set that has been copied under a different name can be modified. (See the createoutfill function.)
		[outfill name]	**Example of Use:**
			import vcs
			a=vcs.init()
			a.show('outfill')
			# Show all the existing outfill graphics methods
			out=a.getoutfill()
			# out instance of 'default' outfill graphics method
			out2=a.getoutfill('quick')
			# out2 instance of existing 'quick' outfill graphics method

Command	Description	Options	Examples

	Object: outfillobject
	
	Description of Function:
	The outfill graphics method fills a set of integer values in any data array.
	Its primary purpose is to display continents by filling their area as defined by a surface type array that indicates land, ocean, and sea-ice points. The example below shows how to apply the outfill graphics method and how to modify
	Fillarea and outfill attributes.
	Other Useful Functions:
	a=vcs.init()
	# Constructor
	a.show('outfill')
	# Show predefined outfill graphics methods
	a.show('line')
	# Show predefined VCS line objects
	a.setcolormap("AMIP")
	# Change the VCS color map
	a.outfill(s,o,'default')
	# Plot data 's' with outfill 'o' and 'default' template
	a.update()
	# Updates the VCS Canvas at user's request a.mode=1, or 0 . If 1, then automatic update, else if 0, then use update function to update the VCS Canvas.
		name
		projection
		xticlabels1
		xticlabels2
		xmtics1
		xmtics2
		yticlabels1
		yticlabels2
		ymtics1
		ymtics2
		datawc_x1
		datawc_y1
		datawc_x2
		datawc_y2
		xaxisconvert
		yaxisconvert
		fillareastyle
		fillareaindex
		fillareacolor
		outfill
			Example of Use:
			import vcs
			a=vcs.init()
			
			# To Create a new instance of outfill use:
			out=a.createoutfill('new','quick')
			# Copies content of 'quick' to 'new'
			out=a.createoutfill('new')
			# Copies content of 'default' to 'new'
			
			# To Modify an existing outfill use:
			out=a.getoutfill('AMIP_psl')
			out.list()
			# Will list all the outfill attribute values
			out.projection='linear'
			lon30={-180:'180W',-150:'150W',0:'Eq'}
			out.xticlabels1=lon30
			out.xticlabels2=lon30
			out.xticlabels(lon30, lon30)
			# Will set them both
			out.xmtics1=''
			out.xmtics2=''
			out.xmtics(lon30, lon30)
			# Will set them both
			out.yticlabels1=lat10
			out.yticlabels2=lat10
			out.yticlabels(lat10, lat10)

Command	Description	Options	Examples

	See Chapter 6 for additional information.		# Will set them both
			out.datawc_y1=-90.0
			out.datawc_y2=90.0
			out.datawc_x1=-180.0
			out.datawc_x2=180.0
			out.datawc(-90, 90, -180, 180) # Will set them all
			xaxisconvert='linear'
			yaxisconvert='linear'
			out.xyscale('linear', 'area_wt') # Will set them both
			# Specify the outfill fill values:
			out.outfill=([0,1,2,3,4]) # Same as below
			out.outfill=(0,1,2,3,4) # Will specify the outfill values
			# There are four possibilities for setting the color index (Ex):
			out.fillareacolor=22 # Same as below
			out.fillareacolor=(22) # Same as below
			out.fillareacolor=([22]) # Will set the outfill to a specific color index
			out.fillareacolor=None # Turns off the color index

Command	Description	Options	Examples
		Outline	
createoutline	**Function:** createoutline **Description of Function:** Create a new outline graphics method given the the name and the existing outline graphics method to copy the attributes from. If no existing outline graphics method name is given, then the default outline graphics method will be used as the graphics method to which the attributes will be copied from. If the name provided already exists, then a error will be returned. Graphics method names must be unique.	new outline name [name of outline to copy attributes from]	**Example of Use:** import vcs a=vcs.init() a.show('outline') out=a.createoutline('example1',) a.show('outline') out=a.createoutline('example2','quick') a.show('outline')

getoutline	**Function:** getoutline
	
	Description of Function:
	VCS contains a list of graphics methods. This function will create a outline class object from an existing VCS outline graphics method. If no outline name is given, then outline 'default' will be used.
	Note, VCS does not allow the modification of 'default' attribute sets. However, a 'default' attribute set that has been copied under a different name can be modified. (See the createoutline function.)
		[outline name]	**Example of Use:**
			import vcs
			a=vcs.init()
			a.show('outline')
			# Show all the existing outline graphics methods
			out=a.getoutline()
			# out instance of 'default' outline graphics method
			out2=a.getoutline('quick')
			# out2 instance of existing 'quick' outline graphics method

Command	Description	Options	Examples

	Object: outlineobject
	
	Description of Function:
	The Outline graphics method outlines a set of integer values in any data array.
	Its primary purpose is to display continental outlines as defined by a surface
	type array that indicates land, ocean, and sea-ice points. The example below
	shows how to change such an outline by modifying its attributes.
	
	Other Useful Functions:
	a=vcs.init()
	# Constructor
	a.show('outline')
	# Show predefined outline graphics methods
	a.show('line')
	# Show predefined VCS line objects
	a.setcolormap("AMIP")
	# Change the VCS color map
	a.outline(s,o,'default')
	# Plot data 's' with outline 'o' and 'default' template
	a.update()
		name
		projection
		xticlabels1
		xticlabels2
		xmtics1
		xmtics2
		yticlabels1
		yticlabels2
		ymtics1
		ymtics2
		datawc_x1
		datawc_y1
		datawc_x2
		datawc_y2
		xaxisconvert
		yaxisconvert
		line
		linecolor
		outline
			Example of Use:
			import vcs
			a=vcs.init()
			
			# To Create a new instance of outline use:
			out=a.createoutline('new','quick')
			# Copies content of 'quick' to 'new'
			out=a.createoutline('new')
			# Copies content of 'default' to 'new'
			
			# To Modify an existing outline use:
			out=a.getoutline('AMIP_psl')
			out.list()
			# Will list all the outline attribute values
			out.projection='linear'
			lon30={-180:'180W',-150:'150W',0:'Eq'}
			out.xticlabels1=lon30
			out.xticlabels2=lon30
			out.xticlabels(lon30, lon30)
			# Will set them both
			out.xmtics1=''
			out.xmtics2=''
			out.xmtics(lon30, lon30)
			# Will set them both
			out.yticlabels1=lat10
			out.yticlabels2=lat10
			out.yticlabels(lat10, lat10)
			# Will set them both
			out.ymtics1=''
			out.ymtics2=''
			out.ymtics(lat10, lat10)

Command	Description	Options	Examples

	# Updates the VCS Canvas at user's request a.mode=1, or 0
	# If 1, then automatic update, else if 0, then use u pdate function to
	update the VCS Canvas.
	See Chapter 6 for additional information.
			# Will set them both
			out.datawc_y1=-90.0
			out.datawc_y2=90.0
			out.datawc_x1=-180.0
			out.datawc_x2=180.0
			out.datawc(-90, 90, -180, 180) # Will set them all
			xaxisconvert='linear'
			yaxisconvert='linear'
			out.xyscale('linear', 'area_wt') # Will set them both
			# Specify the outline fill values:
			out.outline=([0,1,2,3,4]) # Same as below
			out.outline=(0,1,2,3,4) # Will specify the outline values
			# Specify the outline line type:
			out.line=0 # same as out.line = 'solid'
			out.line=1 # same as out.line = 'dash'
			out.line=2 # same as out.line = 'dot'
			out.line=3
			# same as out.line = 'dash-dot'
			out.line=4 # same as out.line = 'long-dash'
			
			# There are four possibilities for setting the line color index (Ex):
			out.linecolor=22 # Same as below
			# Same as below
			out.linecolor=([22]) # Will set the outline to a specific color index
			out.linecolor=None # Turns off the color index

Command	Description	Options	Examples
	Scatter		
createscatter	**Function:** createscatter **Description of Function:** Create a new scatter graphics method given the the name and the existing mscatter graphics method to copy the attributes from. If no existing scatter graphics method name is given, then the default scatter graphics method will be used as the graphics method to which the attributes will be copied from. If the name provided already exists, then a error will be returned. Graphics method names must be unique	new scatter name [name of scatter to copy attributes from]	**Example of Use:** import vcs a=vcs.init() a.show('scatter') sct=a.createscatter('example1',) a.show('scatter') sct=a.createscatter('example2','quick') a.show('scatter')

getscatter	**Function:** getscatter
	
	Description of Function:
	VCS contains a list of graphics methods. This function will create a scatter class object from an existing VCS scatter graphics method. If no scatter name is given, then scatter 'default' will be used.
	Note, VCS does not allow the modification of 'default' attribute sets. However, a 'default' attribute set that has been copied under a different name can be modified. (See the createscatter function.)
		[scatter name]	**Example of Use:**
			import vcs
			a=vcs.init()
			a.show('scatter')
			# Show all the existing scatter graphics methods
			sct=a.getscatter()
			# sct instance of 'default' scatter graphics method
			sct2=a.getscatter('quick')
			# sct2 instance of existing 'quick' scatter graphics method

Command	Description	Options	Examples

	Object: scatterobject
	
	Description of Function:
	The Scatter graphics method displays a scatter plot of two 4-dimensional data arrays, e.g. A(x,y,z,t) and B(x,y,z,t). The example below shows how to change the marker attributes of a scatter plot.
	This class is used to define an scatter table entry used in VCS, or it can be used to change some or all of the scatter attributes in an existing scatter table entry.
	Other Useful Functions:
	a=vcs.init()
	# Constructor
	a.show('scatter')
	# Show predefined scatter graphics methods
	a.show('marker')
	# Show predefined marker objects
	a.setcolormap("AMIP")
	# Change the VCS color map
	a.scatter(s1, s2, s,'default')
	# Plot data 's1' and 's2' with scatter 's' and 'default' template
	a.update()
	# Updates the VCS Canvas at user's request a.mode=1, or 0. If 1, then automatic update, else if 0, then use update function to update the VCS Canvas.
		name
		projection
		xticlabels1
		xticlabels2
		xmtics1
		xmtics2
		yticlabels1
		yticlabels2
		ymtics1
		ymtics2
		datawc_x1
		datawc_y1
		datawc_x2
		datawc_y2
		xaxisconvert
		yaxisconvert
		marker
		markercolor
		markersize
			Example of Use:
			import vcs
			a=vcs.init()
			
			# To Create a new instance of scatter use:
			sr=a.createscatter('new','quick')
			# copies content of 'quick' to 'new'
			sr=a.createscatter('new')
			# copies content of 'default' to 'new'
			
			# To Modify an existing scatter use:
			sr=a.getscatter('AMIP_psl')
			sr.list()
			# Will list all the scatter attribute values
			sr.projection='linear'
			# Can only be 'linear'
			lon30={-180:'180W',-150:'150W',0:'Eq'}
			sr.xticlabels1=lon30
			sr.xticlabels2=lon30
			sr.xticlabels(lon30, lon30)
			# Will set them both
			sr.xmtics1=''
			sr.xmtics2=''
			sr.xmtics(lon30, lon30)
			# Will set them both
			sr.yticlabels1=lat10
			sr.yticlabels2=lat10
			sr.yticlabels(lat10, lat10)
			# Will set them both
			sr.ymtics1=''
			sr.ymtics2=''
			sr.ymtics(lat10, lat10)

Command	Description	Options	Examples

	See Chapter 6 for additional information.		# Will set them both
			sr.datawc_y1=-90.0
			sr.datawc_y2=90.0
			sr.datawc_x1=-180.0
			sr.datawc_x2=180.0
			sr.datawc(-90, 90, -180, 180) # Will set them all
			sr.xaxisconvert='linear'
			sr.yaxisconvert='linear'
			sr.xyscale('linear', 'area_wt') # Will set them both
			# Specify the marker type:
			sr.marker=1 # Same as sr.marker='dot'
			sr.marker=2 # Same as sr.marker='plus'
			sr.marker=3 # Same as sr.marker='star'
			sr.marker=4 # Same as sr.marker='circle'
			sr.marker=5 # Same as sr.marker='cross'
			sr.marker=6 # Same as sr.marker='diamond'
			sr.marker=7
			# Same as sr.marker='triangle_up'
			sr.marker=8 # Same as sr.marker='triangle_down'
			sr.marker=9 # Same as sr.marker='triangle_left'
			sr.marker=10 # Same as sr.marker='triangle_right'
			sr.marker=11 # Same as sr.marker='square'
			sr.marker=12 # Same as sr.marker='diamond_fill'
			sr.marker=13 # Same as sr.marker='triangle_up_fill'

Command	Description	Options	Examples

			sr.marker=14 # Same as sr.marker='triangle_down_fill'
			sr.marker=15 # Same as sr.marker='triangle_left_fill'
			sr.marker=16
			# Same as below
			sr.markercolors=(22) # Same as below
			sr.markercolors=([22]) # Will set the markers to a specific # color index
			sr.markercolors=None # Color index defaults to Black
			
			To set the Marker sizie:
			sr.markersize=5
			sr.markersize=55
			sr.markersize=100
			sr.markersize=300
			sr.markersize=None

Command	Description	Options	Examples
		Vector	
createvector	**Function:** createvector **Description of Function:** Create a new vector graphics method given the the name and the existing vector graphics method to copy the attributes from. If no existing vector graphics method name is given, then the default vector graphics method will be used as the graphics method to which the attributes will be copied from. If the name provided already exists, then a error will be returned. Graphics method names must be unique.	new vector name [name of vector to copy attributes from]	**Example of Use:** import vcs a=vcs.init() a.show('vector') vec=a.createvector('example1',) a.show('vector') vec=a.createvector('example2','quick') a.show('vector')

getvector	**Function:** getvector
	
	Description of Function:
	VCS contains a list of graphics methods. This function will create a vector class object from an existing VCS vector graphics method. If no vector name is given, then vector 'default' will be used.
	Note, VCS does not allow the modification of 'default' attribute sets. However, a 'default' attribute set that has been copied under a different name can be modified. (See the createvector function.)
		[vector name]	**Example of Use:**
			import vcs
			a=vcs.init()
			a.show('vector')
			# Show all the existing vector graphics methods
			vec=a.getvector()
			# vec instance of 'default' vector graphics method
			vec2=a.getvector('quick')
			# vec2 instance of existing 'quick' vector graphics method

Command	Description	Options	Examples

	Object: vectorobject
	
	Description of Function:
	The vector graphics method displays a vector plot of a 2D vector field. Vectors
	are located at the coordinate locations and point in the direction of the data vector field. Vector magnitudes are the product of data vector field lengths and a scaling factor. The example below shows how to modify the vector's line, scale, alignment, type, and reference.
	This class is used to define an vector table entry used in VCS, or it can be used to change some or all of the vector attributes in an existing vector table entry.
	Other Useful Functions:
	a=vcs.init()
	# Constructor
	a.show('vector')
	# Show predefined vector graphics methods
	a.show('line')
	# Show predefined VCS line objects
	a.setcolormap("AMIP")
	# Change the VCS color Map
	a.vector(s1, s2, v,'default')
	# Plot data 's1', and 's2' with vector 'v' and 'default' template
		name
		projection
		xticlabels1
		xticlabels2
		xmtics1
		xmtics2
		yticlabels1
		yticlabels2
		ymtics1
		ymtics2
		datawc_x1
		datawc_y1
		datawc_x2
		datawc_y2
		xaxisconvert
		yaxisconvert
		line
		linecolor
		scale
		alignment
		type
		reference
			Example of Use:
			import vcs
			a=vcs.init()
			
			# To Create a new instance of vector use:
			vc=a.createvector('new','quick')
			# Copies content of 'quick' to 'new'
			vc=a.createvector('new')
			# Copies content of 'default' to 'new'
			
			# To Modify an existing vector use:
			vc=a.getvector('AMIP_psl')
			vc.list()
			# Will list all the vector attribute values
			vc.projection='linear'
			# Can only be 'linear'
			lon30={-180:'180W',-150:'150W',0:'Eq'}
			vc.xticlabels1=lon30
			vc.xticlabels2=lon30
			vc.xticlabels(lon30, lon30)
			# Will set them both
			vc.xmtics1=''
			vc.xmtics2=''
			vc.xmtics(lon30, lon30)
			# Will set them both
			vc.yticlabels1=lat10
			vc.yticlabels2=lat10
			vc.yticlabels(lat10, lat10)
			# Will set them both
			vc.ymtics1=''
			vc.ymtics2=''
			vc.ymtics(lat10, lat10)

Command	Description	Options	Examples

	a.update() # Updates the VCS Canvas at user's request a.mode=1, or 0. If 1, then automatic update, else if 0, then use update function to update the VCS Canvas.
	See Chapter 6 for additional information.
			# Will set them both
			vc.datawc_y1=-90.0
			vc.datawc_y2=90.0
			vc.datawc_x1=-180.0
			vc.datawc_x2=180.0
			vc.datawc(-90, 90, -180, 180) # Will set them all
			xaxisconvert='linear'
			yaxisconvert='linear'
			vc.xyscale('linear', 'area_wt') # Will set them both
			# Specify the line style:
			vc.line=0 # Same as vc.line='solid'
			vc.line=1 # Same as vc.line='davc.line=2 # Same as vc.line='dot'
			vc.line=3 # Same as vc.line='dash-dot'
			vc.line=4 # Same as vc.line='long-dot'
			# Specify the line color of the vectors:
			vc.linecolor=16
			# Color range: 16 to 230, default line color is black
			# Specify the vector scale factor:
			vc.scale=2.0 # Can be an integer or float
			# Specify the vector alignment:
			vc.alignment=0
			# Same as vc.alignment='head'
			vc.alignment=1 # Same as vc.align-ment='center'

Command	Description	Options	Examples

			vc.alignment=2 # Same as vc.alignment='tail'
			# Specify the vector type:
			vc.type=0 # Same as vc.type='arrow head'
			vc.type=1 # Same as vc.type='wind barbs'
			vc.type=2 # Same as vc.type='solid arrow head'
			
			Specify the vector reference:
			vc.reference=4 # Can be an integer or float

XvsY			
createxvsy	**Function:** createxvsy **Description of Function:** Create a new XvsY graphics method given the the name and the existing XvsY graphics method to copy the attributes from. If no existing XvsY graphics method name is given, then the default XvsY graphics method will be used as the graphics method to which the attributes will be copied from. If the name provided already exists, then a error will be returned. Graphics method names must be unique.	new xvsy name [name of xvsy to copy attributes from]	**Example of Use:** import vcs a=vcs.init() a.show('xvsy') xy=a.createxvsy('example1',) a.show('xvsy') xy=a.createxvsy('example2','quick') a.show('xvsy')

Command	Description	Options	Examples

getxvsy	**Function:** getxvsy
	
	Description of Function:
	VCS contains a list of graphics methods. This function will create a XvsY class object from an existing VCS XvsY graphics method. If no XvsY name is given, then XvsY 'default' will be used.
	Note, VCS does not allow the modification of 'default' attribute sets. However, a 'default' attribute set that has been copied under a different name can be modified. (See the createxvsy function.)
		[xvsy name]	**Example of Use:**
			import vcs
			a=vcs.init()
			a.show('xvsy')
			# Show all the existing XvsY xy=a.getxvsy()
			# graphics methods xy instance of 'default' XvsY graphics method
			xy2=a.getxvsy('quick')
			# xy2 instance of existing 'quick' XvsY graphics method

Command	Description	Options	Examples

	Object: xvsyobject
	
	Description of Function:
	The XvsY graphics method displays a line plot from two 1D data arrays, that is X(t) and Y(t), where t represents the 1D coordinate values. The example below shows how to change line and marker attributes for the XvsY graphics method.
	This class is used to define an XvsY table entry used in VCS, or it can be used to change some or all of the XvsY attributes in an existing XvsY table entry.
	Other Useful Functions:
	a=vcs.init()
	# Constructor
	a.show('xvsy')
	# Show predefined XvsY graphics methonds
	a.show('line')
	# Show predefined VCS line objects
	a.show('marker')
	# Show predefined VCS marker objects
	a.setcolormap("AMIP")
	# Change the VCS color map
	a.xvsy(s1, s2, ,x,'default')
	# Plot data 's1' and 's2' with XvsY 'x' and 'default' template
		name
		projection
		xticlabels1
		xticlabels2
		xmtics1
		xmtics2
		yticlabels1
		yticlabels2
		ymtics1
		ymtics2
		datawc_x1
		datawc_y1
		datawc_x2
		datawc_y2
		xaxisconvert
		yaxisconvert
		line
		linecolor
		marker
		markercolor
		markersize
			Example of Use:
			import vcs
			a=vcs.init()
			
			# To Create a new instance of XvsY use:
			xy=a.createxvsy('new','quick')
			# Copies content of 'quick' to 'new'
			xy=a.createxvsy('new')
			# Copies content of 'default' to 'new'
			
			# To Modify an existing XvsY use:
			xy=a.getxvsy('AMIP_psl')
			xy.list()
			# Will list all the XvsY attribute values
			xy.projection='linear'
			# Can only be 'linear'
			lon30={-180:'180W',-150:'150W',0:'Eq'}
			xy.xticlabels1=lon30
			xy.xticlabels2=lon30
			xy.xticlabels(lon30, lon30)
			# Will set them both
			xy.xmtics1=''
			xy.xmtics2=''
			xy.xmtics(lon30, lon30)
			# Will set them both
			xy.yticlabels1=lat10
			xy.yticlabels2=lat10
			xy.yticlabels(lat10, lat10)

Command	Description	Options	Examples

	a.update() # Updates the VCS Canvas at user's request a.mode=1, or 0.
	If 1, then automatic update, else if 0, then use update function to Update the VCS Canvas.
	See Chapter 6 for additional information.
			# Will set them both
			xy.datawc_y1=-90.0
			xy.datawc_y2=90.0
			xy.datawc_x1=-180.0
			xy.datawc_x2=180.0
			xy.datawc(-90, 90, -180, 180) # Will set them all
			xaxisconvert='linear'
			yaxisconvert='linear'
			xy.xyscale('linear', 'area_wt') # Will set them both
			# Specify the XvsY line type:
			xy.line=0 # same as xy.line = 'solid'
			xy.line=1 # same as xy.line = 'dash'
			xy.line=2 # same as xy.line = 'dot'
			xy.line=3 # same as xy.line = 'dash-dot'
			xy.line=4 # same as xy.line = 'long-dash
			# Specify the Xvsy line color:
			xy.line# color range: 16 to 230, default color is black
			# Specify the XvsY marker type:
			xy.marker=1 # Same as xy.marker='dot'
			xy.marker=2 # Same as xy.marker='plus'
			xy.marker=3
			color=16 # color index

Command	Description	Options	Examples

			# Same as xy.marker='star'
			xy.marker=4 # Same as xy.marker='circle'
			xy.marker=5 # Same as xy.marker='cross'
			xy.marker=6 # Same as xy.marker='diamond'
			xy.marker=7 # Same as xy.marker='triangle_up'
			xy.marker=8 # Same as xy.marker='triangle_down'
			xy.marker=9 # Same as xy.marker='triangle_left'
			xy.marker=10 # Same as xy.marker='triangle_right'
			xy.marker=11 # Same as xy.marker='square'
			xy.marker=12
			# Same as xy.marker='square'
			xy.marker=12 # Same as xy.marker='diamond_fill'
			xy.marker=13 # Same as xy.marker='triangle_up_fill'
			xy.marker=14 # Same as xy.marker='triangle_down_fill'
			xy.marker=15 # Same as xy.marker='triangle_left_fill'
			xy.marker=16 # Same as xy.marker='triangle_right_fill'
			xy.marker=17 # Same as xy.marker='square_fill'
			xy.marker=None # Draw no markers
			
			There are four possibilities for setting the marker color index (Ex):

Command	Description	Options	Examples

			xy.markercolors=22 # Same as below
			xy.markercolors=(22) # Same as below
			xy.markercolors=([22]) # Will set the markers to a specific
			xy.markercolors=None # Color index defaults to Black
			
			To set the XvsY Marker sizie:
			xy.markersize=5
			xy.markersize=55
			xy.markersize=100
			xy.markersize=300
			xy.markersize=None

		Xyvsy	
createx-yvsy	**Function:** createxyvsy **Description of Function:** Create a new Xyvsy graphics method given the the name and the existing Xyvsy graphics method to copy the attributes from. If no existing Xyvsy graphics method name is given, then the default Xyvsy graphics method will be used as the graphics method to which the attributes will be copied from. If the name provided already exists, then a error will be returned. Graphics method names must be unique.	new xyvsy name [name of xyvsy to copy attributes from]	**Example of Use:** import vcs a=vcs.init() a.show('xyvsy') xyy=a.createxyvsy('example1',) a.show('xyvsy') xyy=a.createxyvsy('example2','quick') a.show('xyvsy')

Command	Description	Options	Examples

getxyvsy	**Function:** getxyvsy
	
	Description of Function:
	VCS contains a list of graphics methods. This function will create a Xyvsy class object from an existing VCS Xyvsy graphics method. If no Xyvsy name is given, then Xyvsy 'default' will be used.
	Note, VCS does not allow the modification of 'default' attribute sets. However, a 'default' attribute set that has been copied under a different name can be modified. (See the createxyvsy function.)
		[xyvsy name]	**Example of Use:**
			import vcs
			a=vcs.init()
			a.show('xyvsy')
			# Show all the existing Xyvsy graphics methods
			xyy=a.getxyvsy()
			# xyy instance of 'default' Xyvsy graphics method
			xyy2=a.getxyvsy('quick')
			# xyy2 instance of existing 'quick' Xyvsy graphics method

Command	Description	Options	Examples

	Object: xyvsyobject
	
	Description of Function:
	The Xyvsy graphics method displays a line plot from a 1D data array (i.e. a plot of X(y), where y represents the 1D coordinate values). The example below ributes for the Xyvsy graphics method.
	This class is used to define an Xyvsy table entry used in VCS, or it can be used to change some or all of the Xyvsy attributes in an existing Xyvsy table entry.
	Other Useful Functions:
	a=vcs.init()
	# Constructor
	a.show('xyvsy')
	# Show predefined Xyvsy graphics methonds
	a.show('line')
	# Show predefined VCS line objects
	a.show('marker')
	# Show predefined VCS marker objects
	a.setcolormap("AMIP")
	# Change the VCS color map
	a.xyvsy(s, x, 'default')
	# Plot data 's' with Xyvsy 'x' and 'default' template
	a.update()
	# Updates the VCS Canvas at user's request a.mode=1, or 0 . If 1, then automatic update, else if 0, then use update function.
		name
		projection
		xticlabels1
		xticlabels2
		xmtics1
		xmtics2
		yticlabels1
		yticlabels2
		ymtics1
		ymtics2
		datawc_x1
		datawc_y1
		datawc_x2
		datawc_y2
		xaxisconvert
		line
		linecolor
		marker
		markercolor
		markersize
			Example of Use:
			import vcs
			a=vcs.init()
			
			# To Create a new instance of Xyvsy use:
			xyy=a.createxyvsy('new','quick')
			# Copies content of 'quick' to 'new'
			xyy=a.createxyvsy('new')
			# Copies content of 'default' to 'new'
			
			# To Modify an existing Xyvsy use:
			xyy=a.getxyvsy('AMIP_psl')
			xyy.list()
			# Will list all the Xyvsy attribute values
			xyy.projection='linear'
			# Can only be 'linear'
			lon30={-180:'180W',-150:'150W',0:'Eq'}
			xyy.xticlabels1=lon30
			xyy.xticlabels2=lon30
			xyy.xticlabels(lon30, lon30)
			# Will set them both
			xyy.xmtics1=''
			xyy.xmtics2=''
			xyy.xmtics(lon30, lon30)
			# Will set them both
			xyy.yticlabels1=lat10
			xyy.yticlabels2=lat10
			xyy.yticlabels(lat10, lat10)
			# Will set them both
			xyy.ymtics1=''
			xyy.ymtics2=''
			xyy.ymtics(lat10, lat10)

Command	Description	Options	Examples

	See Chapter 6 for additional information.		# Will set them both
			xyy.datawc_y1=-90.0
			xyy.datawc_y2=90.0
			xyy.datawc_x1=-180.0
			xyy.datawc_x2=180.0
			xyy.datawc(-90, 90, -180, 180) # Will set them all
			xyy.xaxisconvert='linear'
			# Specify the Xyvsy line type:
			xyy.line=0 # same as xyy.line = 'solid'
			xyy.line=1 # same as xyy.line = 'dash'
			xyy.line=2 # same as xyy.line = 'dot'
			xyy.line=3
			same as xyy.line = 'dash-dot'
			xyy.line=4 # same as xyy.line = 'long-dash
			# Specify the Xyvsy line color:
			xyy.linecolor=16 # color range: 16 to 230, default color is black
			# Specify the Xyvsy marker type:
			xyy.marker=1 # Same as xyy.marker='dot'
			xyy.marker=2 # Same as xyy.marker='plus'
			xyy.marker=3 # Same as xyy.marker='star'
			xyy.marker=4
			# Same as xyy.marker='circle'
			xyy.marker=5 # Same as xyy.marker='cross'
			xyy.marker=6 # Same as xyy.marker='diamond'
			xyy.marker=7 # Same as xyy.marker='triangle_up'

Command	Description	Options	Examples

			xyy.marker=8 # Same as xyy.marker='triangle_down'
			xyy.marker=9 # Same as xyy.marker='triangle_left'
			xyy.marker=10 # Same as xyy.marker='triangle_right'
			xyy.marker=11 # Same as xyy.marker='square'
			xyy.marker=12 # Same as xyy.marker='diamond_fill'
			xyy.marker=13
			# Same as xyy.marker='triangle_up_fill'
			xyy.marker=14 # Same as xyy.marker='triangle_down_fill'
			xyy.marker=15 # Same as xyy.marker='triangle_left_fill'
			xyy.marker=16 # Same as xyy.marker='triangle_right_fill'
			xyy.marker=17 # Same as xyy.marker='square_fill'
			xyy.marker=None # Draw no markers
			
			There are four possibilities for setting the marker color index (Ex):
			xyy.markercolors=22 # Same as below
			xyy.markercolors=(22) # Same as below
			xyy.markercolors=([22]) # Will set the markers to a specific
			# color index
			xyy.markercolors=None # Color index defaults to Black
			
			To set the Xyvsy Marker sizie:
			xyy.markersize=5

Command	Description	Options	Examples
			xyy.markersize=55 xyy.markersize=100 xyy.markersize=300 xyy.markersize=None
Yxvsx			
createyx-vsx	**Function:** createyxvsx **Description of Function:** Create a new Yxvsx graphics method given the the name and the existing Yxvsx graphics method to copy the attributes from. If no existing Yxvsx graphics method name is given, then the default Yxvsx graphics method will be used as the graphics method to which the attributes will be copied from. If the name provided already exists, then a error will be returned. Graphics method names must be unique	new yxvsx name [name of yxvsx to copy attributes from]	**Example of Use:** import vcs a=vcs.init() a.show('yxvsx') yxx=a.createyxvsx('example1',) a.show('yxvsx') yxx=a.createyxvsx('example2','quick') a.show('yxvsx')

Command	Description	Options	Examples

getyxvsx	**Function:** getyxvsx
	
	Description of Function:
	VCS contains a list of graphics methods. This function will create a Yxvsx class object from an existing VCS Yxvsx graphics method. If no Yxvsx name is given, then Yxvsx 'default' will be used.
	Note, VCS does not allow the modification of 'default' attribute sets. However, a 'default' attribute set that has been copied under a different name can be modified. (See the createyxvsx function.)
		[yxvsx name]	**Example of Use:**
			import vcs
			a=vcs.init()
			a.show('yxvsx')
			# Show all the existing Yxvsx graphics methods
			yxx=a.getyxvsx()
			# yxx instance of 'default' Yxvsx graphics method
			yxx2=a.getyxvsx('quick')
			# yxx2 instance of existing 'quick' Yxvsx graphics method

Command	Description	Options	Examples

	Object: yxvsxobject
	
	Description of Function:
	The Yxvsx graphics method displays a line plot from a 1D data array (i.e. aplot of Y(x), where y represents the 1D coordinate values). The example belowshows how to change line and marker attributes for the Yxvsx graphics method.
	This class is used to define an Yxvsx table entry used in VCS, or it can beused to change some or all of the Yxvsx attributes in an existing Yxvsx table entry.
	Other Useful Functions:
	a=vcs.init()
	# Constructor
	a.show('yxvsx')
	# Show predefined Yxvsx graphics methonds
	a.show('line')
	# Show predefined VCS line objects
	a.show('marker')
	# Show predefined VCS marker objects
	a.setcolormap("AMIP")
	# Change the VCS color map
	a.yxvsx(s, x, 'default')
	# Plot data 's' with Yxvsx 'x' and 'default' template
	a.update()
	# Updates the VCS Canvas at user's request
	a.mode=1, or 0 If 1, then automatic update, else if 0, then use update function.
		name
		projection
		xticlabels1
		xticlabels2
		xmtics1
		xmtics2
		yticlabels1
		yticlabels2
		ymtics1
		ymtics2
		datawc_x1
		datawc_y1
		datawc_x2
		datawc_y2
		yaxisconvert
		line
		linecolor
		marker
		markercolor
		markersize
			Example of Use:
			import vcs
			a=vcs.init()
			
			# To Create a new instance of Yxvsx use:
			yxx=a.createxyvsy('new','quick')
			# Copies content of 'quick' to 'new'
			yxx=a.createxyvsy('new')
			# Copies content of 'default' to 'new'
			
			# To Modify an existing Yxvsx use:
			yxx=a.getxyvsy('AMIP_psl')
			yxx.list()
			# Will list all the Yxvsx attribute values
			yxx.projection='linear'
			# Can only be 'linear'
			lon30={-180:'180W',-150:'150W',0:'Eq'}
			yxx.xticlabels1=lon30
			yxx.xticlabels2=lon30
			yxx.xticlabels(lon30, lon30)
			# Will set them both
			yxx.xmtics1=''
			yxx.xmtics2=''
			yxx.xmtics(lon30, lon30)
			# Will set them both
			yxx.yticlabels1=lat10
			yxx.yticlabels2=lat10
			yxx.yticlabels(lat10, lat10)

Command	Description	Options	Examples

	See Chapter 6 for additional information.		# Will set them both
			yxx.datawc_y1=-90.0
			yxx.datawc_y2=90.0
			yxx.datawc_x1=-180.0
			yxx.datawc_x2=180.0
			yxx.datawc(-90, 90, -180, 180) # Will set them all
			yxx.yaxisconvert='linear'
			# Specify the Yxvsx line type:
			yxx.line=0 # same as yxx.line = 'solid'
			yxx.line=1 # same as yxx.line = 'dash'
			yxx.line=2 # same as yxx.line = 'dot'
			yxx.line=3
			# same as yxx.line = 'dash-dot'
			yxx.line=4 # same as yxx.line = 'long-dash
			# Specify the Yxvsx line color:
			yxx.linecolor=16 # color range: 16 to 230, default color is black
			yxx.linecolor=16 # color range: 16 to 230, default color is black
			# Specify the Yxvsx marker type:
			yxx.marker=1 # Same as yxx.marker='dot'
			yxx.marker=2 # Same as yxx.marker='plus'
			yxx.marker=3 # Same as yxx.marker='star'
			yxx.marker=4 # Same as yxx.marker='circle'
			yxx.marker=5 # Same as yxx.marker='cross'
			yxx.marker=6 # Same as yxx.marker='diamond'

Command	Description	Options	Examples

			yxx.marker=7 # Same as yxx.marker='triangle_up'
			yxx.marker=8# Same as yxx.marker='triangle_down'
			yxx.marker=9 # Same as yxx.marker='triangle_left'
			yxx.marker=10 # Same as yxx.marker='triangle_right'
			yxx.marker=11 # Same as yxx.marker='square'
			yxx.marker=12 # Same as yxx.marker='diamond_fill'
			yxx.marker=13 # Same as yxx.marker='triangle_up_fill'
			yxx.marker=14 # Same as yxx.marker='triangle_down_fill'
			yxx.marker=15 # Same as yxx.marker='triangle_left_fill'
			yxx.marker=16 # Same as yxx.marker='triangle_right_fill'
			yxx.marker=17 # Same as yxx.marker='square_fill'
			yxx.marker=None # Draw no markers
			
			There are four possibilities for setting the marker color index (Ex):
			yxx.markercolors=22 # Same as below
			yxx.markercolors=(22) # Same as below
			yxx.markercolors=([22])
			# Will set the markers to a specific
			color index
			yxx.markercolors=None # Color

Command	Description	Options	Examples
			index defaults to Black To set the Yxvsx Marker size: yxx.markersize=5 yxx.markersize=55 yxx.markersize=100 yxx.markersize=300 yxx.markersize=None
Picture Template			

createtem-plate	**Function:** createtemplate
	
	Description of Function:
	Create a new template given the the name and the existing template to copy the attributes from. If no existing template name is given, then the default template will be used as the template to which the attributes will be copied from.
	If the name provided already exists, then a error will be returned. Template names must be unique.
		new template name
		
		[name of tem-plate to copy attributes from]
			Example of Use:
			import vcs
			a=vcs.init()
			a.show('template')
			# Show all the existing templates
			con=a.createtemplate('example1')
			# create 'example1' template from 'default' template
			a.show('template')
			# Show all the existing templates
			con=a.createtemplate('example2','quick')
			# create 'example2' from 'quick' template
			a.listelements('template')
			# Show all the templates as a Python list

Command	Description	Options	Examples

gettemplate	**Function:** gettemplate
	
	Description of Function:
	VCS contains a list of pre-defined templates. This function will create a template class object from an existing VCS template. If no template name is given, then template 'default' will be used.
	Note, VCS does not allow the modification of 'default' attribute sets. However, a 'default' attribute set that has been copied under a different name can be modified. (See the createtemplate function.)
		[template name]	**Example of Use:**
			import vcs
			a=vcs.init()
			a.show('template')
			# Show all the existing templates
			templt=a.gettemplate()
			# templt instance of 'default' template
			templt2=a.gettemplate('quick')
			# templt2 contains 'quick' template

Command	Description	Options	Examples

	Object: templateobject
	
	Description of Function:
	The template primary method (P) determines the location of each picture segment, the space to be allocated to it, and related properties relevant to its display.
	Other Useful Functions:
	a.show('template')
	# Show predefined templates
	a.show('texttable')
	# Show predefined text table methods
	a.show('textorientation')
	# Show predefined text orientation methods
	a.show('line')
	# Show predefined line methods
	a.listelements('template')
	# Show templates as a Python list
	a.update()
	# Updates the VCS Canvas at user's request a.mode=1, or 0. If 1, then automatic update, else if 0, then use the update function toupdate the VCS Canvas.
		See Chapter 6 for the long list of options.	**Example of Use:**
			import vcs
			a=vcs.init()
			
			# To Create a new instance of boxfill use:
			box=a.createboxfill('new','quick')
			# Copies content of 'quick' to 'new'
			box=a.createboxfill('new')
			# Copies content of 'default' to 'new'
			
			# To Modify an existing boxfill use:
			box=a.getboxfill('AMIP_psl')
			
			# To Create a new instance of template use:
			tpl=a.createtemplate('new','AMIP')
			# Copies content of 'AMIP' to 'new'
			tpl=a.createtemplate('new')
			# Copies content of 'default' to 'new'
			
			# To Modify an existing template use:
			tpl=a.gettemplate('AMIP')

Command	Description	Options	Examples
Secondary Objects			
Colormap Commands			
setcolor-map	**Function:** setcolormap **Description of Function:** It is necessary to change the colormap. This routine will change the VCS color map. If the the visul display is 16-bit, 24-bit, or 32-bit TrueColor, then a redrawing of the VCS Canvas is made evertime the colormap is changed.		**Example of Use:** import vcsa=vcs.init() a.plot(array,'default','isofill','quick') a.setcolormap("AMIP")
setcolorcell	**Function:** setcolorcell **Description of Function:** Set a individual color cell in the active colormap. If default is the active colormap, then return an error string. If the the visul display is 16-bit, 24-bit, or 32-bit TrueColor, then a redrawing of the VCS Canvas is made evertime the color cell is changed. Note, the user can only change color cells 0 through 239 and R,G,Bvalue must range from 0 to 100. Where 0 represents no color intensity and 100 is the greatest color intensity.	colormap index: 0 to 239 **R** - Red intensity value: 0 to 100 **G** - Green intensity value: 0 to 100 **B** - Blue intensity value: 0 to 100	**Example of Use:** import vcs a=vcs.init() a.plot(array,'default','isofill','quick') a.setcolormap("AMIP") a.setcolorcell(11,0,0,0) a.setcolorcell(21,100,0,0) a.setcolorcell(31,0,100,0) a.setcolorcell(41,0,0,100) a.setcolorcell(51,100,100,100) a.setcolorcell(61,70,70,70)

Command	Description	Options	Examples
color-mapgui	**Function:** colormapgui **Description of Function:** Run the VCS colormap interface. The colormapgui command is used to bring up the VCS colormap interface. The interface is used to select, create, change, or remove colormaps. Note: The colormapgui GUI will only work for 8-bit 'Pseudo Color'.		**Example of Use:** import vcs a=vcs.init() a.colormapgui()
	Fill Area		
createfil-larea	**Function:** createfillarea **Description of Function:** Create a new fillarea secondary method given the the name and the existing fillarea secondary method to copy the attributes from. If no existing fillarea secondary method name is given, then the default fillarea secondary method will be used as the secondary method to which the attributes will be copied from. If the name provided already exists, then a error will be returned. Secondary method names must be unique.	new fillarea name [name of fillarea to copy attributes from]	**Example of Use:** import vcs a=vcs.init() a.show('fillarea') fa=a.createfillarea('example1',) a.show('fillarea') fa=a.createfillarea('example2','black') a.show('fillarea')

Command	Description	Options	Examples

getfillarea	**Function:** getfillarea
	
	Description of Function:
	VCS contains a list of secondary methods. This function will create a fillarea class object from an existing VCS fillarea secondary method. If no fillarea name is given, then fillarea 'default' will be used.
	Note, VCS does not allow the modification of 'default' attribute sets. However, a 'default' attribute set that has been copied under a different name can be modified. (See the createfillarea function.)
		[fillarea name]	**Example of Use:**
			import vcs
			a=vcs.init()
			a.show('fillarea')
			# Show all the existing fillarea secondary methods
			fa=a.getfillarea()
			# fa instance of 'default' fillarea secondary method
			fa2=a.getfillarea('quick')
			# fa2 instance of existing 'quick' fillarea secondary method

Command	Description	Options	Examples

	Object: fillareaobject
	
	Description of Function:
	The Fillarea class object allows the user to edit fillarea attributes, including
	fillarea interior style, style index, and color index.
	This class is used to define an fillarea table entry used in VCS, or itcan be used to change some or all of the fillarea attributes in an existing fillarea table entry.
	Other Useful Functions:
	a=vcs.init()
	# Constructor
	a.show('fillarea')
	# Show predefined fillarea objects
	a.update()
	# Updates the VCS Canvas at user's requesta.mode=1,or0
	# If 1, then automatic update, else if 0, then use update function to update the VCS Canvas.
		name
		style
		index
		color
			Example of Use:
			import vcs
			a=vcs.init()
			
			# To Create a new instance of fillarea use:
			fa=a.createfillarea('new','def37')
			# Copies content of 'def37' to 'new'
			fa=a.createfillarea('new')
			# Copies content of 'default' to 'new'
			
			# To Modify an existing fillarea use:
			fa=a.getfillarea('red')
			
			# fa.list()
			# Will list all the fillarea attribute values
			
			# There are three possibilities for setting the isofill style (Ex):
			fa.style = 'solid'
			fa.style = 'hatch'
			fa.style = 'pattern'
			fa.index=1
			# Range from 1 to 20
			fa.color=100
			# Range from 1 to 256
			
			# Specify the fillarea index:
			fa.index=1
			fa.index=2
			fa.index=3
			fa.index=4
			fa.index=5
			fa.index=6
			fa.index=7

Command	Description	Options	Examples
			fa.index=8 fa.index=9 fa.index=10 fa.index=11 fa.index=12 fa.index=13 fa.index=14 fa.index=15 fa.index=16 fa.index=17 fa.index=18 fa.index=19 fa.index=20
Line			
createline	**Function:** createline **Description of Function:** Create a new line secondary method given the the name and the existing line secondary method to copy the attributes from. If no existing line secondary method name is given, then the default line secondary method will be used as the secondary method to which the attributes will be copied from. If the name provided already exists, then a error will be returned. Secondary method names must be unique	new line name [name of line to copy attributes from]	**Example of Use:** import vcs a=vcs.init() a.show('line') ln=a.createline('example1',) a.show('line') ln=a.createline('example2','black') a.show('line')

Command	Description	Options	Examples

getline	**Function:** getline
	
	Description of Function:
	VCS contains a list of secondary methods. This function will create a line class object from an existing VCS line secondary method. If no line name is given, then line 'default' will be used.
	Note, VCS does not allow the modification of 'default' attribute sets.
	However, a 'default' attribute set that has been copied under a different name can be modified. (See the createline function.)
		[line name]	**Example of Use:**
			import vcs
			a=vcs.init()
			a.show('line')
			# Show all the existing line secondary methods
			ln=a.getline()
			# ln instance of 'default' line secondary method
			ln2=a.getline('quick')
			# ln2 instance of existing 'quick' line secondary method

Command	Description	Options	Examples

	Object: lineobject
	
	Description of Function:
	The Line object allows the manipulation of line type, width, and color index.
	This class is used to define an line table entry used in VCS, or itcan be used to change some or all of the line attributes in an existing line table entry.
	Other Useful Functions:
	a=vcs.init()
	# Constructor
	a.show('line')
	# Show predefined line objects
	a.update()
	# Updates the VCS Canvas at user's request a.mode=1, or 0
	# If 1, then automatic update, else if 0, then use update function to update the VCS Canvas.
		name
		type
		width
		color
			Example of Use:
			import vcs
			a=vcs.init()
			
			# To Create a new instance of line use:
			ln=a.createline('new','red')
			# Copies content of 'red' to 'new'
			ln=a.createline('new')
			# Copies content of 'default' to 'new'
			
			# To Modify an existing line use:
			ln=a.getline('red')
			ln.list()
			# Will list all the line attribute values
			ln.color=100
			# Range from 1 to 256
			ln.width=100
			# Range from 1 to 300
			
			# Specify the line type:
			ln.type='solid'
			# Same as ln.type=0
			ln.type='dash'
			# Same as ln.type=1
			ln.type='dot'
			# Same as ln.type=2
			ln.type='dash-dot'
			# Same as ln.type=3
			ln.type='long-dash'
			# Same as ln.type=4

Command	Description	Options	Examples
	Marker		
createmarker	**Function:** createmarker **Description of Function:** Create a new marker secondary method given the the name and the existing marker secondary method to copy the attributes from. If no existing marker secondary method name is given, then the default marker secondary method will be used as the secondary method to which the attributes will be copied from. If the name provided already exists, then a error will be returned. Secondary method names must be unique.	new marker name [name of marker to copy attributes from]	**Example of Use:** import vcs a=vcs.init() a.show('marker') mrk=a.createmarker('example1',) a.show('marker') mrk=a.createmarker('example2','black') a.show('boxfill')

getmarker	**Function:** getmarker
	Description of Function:
	VCS contains a list of secondary methods. This function will create a marker class object from an existing VCS marker secondary method. If no marker name is given, then marker 'default' will be used.
	Note, VCS does not allow the modification of 'default' attribute sets.
	However, a 'default' attribute set that has been copied under a different name can be modified. (See the createmarker function.)
		[marker name]	**Example of Use:**
			import vcs
			a=vcs.init()
			a.show('marker')
			# Show all the existing marker secondary methods
			mrk=a.getmarker()
			# mrk instance of 'default' marker secondary method
			mrk2=a.getmarker('quick')
			# mrk2 instance of existing 'quick' marker secondary method

Command	Description	Options	Examples

	Object: markerobject
	
	Description of Function:
	The Marker object allows the manipulation of marker type, size, and color index.
	This class is used to define an marker table entry used in VCS, or it can be used to change some or all of the marker attributes in an existing marker table entry.
	Other Useful Functions:
	a=vcs.init()
	# Constructor
	a.show('marker')
	# Show predefined marker objects
	a.update()
	# Updates the VCS Canvas at user's request a.mode=1, or 0. If 1, then automatic update, else if 0, then use update function to update the VCS Canvas.
		name
		type
		size
		color
			Example of Use:
			import vcs
			a=vcs.init()
			
			# To Create a new instance of marker use:
			mk=a.createmarker('new','red')
			# Copies content of 'red' to 'new'
			mk=a.createmarker('new')
			# Copies content of 'default' to 'new'
			
			# To Modify an existing marker use:
			mk=a.getmarker('red')
			
			mk.list()
			# Will list all the marker attribute values
			mk.color=100
			# Range from 1 to 256
			mk.size=100
			# Range from 1 to 300
			
			# Specify the marker type:
			mk.type='dot'
			# Same as mk.type=1
			mk.type='plus'
			# Same as mk.type=2
			mk.type='star'
			# Same as mk.type=3
			mk.type='circle'
			# Same as mk.type=4
			mk.type='cross'
			# Same as mk.type=5
			mk.type='diamond'
			# Same as mk.type=6
			mk.type='triangle_up'
			# Same as mk.type=7

Command	Description	Options	Examples

			mk.type='triangle_down' # Same as mk.type=8
			mk.type='triangle_left' # Same as mk.type=9
			mk.type='triangle_right' # Same as mk.type=10
			mk.type='square'
			# Same as mk.type=11
			mk.type='diamond_fill' # Same as mk.type=12
			mk.type='triangle_up_fill' # Same as mk.type=13
			mk.type='triangle_down_fill' # Same as mk.type=14
			mk.type='triangle_left_fill' # Same as mk.type=15
			mk.type='triangle_right_fill' # Same as mk.type=16
			mk.type='square_fill' # Same as mk.type=17

Command	Description	Options	Examples
		Text-Combined	
createtext-combined	**Function:** createtextcombined **Description of Function:** Create a new textcombined secondary method given the the names and the existing texttable and textorientation secondary methods to copy the attributes from. If no existing texttable and textorientation secondary method names are given, then the default texttable and textorientation secondary methods will be used as the secondary method to which the attributes will be copied from. If the name provided already exists, then a error will be returned. Secondary method names must be unique.	new textcombined name [name of textcombined to copy attributes from]	**Example of Use:** import vcs a=vcs.init() a.show('texttable') a.show('textorientation') tc=a.createtextcombined('example1','std','example1','7left') a.show('texttable') a.show('textorientation')

Command	Description	Options	Examples

gettext-combined	**Function:** gettextcombined
	
	Description of Function:
	VCS contains a list of secondary methods. This function will create a textcombined class object from an existing VCS texttable secondary method and an existing VCS textorientation secondary method. If no texttable or textorientation names are given, then the 'default' names will be used in both cases.
	Note, VCS does not allow the modification of 'default' attribute sets.
	However, a 'default' attribute set that has been copied under a different name can be modified. (See the createtextcombined function.)
		[textcombined name]	**Example of Use:**
			import vcs
			a=vcs.init()
			a.show('texttable')
			# Show all the existing texttable secondary methods
			a.show('textorientation')
			# Show all the existing textorientation secondary methods
			tc=a.gettextcombined()
			# Use 'default' for texttable and textorientation
			tc2=a.gettextcombined('std','7left')
			# Use 'std' texttable and '7left' textorientation
			if istextcombined(tc):
			# Check to see if tc is a textcombined
			tc.list()
			# Print out all its attriubtes

Command	Description	Options	Examples

	Object: textcombinedobject
	
	Description of Function:
	The (Tc) Text Combined class will combine a text table class and a text orientation class together. From combining the two classess, the user will be able to set attributes for both classes (i.e., define the font, spacing, expansion, color index, height, angle, path, vertical alignment, and horizontal alignment).
	This class is used to define and list a combined text table and text orientation
	entry used in VCS.
		name
		font
		spacing
		expansion
		color
		name
		height
		angel
		path
		halign
		valign
			Example of Use:
			import vcs
			a=vcs.init()
			
			# To Create a new instance of text table use:
			tc=a.createtextcombined('new_tt','std','new_to','7left')
			# Copies content of 'std' to 'new_tt' and '7left' to 'new_to'
			
			# To Modify an existing texttable use:
			tc=a.gettextcombined('std','7left')
			tc.list()
			# Will list all the textcombined attribute values (i.e., texttable and textorientation attributes
			
			# Specify the text font type:
			tc.font=1
			# The font value must be in the range 1 to 9
			
			#Specify the text spacing:
			tc.spacing=2
			# The spacing value must be in the range -50 to 50
			
			# Specify the text expansion:
			tc.expansion=100
			# The expansion value ranges from 50 to 150
			
			# Specify the text color:
			tc.color=241
			# The text color value ranges from 1 to 257

Command	Description	Options	Examples

			# Specify the text height:
			tc.height=20 # The height value must be in the range 1 to 100
			# Specify the text angle:
			tc.angle=0 # The angle value ran# Specify the text path:
			tc.path='right' # Same as tc.path=0
			tc.path='left' # Same as tc.path=1
			tc.path='up' # Same as tc.path=2 ges from 0 to 360
			tc.path='down' # Same as tc.path=3
			# Specify the text horizontal alignment:
			tc.halign='right' # Same as tc.halign=0
			tc.halign='center' # Same as tc.halign=1
			tc.halign='right' # Same as tc.halign=2
			# Specify the text vertical alignment:
			tc.valign='tcp' # Same as tcvalign=0
			tc.valign='cap' # Same as tcvalign=1
			tc.valign='half' # Same as tcvalign=2
			tc.valign='base' # Same as tcvalign=3
			tc.valign='bottom' # Same as tcvalign=4

Command	Description	Options	Examples
	Text-Orientation		
createtex-torientation	**Function:** createtextorientation **Description of Function:** Create a new textorientation secondary method given the the name and he existing textorientation secondary method to copy the attributes from. If no existing textorientation secondary method name is given, then the default textorientation secondary method will be used as the secondary method to which the attributes will be copied from. If the name provided already exists, then a error will be returned. Secondary method names must be unique.	new textorientation name [name of textorientation to copy attributes from]	**Example of Use:** import vcs a=vcs.init() a.show('textorientation') to=a.createtextorientation('example1') a.show('textorientation') to=a.createtextorientation('example2','black') a.show('textorientation')

Command	Description	Options	Examples

gettextori-entation	**Function:** gettextorientation
	
	Description of Function:
	VCS contains a list of secondary methods. This function will create a textorientation class object from an existing VCS textorientation secondary method. If no textorientation name is given, then textorientation 'default' will be used.
	Note, VCS does not allow the modification of 'default' attribute sets.
	However, a 'default' attribute set that has been copied under a different name can be modified. (See the createtextorientation function.)
		[textorienta-tion name]	**Example of Use:**
			import vcs
			a=vcs.init()
			a.show('textorientation')
			# Show all the existing textorientation secondary methods
			to=a.gettextorientation()
			# to instance of 'default' textorientation secondary method
			to2=a.gettextorientation('quick')
			# to2 instance of existing 'quick' textorientation secondary method

Command	Description	Options	Examples

	Object: textorientationobject
	
	Description of Function:
	The (To) Text Orientation lists text attribute set names that define the font, spacing, expansion, and color index.
	This class is used to define an text orientation table entry used in VCS, or it can be used to change some or all of the text orientation attributes in an existing text orientation table entry.
	Other Useful Functions:
	a=vcs.init()
	# Constructor
	a.show('textorientation')
	# Show predefined text orientation objects
	a.update()
	# Updates the VCS Canvas at user's request a.mode=1, or 0. If 1, then automatic update, else if 0, then use update function to update the VCS Canvas.
		name
		height
		angel
		path
		halign
		valign
			Example of Use:
			import vcs
			a=vcs.init()
			
			# To Create a new instance of text orientation use:
			to=a.createtextorientation('new','7left')
			# Copies content of '7left' to 'new'
			to=a.createtextorientation('new')
			# Copies content of 'default' to 'new'
			
			# To Modify an existing textorientation use:
			to=a.gettextorientation('7left')
			to.list()
			# Will list all the textorientation attribute values
			
			# Specify the text height:
			to.height=20
			# The height value must be in the range 1 to 100
			
			#ecify the text angle:
			to.angle=0
			# The angle value must be in the range 0 to 360
			
			# Specify the text path:
			to.path='right'
			# Same as to.path=0
			to.path='left'
			# Same as to.path=1
			to.path='up'
			# Same as to.path=2
			to.path='down'
			# Same as to.path=3

Command	Description	Options	Examples

			# Specify the text horizontal alignment:
			to.halign='right' # Same as to.halign=0
			to.halign='center' # Same as to.halign=1
			to.halign='right'
			# Same as to.halign=2
			# Specify the text vertical alignment:
			to.valign='top' # Same as tov-align=0
			to.valign='cap'
			# Same as tovalign=1
			to.valign='half' # Same as tov-align=2
			to.valign='base' # Same as tov-align=3
			to.valign='bottom'
			# Same as tovalign=4

Command	Description	Options	Examples
	Text-Table		
createtext-table	**Function:** createtexttable **Description of Function:** Create a new texttable secondary method given the the name and the existing texttable secondary method to copy the attributes from. If no existing texttable secondary method name is given, then the default texttable secondary method will be used as the secondary method to which the attributes will be copied from. If the name provided already exists, then a error will be returned. Secondary method names must be unique.	new texttable name [name of text-table to copy attributes from]	**Example of Use:** import vcs a=vcs.init() a.show('texttable') tt=a.createtexttable('example1',) a.show('texttable') tt=a.createtexttable('example2','black') a.show('texttable')

gettexttable	**Function:** gettexttable
	
	Description of Function:
	VCS contains a list of secondary methods. This function will create a texttable class object from an existing VCS texttable secondary method. If no texttable name is given, then texttable 'default' will be used.
	Note, VCS does not allow the modification of 'default' attribute sets.
	However, a 'default' attribute set that has been copied under a different name can be modified. (See the createtexttable function.)
		[texttable name]	**Example of Use:**
			import vcs
			a=vcs.init()
			a.show('texttable')
			# Show all the existing texttable secondary methods
			tt=a.gettexttable()
			# tt instance of 'default' texttable secondary method
			tt2=a.gettexttable('quick')
			# tt2 instance of existing 'quick' texttable secondary method

Command	Description	Options	Examples

	Object: texttableobject
	
	Description of Function:
	The (Tt) Text Table lists text attribute set names that define the font, spacing, expansion, and color index.
	This class is used to define an text table table entry used in VCS, or it can be used to change some or all of the text table attributes in an existing text table table entry.
	Other Useful Functions:
	a=vcs.init()
	# Constructor
	a.show('texttable')
	# Show predefined text table objects
	a.update()
	# Updates the VCS Canvas at user's request a.mode=1, or 0. If 1, then automatic update, else if 0, then use update function to update the VCS Canvas.
		name
		font
		spacing
		expansion
		color
			Example of Use:
			import vcs
			a=vcs.init()
			
			# To Create a new instance of text table use:
			tt=a.createtexttable('new','std')
			# Copies content of 'std' to 'new'
			tt=a.createtexttable('new')
			# Copies content of 'default' to 'new'
			
			# To Modify an existing texttable use:
			tt=a.gettexttable('std')
			tt.list()
			# Will list all the texttable attribute values
			
			# Specify the text font type:
			tt.font=1
			# The font value must be in the range 1 to 9
			
			# Specify the text spacing:
			tt.spacing=2
			# The spacing value must be in the range -50 to 50
			
			# Specify the text expansion:
			tt.expansion=100
			# The expansion value must be in the range 50 to 150
			
			# Specify the text color:
			tt.color=241
			# The text color attribute value must be in the range 1 to 257

Command	Description	Options	Examples
Remove Objects			

removeob-ject	**Function:** removeobject
	
	Description of Function:
	The user has the ability to create primary and secondary class objects. This function allows the user to remove these objects from the appropriate class list.
	Note, To remove the object completely from Python, remember to use the "del" function.
	Also note, The user is not allowed to remove a "default" class object.
		object	**Example of Use:**
			import vcs
			a=vcs.init()
			line=a.getline('red')
			# To Modify an existing line object
			iso=a.createisoline('dean')
			# Create an instance of an isoline object
			...
			a.removeobject(line)
			# Removes line object from VCS list
			del line
			# Destroy instance "line", garbage collection
			a.removeobject(iso)
			# Remove isoline object from VCS list
			del iso
			# Destroy instance "iso", garbage collection

Command	Description	Options	Examples
Set Continents Type			

setcontinentstype	**Function:** setcontinentstype
	
	Description of Function:
	One has the option of using continental maps that are predefined or that are user-defined. Predefined continental maps are either internal to VCS or are specified by external files. User-defined continental maps are specified by additional external files that must be read as input.
	The continents-type values are integers ranging from 0 to 11, where:
	0 signifies "No Continents"
	1 signifies "Fine Continents"
	2 signifies "Coarse Continents"
	3 signifies "United States" (with "Fine Continents")
	4 signifies "Political Borders" (with "Fine Continents")
	5 signifies "Rivers" (with "Fine Continents")
	Values 6 through 11 signify the line type defined by the files
	data_continent_other7 through data_continent_other12.
		continents type number, ranging from 0 to 11	**Example of Use:**
			import vcs
			a=vcs.init()
			# Set continents to “United States”
			a.setcontinentstype(3)
			a.plot(array,’default’,’isofill’,’quick’

Command	Description	Options	Examples
Set Default Picture Template and Graphics Methods			

set	**Function:** set
	
	Description of Function:
	Set the default VCS primary class objects: template and graphics methods.
	Keep in mind the template, determines the appearance of each graphics segment; the graphic method specifies the display technique; and the data defines what is to be displayed. Note, the data cannot be set with this function.
		template or graphics methods type,
		[template name or graphics method name]
			Example of Use:
			import vcs
			a=vcs.init()
			a.set('isofill','quick')
			# Changes the default graphics method to Isofill: 'quick'
			a.plot(array)

Animation			
animate	**Function:** animate **Description of Function:** Animate the contents of the VCS Canvas. Currently, only one display can be shown in the VCS Canvas for the animation to work. This function pops up the animation GUI. Note: The animation GUI will only work for 8-bit 'Pseudo Color'. See the animation GUI documenation located at URL: http://www-pcmdi.llnl.gov/software/vcs		**Example of Use:** import vcs a=vcs.init() a.plot(array,'default','isofill','quick') a.animate()

Command	Description	Options	Examples
		Flush	
flush	**Function:** flush **Description of Function:** The flush command executes all buffered X events in the que.		**Example of Use:** import vcs a=vcs.init() a.plot(array,'default','isofill','quick') a.flush()
		Grid	
grid	**Function:** grid **Description of Function:** Set the default plotting region for variables that have more dimension values than the graphics method. This will also be used for animating plots over the third and fourth dimensions.	([first dim's 1st value, first dim's last value] ,..., [last dim's 1st value, last dim's last value]	**Example of Use:** import vcs a=vcs.init() a=vcs.init() a.grid(12,24, -70,70, -150,150) a.plot(array,'default','isofill','quick')
resetgrid	**Function:** resetgrid **Description of Function:** Set the plotting region to default values. That is, let the variable's dimension values determine the grid		**Example of Use:** import vcs a=vcs.init() a=vcs.init() a.resetgrid() a.plot(array,'default','isofill','quick')

CHAPTER 6 VCS Primary Objects

In VCS, primary objects define what will be displayed on the VCS Canvas. This section defines two of the three primary objects and how to modify them dynamically in the system: graphics methods, which specifies the display technique; and the picture template, which determines the appearance of each segment of the display.

Note, to see the list of object attributes, use the list() function. For example:

```
import vcs
a=vcs.init()
box=a.createboxfill('new')
box.list()          # This call will list the boxfill attributes and their values
isof=a.createisofill('new')
isof.list()         # This call will list the isofill attributes and their values
tpl=a.createtemplate('new')
tpl.list()          # This call will list the template attributes and their values
```

Table 6.1

Object	Attributes (or Members)	Expected Type(s)
boxfill (graphics method class name is Gfb)	name	StringType
	projection	Either: ‘linear’, ‘mollweide’, ‘robinson’, or ‘polar’
	xticlabels1	StringType or DictType
	xticlabels2	StringType or DictType
	xmtics1	StringType or DictType
	xmtics2	StringType or DictType
	yticlabels1	StringType or DictType
	yticlabels2	StringType or DictType
	ymtics1	StringType or DictType
	ymtics2	StringType or DictType
	datawc_x1	IntType or FloatType
	datawc_y1	IntType or FloatType
	datawc_x2	IntType or FloatType
	datawc_y2	IntType or FloatType
	xaxisconvert	Either: ’linear’, ’log10’, ’ln’,’exp’,’area_wt’
	yaxisconvert	Either: ’linear’, ’log10’, ’ln’,’exp’,’area_wt’
	level_1	IntType or FloatType
	level_2	IntType or FloatType
	color_1	IntType: (Range: 0 to 255)
	color_2	IntType: (Range: 0 to 255)
	legend_type	Either: ’VCS’, ’Pts’, ’List’
	legend	StringType
	ext_1	Either: ’n’, ’y’
	ext_2	Either: ’n’, ’y’
	missing	IntType: (Range: 0 to 255)

Table 6.1

Object	Attributes (or Members)	Expected Type(s)
continents (graphics method class name is Gcon)	name	StringType
	projection	Either: ‘linear’, ‘mollweide’, ‘robinson’, or ‘polar’
	xticlabels1	StringType or DictType
	xticlabels2	StringType or DictType
	xmtics1	StringType or DictType
	xmtics2	StringType or DictType
	yticlabels1	StringType or DictType
	yticlabels2	StringType or DictType
	ymtics1	StringType or DictType
	ymtics2	StringType or DictType
	datawc_x1	IntType or FloatType
	datawc_y1	IntType or FloatType
	datawc_x2	IntType or FloatType
	datawc_y2	IntType or FloatType
	line	LineObject or (Either: ’solid’,’dash’,’dot’, ’dash-dot’,’long-dash’, 0,1,2,3,4)
	linecolor	IntType: (Range 0 to 255)
	type	IntType: (Range 0 to 11)

Table 6.1

Object	Attributes (or Members)	Expected Type(s)
isofill (graphics method class name is Gfi)	name	StringType
	projection	Either: ‘linear’, ‘mollweide’, ‘robinson’, or ‘polar’
	xticlabels1	StringType or DictType
	xticlabels2	StringType or DictType
	xmtics1	StringType or DictType
	xmtics2	StringType or DictType
	yticlabels1	StringType or DictType
	yticlabels2	StringType or DictType
	ymtics1	StringType or DictType
	ymtics2	StringType or DictType
	datawc_x1	IntType or FloatType
	datawc_y1	IntType or FloatType
	datawc_x2	IntType or FloatType
	datawc_y2	IntType or FloatType
	xaxisconvert	Either: ’linear’, ’log10’, ’ln’,’exp’,’area_wt’
	yaxisconvert	Either: ’linear’, ’log10’, ’ln’,’exp’,’area_wt’
	missing	IntType: (Range 0 to 255)
	ext_1	Either: ’n’, ’y’
	ext_2	Either: ’n’, ’y’
	fillareaindices	FillareaObject, or (Either: ListType, or TupleType [Values range 1 to 20])
	fillareastyle	Either: ’solid’, ’hatch’, ’pattern’
	fillareacolors	ListType, or TupleType (Index values range 0 to 255)
	levels	ListType, TupleType (Values are either integers, floats)

Table 6.1

Object	Attributes (or Members)	Expected Type(s)
isoline (graphics method class name is Gi)	name	StringType
	projection	Either: ‘linear’, ‘mollweide’, ‘robinson’, or ‘polar’
	xticlabels1	StringType or DictType
	xticlabels2	StringType or DictType
	xmtics1	StringType or DictType
	xmtics2	StringType or DictType
	yticlabels1	StringType or DictType
	yticlabels2	StringType or DictType
	ymtics1	StringType or DictType
	ymtics2	StringType or DictType
	datawc_x1	IntType or FloatType
	datawc_y1	IntType or FloatType
	datawc_x2	IntType or FloatType
	datawc_y2	IntType or FloatType
	xaxisconvert	Either: ’linear’, ’log10’, ’ln’,’exp’,’area_wt’
	yaxisconvert	Either: ’linear’, ’log10’, ’ln’,’exp’,’area_wt’
	label	Either: ’n’, ’y’, 0, 1
	line	LineObject or (Either: "solid", "dash", "dot", "dash-dot", "long-dash", 0, 1, 2, 3, 4)
	linecolors	ListType, TupleType (Values range 0 to 255)
	text	TextcombinedObject, TextOrientationObject, TexttableObject, or (Either: ListType, TupleType, IntType [Values range 1 to 9])
	textcolors	ListType, TupleType (Values range 0 to 255)
	level	ListType, TupleType, IntType, FloatType

Table 6.1

Object	Attributes (or Members)	Expected Type(s)
outfill (graphics method class name is Gfo)	name	StringType
	projection	Either: ‘linear’, ‘mollweide’, ‘robinson’, or ‘polar’
	xticlabels1	StringType or DictType
	xticlabels2	StringType or DictType
	xmtics1	StringType or DictType
	xmtics2	StringType or DictType
	yticlabels1	StringType or DictType
	yticlabels2	StringType or DictType
	ymtics1	StringType or DictType
	ymtics2	StringType or DictType
	datawc_x1	IntType or FloatType
	datawc_y1	IntType or FloatType
	datawc_x2	IntType or FloatType
	datawc_y2	IntType or FloatType
	xaxisconvert	Either: ’linear’, ’log10’, ’ln’,’exp’,’area_wt’
	yaxisconvert	Either: ’linear’, ’log10’, ’ln’,’exp’,’area_wt’
	fillareastyle	FillareaObject or (Either: ’solid’, ’hatch’, ’pattern’, 0, 1, 2, 3)
	fillareaindex	IntType: (Range 0 to 20)
	fillareacolor	IntType: (Range 0 to 255)
	outfill	ListType, TupleType, IntType, FloatType (must be less than 10 values)

Table 6.1

Object	Attributes (or Members)	Expected Type(s)
outline (graphics method class name is Go)	name	StringType
	projection	Either: ‘linear’, ‘mollweide’, ‘robinson’, or ‘polar’
	xticlabels1	StringType or DictType
	xticlabels2	StringType or DictType
	xmtics1	StringType or DictType
	xmtics2	StringType or DictType
	yticlabels1	StringType or DictType
	yticlabels2	StringType or DictType
	ymtics1	StringType or DictType
	ymtics2	StringType or DictType
	datawc_x1	IntType or FloatType
	datawc_y1	IntType or FloatType
	datawc_x2	IntType or FloatType
	datawc_y2	IntType or FloatType
	xaxisconvert	Either: ’linear’, ’log10’, ’ln’,’exp’,’area_wt’
	yaxisconvert	Either: ’linear’, ’log10’, ’ln’,’exp’,’area_wt’
	line	LineObject or (Either: "solid", "dash", "dot", "dash-dot", "long-dash", 0, 1, 2, 3, 4)
	linecolor	ListType, TupleType (Values range 0 to 255)
	outline	ListType, TupleType, IntType, FloatType (must be less than 10 values)

Table 6.1

Object	Attributes (or Members)	Expected Type(s)
scatter (graphics method class name is GSp)	name	StringType
	projection	Either: ‘linear’, ‘mollweide’, ‘robinson’, or ‘polar’
	xticlabels1	StringType or DictType
	xticlabels2	StringType or DictType
	xmtics1	StringType or DictType
	xmtics2	StringType or DictType
	yticlabels1	StringType or DictType
	yticlabels2	StringType or DictType
	ymtics1	StringType or DictType
	ymtics2	StringType or DictType
	datawc_x1	IntType or FloatType
	datawc_y1	IntType or FloatType
	datawc_x2	IntType or FloatType
	datawc_y2	IntType or FloatType
	xaxisconvert	Either: ’linear’, ’log10’, ’ln’,’exp’,’area_wt’
	yaxisconvert	Either: ’linear’, ’log10’, ’ln’,’exp’,’area_wt’
	marker	MarkerObject or (Either: ’dot’, ’plus’, ’star’, ’circle’, ’cross’, ’diamond’, ’triangle_up’, ’triangle_down’, ’triangle_left’, ’triangle_right’, ’square’, ’diamond_fill’, ’triangle_up_fill’, ’triangle_down_fill’, ’triangle_left_fill’, ’triangle_right_fill’, ’square_fill’, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14 ,15, 16, 17
	markercolor	IntType: (Range 0 to 255)
	markersize	IntType: (Range 1 to 300)

Table 6.1

Object	Attributes (or Members)	Expected Type(s)
vector (graphics method class name is Gv)	name	StringType
	projection	Either: ‘linear’, ‘mollweide’, ‘robinson’, or ‘polar’
	xticlabels1	StringType or DictType
	xticlabels2	StringType or DictType
	xmtics1	StringType or DictType
	xmtics2	StringType or DictType
	yticlabels1	StringType or DictType
	yticlabels2	StringType or DictType
	ymtics1	StringType or DictType
	ymtics2	StringType or DictType
	datawc_x1	IntType or FloatType
	datawc_y1	IntType or FloatType
	datawc_x2	IntType or FloatType
	datawc_y2	IntType or FloatType
	xaxisconvert	Either: ’linear’, ’log10’, ’ln’,’exp’,’area_wt’
	yaxisconvert	Either: ’linear’, ’log10’, ’ln’,’exp’,’area_wt’
	line	LineObject or (Either: "solid", "dash", "dot", "dash-dot", "long-dash", 0, 1, 2, 3, 4)
	linecolor	ListType, TupleType (Values range 0 to 255)
	scale	IntType, FloatType
	alignment	Either: ’head’, ’center’, ’tail’, 0, 1, 2
	type	Either: ’arrows’, ’barbs’, ’solidarrows’, 0, 1, 2
	reference	FloatType, IntType

Table 6.1

Object	Attributes (or Members)	Expected Type(s)
xvsy (graphics method class name is GXY)	name	StringType
	projection	Either: ‘linear’, ‘mollweide’, ‘robinson’, or ‘polar’
	xticlabels1	StringType or DictType
	xticlabels2	StringType or DictType
	xmtics1	StringType or DictType
	xmtics2	StringType or DictType
	yticlabels1	StringType or DictType
	yticlabels2	StringType or DictType
	ymtics1	StringType or DictType
	ymtics2	StringType or DictType
	datawc_x1	IntType or FloatType
	datawc_y1	IntType or FloatType
	datawc_x2	IntType or FloatType
	datawc_y2	IntType or FloatType
	xaxisconvert	Either: ’linear’, ’log10’, ’ln’,’exp’,’area_wt’
	yaxisconvert	Either: ’linear’, ’log10’, ’ln’,’exp’,’area_wt’
	line	LineObject or (Either: "solid", "dash", "dot", "dash-dot", "long-dash", 0, 1, 2, 3, 4)
	linecolor	ListType, TupleType (Values range 0 to 255)
	marker	MarkerObject or (Either: ’dot’, ’plus’, ’star’, ’circle’, ’cross’, ’diamond’, ’triangle_up’, ’triangle_down’, ’triangle_left’, ’triangle_right’, ’square’, ’diamond_fill’, ’triangle_up_fill’, ’triangle_down_fill’, ’triangle_left_fill’, ’triangle_right_fill’, ’square_fill’, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14 ,15, 16, 17
	markercolor	IntType: (Range 0 to 255)
	markersize	IntType: (Range 1 to 300)

Table 6.1

Object	Attributes (or Members)	Expected Type(s)
xyvsy (graphics method class name is GXy)	name	StringType
	projection	Either: ‘linear’, ‘mollweide’, ‘robinson’, or ‘polar’
	xticlabels1	StringType or DictType
	xticlabels2	StringType or DictType
	xmtics1	StringType or DictType
	xmtics2	StringType or DictType
	yticlabels1	StringType or DictType
	yticlabels2	StringType or DictType
	ymtics1	StringType or DictType
	ymtics2	StringType or DictType
	datawc_x1	IntType or FloatType
	datawc_y1	IntType or FloatType
	datawc_x2	IntType or FloatType
	datawc_y2	IntType or FloatType
	xaxisconvert	Either: ’linear’, ’log10’, ’ln’,’exp’,’area_wt’
	line	LineObject or (Either: "solid", "dash", "dot", "dash-dot", "long-dash", 0, 1, 2, 3, 4)
	linecolor	ListType, TupleType (Values range 0 to 255)
	marker	MarkerObject or (Either: ’dot’, ’plus’, ’star’, ’circle’, ’cross’, ’diamond’, ’triangle_up’, ’triangle_down’, ’triangle_left’, ’triangle_right’, ’square’, ’diamond_fill’, ’triangle_up_fill’, ’triangle_down_fill’, ’triangle_left_fill’, ’triangle_right_fill’, ’square_fill’, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14 ,15, 16, 17
	markercolor	IntType: (Range 0 to 255)
	markersize	IntType: (Range 1 to 300)

Table 6.1

Object	Attributes (or Members)	Expected Type(s)
yxvsx (graphics method class name is GYx)	name	StringType
	projection	Either: ‘linear’, ‘mollweide’, ‘robinson’, or ‘polar’
	xticlabels1	StringType or DictType
	xticlabels2	StringType or DictType
	xmtics1	StringType or DictType
	xmtics2	StringType or DictType
	yticlabels1	StringType or DictType
	yticlabels2	StringType or DictType
	ymtics1	StringType or DictType
	ymtics2	StringType or DictType
	datawc_x1	IntType or FloatType
	datawc_y1	IntType or FloatType
	datawc_x2	IntType or FloatType
	datawc_y2	IntType or FloatType
	yaxisconvert	Either: ’linear’, ’log10’, ’ln’,’exp’,’area_wt’
	line	LineObject or (Either: "solid", "dash", "dot", "dash-dot", "long-dash", 0, 1, 2, 3, 4)
	linecolor	ListType, TupleType (Values range 0 to 255)
	marker	MarkerObject or (Either: ’dot’, ’plus’, ’star’, ’circle’, ’cross’, ’diamond’, ’triangle_up’, ’triangle_down’, ’triangle_left’, ’triangle_right’, ’square’, ’diamond_fill’, ’triangle_up_fill’, ’triangle_down_fill’, ’triangle_left_fill’, ’triangle_right_fill’, ’square_fill’, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14 ,15, 16, 17
	markercolor	IntType: (Range 0 to 255)
	markersize	IntType: (Range 1 to 300)

Table 6.1

Object	Attributes (or Members)	Expected Type(s)
template (Picture Template class name is P)	member = file	
	priority	IntType
	x	IntType, FloatType
	y	IntType, FloatType
	texttable	TexttableObject, StringType
	textorientation	TextorientationObject, StringType
	member = function	
	priority	IntType
	x	IntType, FloatType
	y	IntType, FloatType
	texttable	TexttableObject, StringType
	textorientation	TextorientationObject, StringType
	member = logicalmask	
	priority	IntType
	x	IntType, FloatType
	y	IntType, FloatType
	texttable	TexttableObject, StringType
	textorientation	TextorientationObject, StringType
	member = transformation	
	priority	IntType
	x	IntType, FloatType
	y	IntType, FloatType
	texttable	TexttableObject, StringType
	textorientation	TextorientationObject, StringType
	member = source	
	priority	IntType
	x	IntType, FloatType
	y	IntType, FloatType
	texttable	TexttableObject, StringType
	textorientation	TextorientationObject, StringType

Table 6.1

Object	Attributes (or Members)	Expected Type(s)
	member = dataname	
	priority	IntType
	x	IntType, FloatType
	y	IntType, FloatType
	texttable	TexttableObject, StringType
	textorientation	TextorientationObject, StringType
	member = title	
	priority	IntType
	x	IntType, FloatType
	y	IntType, FloatType
	texttable	TexttableObject, StringType
	textorientation	TextorientationObject, StringType
	member = units	
	priority	IntType
	x	IntType, FloatType
	y	IntType, FloatType
	texttable	TexttableObject, StringType
	textorientation	TextorientationObject, StringType
	member = crdate	
	priority	IntType
	x	IntType, FloatType
	y	IntType, FloatType
	texttable	TexttableObject, StringType
	textorientation	TextorientationObject, StringType
	member = crtime	
	priority	IntType
	x	IntType, FloatType
	y	IntType, FloatType
	texttable	TexttableObject, StringType
	textorientation	TextorientationObject, StringType

Table 6.1

Object	Attributes (or Members)	Expected Type(s)
	member = comment1	
	priority	IntType
	x	IntType, FloatType
	y	IntType, FloatType
	texttable	TexttableObject, StringType
	textorientation	TextorientationObject, StringType
	member = comment2	
	priority	IntType
	x	IntType, FloatType
	y	IntType, FloatType
	texttable	TexttableObject, StringType
	textorientation	TextorientationObject, StringType
	member = comment3	
	priority	IntType
	x	IntType, FloatType
	y	IntType, FloatType
	texttable	TexttableObject, StringType
	textorientation	TextorientationObject, StringType
	member = comment4	
	priority	IntType
	x	IntType, FloatType
	y	IntType, FloatType
	texttable	TexttableObject, StringType
	textorientation	TextorientationObject, StringType
	member = xname	
	priority	IntType
	x	IntType, FloatType
	y	IntType, FloatType
	texttable	TexttableObject, StringType
	textorientation	TextorientationObject, StringType

Table 6.1

Object	Attributes (or Members)	Expected Type(s)
	member = yname	
	priority	IntType
	x	IntType, FloatType
	y	IntType, FloatType
	texttable	TexttableObject, StringType
	textorientation	TextorientationObject, StringType
	member = zname	
	priority	IntType
	x	IntType, FloatType
	y	IntType, FloatType
	texttable	TexttableObject, StringType
	textorientation	TextorientationObject, StringType
	member = tname	
	priority	IntType
	x	IntType, FloatType
	y	IntType, FloatType
	texttable	TexttableObject, StringType
	textorientation	TextorientationObject, StringType
	member = xunits	
	priority	IntType
	x	IntType, FloatType
	y	IntType, FloatType
	texttable	TexttableObject, StringType
	textorientation	TextorientationObject, StringType
	member = yunits	
	priority	IntType
	x	IntType, FloatType
	y	IntType, FloatType
	texttable	TexttableObject, StringType
	textorientation	TextorientationObject, StringType

Table 6.1

Object	Attributes (or Members)	Expected Type(s)
	member = zunits	
	priority	IntType
	x	IntType, FloatType
	y	IntType, FloatType
	texttable	TexttableObject, StringType
	textorientation	TextorientationObject, StringType
	member = tunits	
	priority	IntType
	x	IntType, FloatType
	y	IntType, FloatType
	texttable	TexttableObject, StringType
	textorientation	TextorientationObject, StringType
	member = xvalue	
	priority	IntType
	x	IntType, FloatType
	y	IntType, FloatType
	format	Currently not able to set
	texttable	TexttableObject, StringType
	textorientation	TextorientationObject, StringType
	member = yvalue	
	priority	IntType
	x	IntType, FloatType
	y	IntType, FloatType
	format	Currently not able to set
	texttable	TexttableObject, StringType
	textorientation	TextorientationObject, StringType

Table 6.1

Object	Attributes (or Members)	Expected Type(s)
	member = zvalue	
	priority	IntType
	x	IntType, FloatType
	y	IntType, FloatType
	format	Currently not able to set
	texttable	TexttableObject, StringType
	textorientation	TextorientationObject, StringType
	member = tvalue	
	priority	IntType
	x	IntType, FloatType
	y	IntType, FloatType
	format	Currently not able to set
	texttable	TexttableObject, StringType
	textorientation	TextorientationObject, StringType
	member = mean	
	priority	IntType
	x	IntType, FloatType
	y	IntType, FloatType
	format	Currently not able to set
	texttable	TexttableObject, StringType
	textorientation	TextorientationObject, StringType
	member = min	
	priority	IntType
	x	IntType, FloatType
	y	IntType, FloatType
	format	Currently not able to set
	texttable	TexttableObject, StringType
	textorientation	TextorientationObject, StringType

Table 6.1

Object	Attributes (or Members)	Expected Type(s)
	member = max	
	priority	IntType
	x	IntType, FloatType
	y	IntType, FloatType
	format	Currently not able to set
	texttable	TexttableObject, StringType
	textorientation	TextorientationObject, StringType
	member = xtic1	
	priority	IntType
	y1	IntType, FloatType
	y2	IntType, FloatType
	line	LineObject, StringType
	member = xtic2	
	priority	IntType
	y1	IntType, FloatType
	y2	IntType, FloatType
	line	LineObject, StringType
	member = xmintic1	
	priority	IntType
	y1	IntType, FloatType
	y2	IntType, FloatType
	line	LineObject, StringType
	member = xmintic2	
	priority	IntType
	y1	IntType, FloatType
	y2	IntType, FloatType
	line	LineObject, StringType

Table 6.1

Object	Attributes (or Members)	Expected Type(s)
	member = ytic1	
	priority	IntType
	x1	IntType, FloatType
	x2	IntType, FloatType
	line	LineObject, StringType
	member = ytic2	
	priority	IntType
	x1	IntType, FloatType
	x2	IntType, FloatType
	line	LineObject, StringType
	member = ymintic1	
	priority	IntType
	x1	IntType, FloatType
	x2	IntType, FloatType
	line	LineObject, StringType
	member = ymintic2	
	priority	IntType
	x1	IntType, FloatType
	x2	IntType, FloatType
	line	LineObject, StringType
	member = xlabel1	
	priority	IntType
	y	IntType, FloatType
	texttable	TexttableObject, StringType
	textorientation	TextorientationObject, StringType
	member = xlabel2	
	priority	IntType
	y	IntType, FloatType
	texttable	TexttableObject, StringType
	textorientation	TextorientationObject, StringType

Table 6.1

Object	Attributes (or Members)	Expected Type(s)
	member = ylabel1	
	priority	IntType
	x	IntType, FloatType
	texttable	TexttableObject, StringType
	textorientation	TextorientationObject, StringType
	member = ylabel2	
	priority	IntType
	x	IntType, FloatType
	texttable	TexttableObject, StringType
	textorientation	TextorientationObject, StringType
	member = box1	
	priority	IntType
	x1	IntType, FloatType
	y1	IntType, FloatType
	x2	IntType, FloatType
	y2	IntType, FloatType
	line	LineObject, StringType
	member = box2	
	priority	IntType
	x1	IntType, FloatType
	y1	IntType, FloatType
	x2	IntType, FloatType
	y2	IntType, FloatType
	line	LineObject, StringType

Table 6.1

Object	Attributes (or Members)	Expected Type(s)
	member = box3	
	priority	IntType
	x1	IntType, FloatType
	y1	IntType, FloatType
	x2	IntType, FloatType
	y2	IntType, FloatType
	line	LineObject, StringType
	member = box4	
	priority	IntType
	x1	IntType, FloatType
	y1	IntType, FloatType
	x2	IntType, FloatType
	y2	IntType, FloatType
	line	LineObject, StringType
	member = line1	
	priority	IntType
	x1	IntType, FloatType
	y1	IntType, FloatType
	x2	IntType, FloatType
	y2	IntType, FloatType
	line	LineObject, StringType
	member = line2	
	priority	IntType
	x1	IntType, FloatType
	y1	IntType, FloatType
	x2	IntType, FloatType
	y2	IntType, FloatType
	line	LineObject, StringType

Table 6.1

Object	Attributes (or Members)	Expected Type(s)
	member = line3	
	priority	IntType
	x1	IntType, FloatType
	y1	IntType, FloatType
	x2	IntType, FloatType
	y2	IntType, FloatType
	line	LineObject, StringType
	member = line4	
	priority	IntType
	x1	IntType, FloatType
	y1	IntType, FloatType
	x2	IntType, FloatType
	y2	IntType, FloatType
	line	LineObject, StringType
	member = legend	
	priority	IntType
	x1	IntType, FloatType
	y1	IntType, FloatType
	x2	IntType, FloatType
	y2	IntType, FloatType
	line	LineObject, StringType
	texttable	TexttableObject, StringType
	textorientation	TextorientationObject, StringType
	member = data	
	priority	IntType
	x1	IntType, FloatType
	y1	IntType, FloatType
	x2	IntType, FloatType
	y2	IntType, FloatType

CHAPTER 7

VCS Secondary Objects

Secondary objects help to specify and define primary objects in VCS. The list of secondary objects include: colormap, fill area, format, line, marker, list, text-combined, text-orientation, and text-table. Note, although the colormap is considered a secondary object, it is acessed differenly. Therefore, the colormap object will not be described below. See the “VCS Command Reference Guide” for colormap commands.

Note, to see the list of object attributes, use the list() function. For example:

```
import vcs
a=vcs.init()
ln=a.createline(‘new’)
ln.list()          # This call will list the line attributes and their values
tt=a.createtexttable(‘new’)
tt.list()              # This call will list the text-table attributes and their values
```

Table 7.1

Object	Attributes (or Members)	Exception Type(s)
fillarea (graphics method class name is Tf)	name style index color	StringType Either: ’solid’, ’hatch’, ’pattern’ IntType: (Range 1 to 20) IntType: (Range 0 to 255)
line (graphics method class name is Tl)	name type width color	StringType Either: ’solid’, ’dash’, ’dot’, ’dash-dot’, ’long-dash’, 0, 1, 2, 3, 4 IntType: (Range 1 to 300) IntType: (Range 0 to 255)

Table 7.1

Object	Attributes (or Members)	Exception Type(s)
marker (graphics method class name is Tm)	name	StringType
	type	Either: "dot", "plus", "star", "circle", "cross", "diamond", "triangle_up", "triangle_down", "triangle_left", "triangle_right", "square", "diamond_fill", "triangle_up_fill", "triangle_down_fill", "triangle_left_fill", "triangle_right_fill", "square_fill", 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17
	size	IntType: (Range 1 to 300)
	color	IntType: (Range 0 to 255)
text-combined (graphics method class name is Tc)	name	StringType
	font	IntType: (Range 1 to 9)
	spacing	IntType: (Range -50 to 50)
	expansion	IntType: (Range 50 to 150)
	color	IntType: (Range 0 to 255)
	name	StringType
	height	IntType: (Range 1 to 100)
	angel	IntType: (Range 0 to 360)
	path	Either: ’right’, ’left’, ’up’, ’down’, 0, 1, 2, 3
	halign	Either: ’left’, ’center’, ’right’, 0, 1, 2
	valign	Either: ’top’, ’cap’, ’half’, ’base’, ’bottom’, 0, 1, 2, 3, 4
text-orientation (graphics method class name is To)	name	StringType
	height	IntType: (Range 1 to 100)
	angel	IntType: (Range 0 to 360)
	path	Either: ’right’, ’left’, ’up’, ’down’, 0, 1, 2, 3
	halign	Either: ’left’, ’center’, ’right’, 0, 1, 2
	valign	Either: ’top’, ’cap’, ’half’, ’base’, ’bottom’, 0, 1, 2, 3, 4

Table 7.1

Object	Attributes (or Members)	Exception Type(s)
text-table (graphics method class name is Tt)	name font spacing expansion color	StringType IntType: (Range 1 to 9) IntType: (Range -50 to 50) IntType: (Range 50 to 150) IntType: (Range 0 to 255)

CHAPTER 8 VCS Examples

This section shows useful VCS examples.

Simple Plotting Example:

```
#
# Simple Plot module
#
##############################################################################
#                                                                            #
# Module:       simpleplot module                                             #
#                                                                            #
# Copyright:    2000, Regents of the University of California                 #
#               This software may not be distributed to others without       #
#               permission of the author.                                     #
#                                                                            #
# Author:       Dean N. Williams, Lawrence Livermore National Laboratory     #
#               williams13@llnl.gov                                          #
#                                                                            #
# Description:  Simple plotting example.                                      #
#                                                                            #
# Version:      1.0                                                           #
#                                                                            #
##############################################################################
#
#
#
##############################################################################
#                                                                            #
# Import: vcs  and cdms modules.                                              #
#                                                                            #
##############################################################################
def simpleplot():
   import vcs,cdms                      # import vcs and cdms

   ###########################################################################
```

```
    # See the CDMS document on how to ingest data. Also see the CU and  #
    # the Numeric documents on alternative ways to import data into VCS #
    ####################################################################

    f=cdms.openDataset('example.nc')     # open example file
    clt=f.variables['clt']               # get variable clt
    s=clt[0]                            # get clt data

    ####################################################################
    # Basically to plot using the VCS module, three steps are required:   #
    # importing the vcs module; initializing the VCS Canvas Object, and   #
    # plotting the data on the VCS Canvas.                                #
   ####################################################################
    x=vcs.init()                         # initialize vcs
    x.plot(s,variable=clt)               # plot data using default values

    print '*************************************************************'
    print '******                                                 ******'
    print '******   S I M P L E   P L O T T I N G   C O M P L E T E D   ******'
    print '******                                                 ******'
    print '*************************************************************'

if __name__=="__main__":
    simpleplot()
```

Simple Overlay Plot Example:

```
#
# Simple Overlay Plot module
#
##############################################################################
#                                                                            #
# Module:       simpleoverlay module                                         #
#                                                                            #
# Copyright:    2000, Regents of the University of California                 #
#               This software may not be distributed to others without       #
#               permission of the author.                                    #
#                                                                            #
# Author:       Dean N. Williams, Lawrence Livermore National Laboratory     #
#               williams13@llnl.gov                                          #
#                                                                            #
# Description:  Simple overlay plotting example                              #
#                                                                            #
```

```
# Version:      1.0                                                          #
#                                                                            #
##############################################################################
#
#
#
##############################################################################
#                                                                            #
# Import: vcs  and cdms modules.                                              #
#                                                                            #
##############################################################################
def simpleoverlay():
   import vcs,cdms                      # import vcs and cdms

   ###########################################################################
   # See the CDMS document on how to ingest data. Also see the CU and  #
   # the Numeric documents on alternative ways to import data into VCS #
   ###########################################################################

   f=cdms.openDataset('example.nc')     # open example file
   clt=f.variables['clt']               # get variable clt
   s=clt[0]                            # get clt data

   ###########################################################################
   # Basically to plot using the VCS module, three steps are required:    #
   # importing the vcs module; initializing the VCS Canvas Object, and    #
   # plot the data on the VCS Canvas.                                      #
   #                                                                      #
   # Note:                                                                #
   #  In the example below, we are using isofill and isoline to plot the  #
   #  data. We could just as easily used the 'plot()' function to achieve #
   #  the same result:                                                    #
   #  x.plot(s,'default', 'isofill', variable=clt)                         #
   #  x.plot(s,'default_dud', 'isoline', variable=clt)                    #
   # Note:                                                                #
   #   'default' and 'default_dud' are passed as the second argument in   #
   #   plot routines, respectfully. 'default' represents a VCS template   #
   #   that displays the text and plot lengend, while 'default_dud' is a  #
   #   VCS template that will only display the data on the VCS Canvas.    #
  ###########################################################################
   x=vcs.init()                                # initialize vcs
   x.isofill(s,'default',variable=clt)      # plot data using default values
   x.isoline(s,'default_dud',variable=clt)  # overlay isolines over isofill plot

   print '*************************************************************'
   print '******                                                 ******'
```

```
   print '******   S I M P L E   O V E R L A Y   C O M P L E T E D   ******'
   print '******                                                     ******'
   print '*****************************************************************'

if __name__=="__main__":
   simpleoverlay()
```

Boxfill Graphics Method Example:

```
#
# Example Boxfill (Gfb) module
#
#############################################################################
#                                                                           #
# Module:       exampleboxfill module                                        #
#                                                                           #
# Copyright:    2000, Regents of the University of California               #
#               This software may not be distributed to others without      #
#               permission of the author.                                    #
#                                                                           #
# Author:       Dean N. Williams, Lawrence Livermore National Laboratory    #
#               williams13@llnl.gov                                         #
#                                                                           #
# Description:  Example use of VCS's boxfill graphics method.               #
#                                                                           #
# Version:      1.0                                                         #
#                                                                           #
#############################################################################
#
#
#
#############################################################################
#                                                                           #
# Import: vcs  and cdms modules.                                            #
#                                                                           #
#############################################################################
def exampleboxfill():
   import vcs,cdms                      # import vcs and cdms

   f=cdms.openDataset('example.nc')     # open example file
   clt=f.variables['clt']               # get variable clt
   s=clt[0]                            # get clt data
   x=vcs.init()                         # construct vcs canvas
```

```
x.plot(s,'default','boxfill','quick',variable=clt)# plot slab the old way
x.geometry(450,337)                  # change the geometry

print x.listelements('boxfill')      # print boxfill Python list
x.show('boxfill')                    # show list of boxfill graphics methods
a=x.getboxfill('quick')              # get 'quick' boxfill graphics method
if x.isgraphicsmethod(a):            # check for graphics method
   print 'Yes, this is a graphics method'
   if x.isboxfill(a):                # check for boxfill
      print 'Yes, this is a isofill graphics method'
      a.list()                       # list its attributes

a.color_1=50                         # change color_1 index attribute
a.xticlabels('lon30','lon30')        # change xlabels attribute
a.xticlabels('','')                  # change remove xlables from plot
a.datawc(-45.0, 45.0, -90.0, 90.0)   # change region
a.datawc(1e20,1e20,1e20,1e20)        # change region back
a.xticlabels('*')                    # change attribute labels back

x.mode=0                             # turn atomatic update off
a.color_1=100                        # change color_1 attribute
a.color_2=200                        # change color_2 index value
a.xticlabels('lon30','lon30')        # change attribute
a.yticlabels('','')                  # change y-labels off attribute
a.datawc(-45.0, 45.0, -90.0, 90.0)   # change region
x.update()                           # view changes now

a.script('test','w')                 # save 'quick' boxfill as a Python script

x.mode=1                             # turn atomatic update mode back on
a.color_1=16                         # change color_1 attribute
a.color_2=239                        # change color_2 index value
a.level_1=20                         # change level_1
a.level_2=80                         # change level_2
a.datawc(1e20,1e20,1e20,1e20)        # change region back
a.yticlabels('*')                    # change y-labels attribute

x.scriptobject(a,'test', 'a')        # append 'quick' to the existing file
a.script('test.scr','w')          # save 'quick' as a VCS script file

x.show('template')                # show the list of templates
t=x.createtemplate('test','AMIPDUD')# create template 'test' from AMIPDUD

x.clear()                            # clear the VCS Canvas
x.boxfill(s,a,'default')             # plot using default template
```

```
    x.clear()                           # clear the VCS Canvas
    x.boxfill(a,'default',s)            # plot using default template, but,
        reverse the order
    x.clear()                           # clear the VCS Canvas
    x.boxfill(s,a,t)                    # plot using template 'test'
    x.clear()                           # clear the VCS Canvas
    x.boxfill(a,s,t)                   # plot using template 'test', but reverse
        the objects
    x.clear()                           # clear the VCS Canvas
    x.boxfill(t,a,s)                   # plot using template 'test', but reverse
        the objects
    x.clear()                           # clear the VCS Canvas

    x.plot(t,a,s)                       # plot using the new way
    x.clear()                           # clear the VCS Canvas
    x.plot(a,t,s)                       # plot using the new way
    x.clear()                           # clear the VCS Canvas
    x.plot(s,t,a)                       # plot using the new way
    x.clear()                           # clear the VCS Canvas
    x.plot('default',a,s)               # plot using the new way
    x.clear()                           # clear the VCS Canvas
    x.plot('default',s)                 # plot using the new way


    x.show('boxfill')                   # show boxfill list without test2
    a=x.createboxfill('test2','quick')  # create 'test2' from 'quick'
    a.color_1=50                        # change color level
    a.list()                            # list its attributes
    x.show('boxfill')                   # show boxfill list with test2
    x.removeobject(a)                   # remove test2 from boxfill list
    x.show('boxfill')                   # show boxfill list without test2

    print '*************************************************************'
    print '******                                                 ******'
    print '******   B O X F I L L   E X A M P L E   C O M P L E T E D   ******'
    print '******                                                 ******'
    print '*************************************************************'

if __name__=="__main__":
    exampleboxfill()
```

Continents Graphics Method Example:

```
#
# Example Continents (Gcon) module
#
#############################################################################
#                                                                           #
# Module:       examplecontinents module                                    #
#                                                                           #
# Copyright:    2000, Regents of the University of California               #
#               This software may not be distributed to others without      #
#               permission of the author.                                   #
#                                                                           #
# Author:       Dean N. Williams, Lawrence Livermore National Laboratory    #
#               williams13@llnl.gov                                         #
#                                                                           #
# Description:  Example of VCS's continents graphics method.                #
#                                                                           #
# Version:      1.0                                                         #
#                                                                           #
#############################################################################
#
#
#
#############################################################################
#                                                                           #
# Import: vcs modules.                                                      #
#                                                                           #
#############################################################################
def examplecontinents():
   import vcs                          # import vcs and cu

   x=vcs.init()                        # construct vcs canvas

   x.plot('default','continents','quick')# plot slab the old way
   x.geometry(450,337,100,0)           # change the geometry and location


   x.show('continents')                # show list of continents
   a=x.getcontinents('quick')          # get 'quick' continents
   if x.isgraphicsmethod(a):           # test object 'a' for graphics method
      print 'Yes, this is a graphics method'
      if x.iscontinents(a):            # test object 'a' if continents
         print 'Yes, this is an continents graphics method'
         a.list()                      # list the continents' attributes and
```

```
    values

a.script('test','w')                # save 'quick' continents as a Python
    script

a.xticlabels('','')                 # remove the x-axis
a.xticlabels('lon30','lon30')       # change the x-axis
a.xticlabels('*')                   # put the x-axis
a.datawc(-45.0, 45.0, -90.0, 90.0)  # change the region
a.datawc(1e20,1e20,1e20,1e20)       # put the region back

a.line=1                            # same as 'dash', change the line style
a.line=2                            # same as 'dot', change the line style
a.line=3                           # same as 'dash-dot', change the line style
a.line=0                            # same as 'solid', change the line style
a.line=4                          # same as 'long-dash', change the line style
a.linecolor=(77)                    # change the line color
a.linecolor=16                      # change the line color
a.linecolor=44                      # same as a.linecolor=(44)
a.linecolor=None                    # use the default line color, black
a.line=None                        # use default line style, solid black line

x.clear()                           # clear the VCS Canvas
x.continents(a,'default')          # plot continents using 'default' template

x.show('template')                  # show the list of templates
t=x.createtemplate('test')          # create template 'test' from 'default'
    template
if x.istemplate(t):                # test whether 't' is a template or not
   x.show('template')               # show the list of templates

x.clear()                           # clear the VCS Canvas
x.plot(t,a)                        # plot continents using template 't', and
    continents 'a'
x.clear()                           # clear the VCS Canvas
x.continents(a,t)                   # plot continents

###############################################################################
# Create line object 'l' from the default line                                #
###############################################################################
x.show('line')
l=x.createline('test')
if x.issecondaryobject(l):          # check to see if it is a secondary object
   print 'Yes, this is a secondary object.'
   if x.isline(l):                  # check to see if it is a line object
      print 'Yes, this is a line object.'
```

```
        l.list()                        # list the line's attributes and values

    ##########################################################################
    # Use the create line object 'l' from above and modify the line object  #
    ##########################################################################
    a.line=l                            # use the line object
    l.list()                          # list the line object attributes and values
    l.color = 44                        # change the line color
    l.type ='dash'                      # change the line type

    x.show('continents')                # show list of continents
    r=x.createcontinents('test2','quick')# create continents 'test2'
    x.show('continents')                # show list of continents
    x.removeobject(r)                   # remove continents 'test2'
    x.show('continents')                # show list of continents

    ##########################################################################
    # to see how x.update and x.mode work, see testoutline.py           #
    ##########################################################################
    #x.update()
    #x.mode=1
    #x.mode=0
    print '*********************************************************'
    print '******                                                 ******'
    print '******   C O N T I N E N T S   C O M P L E T E D    ******'
    print '******                                                 ******'
    print '*********************************************************'


if __name__=="__main__":
    examplecontinents()
```

Isofill Graphics Method Example:

```
#
# Example Isofill (Gfi) module
#
##############################################################################
#                                                                            #
# Module:       exampleisofill module                                         #
#                                                                            #
# Copyright:    2000, Regents of the University of California                 #
#               This software may not be distributed to others without       #
```

```
#               permission of the author.                                    #
#                                                                            #
# Author:       Dean N. Williams, Lawrence Livermore National Laboratory     #
#               williams13@llnl.gov                                          #
#                                                                            #
# Description:  Example use of VCS's isofill graphics method.                #
#                                                                            #
# Version:      1.0                                                          #
#                                                                            #
##############################################################################
#
#
#
##############################################################################
#                                                                            #
# Import: vcs and cdms modules.                                              #
#                                                                            #
##############################################################################
def exampleisofill():
   import vcs,cdms                      # import vcs and cdms

   f=cdms.openDataset('clt.nc')         # open example file
   clt=f.variables['clt']               # get variable clt
   s=clt[0]                             # get clt data
   x=vcs.init()                         # construct vcs canvas

   x.plot(s,'default','isofill','quick')# plot slab the old way
   x.geometry(450,337,100,0)            # change the geometry and location

   x.show('isofill')                    # show list of isofill graphics method
   a=x.getisofill('quick')              # get 'quick' isofill graphics method
   if x.isgraphicsmethod(a):            # test object 'a' for graphics method
      print 'Yes, this is a graphics method'
      if x.isisofill(a):                # test object 'a' if isofill
         print 'Yes, this is an isofill graphics method'
        a.list()                       # list the isofill's attributes and values

   a.script('test','w')                 # save 'quick' isofill as a Python script


   x.show('colormap')                # list all the colormaps
   x.setcolormap("AMIP")             # change the colormap from default to AMIP

   a.missing=241                   # change the missing background color to black

   a.xticlabels('lon30','lon30')        # change the x-axis
```

```
a.xticlabels('','')                  # remove the x-axis
a.xticlabels('*')                    # put the x-axis
a.datawc(-45.0, 45.0, -90.0, 90.0)   # change the region
a.datawc(1e20,1e20,1e20,1e20)        # put the region back

a.levels=([0,220],[230,240],[250,260])       # change the isofill levels
a.levels=([0,220,225,230,235,240],[230,240],[250,260]) # change the isofill
    levels
a.levels=([0,220,225,230,235,240],)  # change the isofill levels
a.levels=([0,220,225,230,235,240,245,250])   # change the isofill levels
a.levels=[0,220,225,230,235,240]     # change the isofill levels
a.levels=(0.0,220.0,225.0,230.0,235.0,240.0,250.0)    # change the isofill
    levels
a.levels=([1e20])                    # change back to default settings
a.levels=(0,220,225,230,235,240,250,260,270) # change the isofill levels

###############################################################################
# Below will produce an error. Later, if needed, I will add this         #
# functionality.                                                          #
#a.levels=('0','20','25','30')        # this will produce an error       #
###############################################################################

a.ext_1='y'                         # add the extended legend arrow to the left
a.ext_1='n'                          # remove the extended legend arrow to the
    left
a.ext_2='y'                          # add the extended legend arrow to the
    right
a.ext_2='n'                         # remove the extended legend arrow to the
    right
a.exts('y','y')                      # add the extended legend arrow to left
    and right
a.exts('n','n')                     # remove the extended legend arrow to left
    and right

a.fillareastyle='pattern'            # change the fill style to pattern
a.fillareastyle='hatch'              # change the fill style to hatch
a.fillareaindices=([1,3,5,6,9,20])   # set the hatch index patterns

a.fillareacolors=([22,33,44,55,66,77])      # set the fill area color indices
a.fillareacolors=None                        # use default color indices
a.fillareastyle='solid'                      # change the fill style back to
    solid

x.clear()                           # clear the VCS Canvas
x.isofill(s,a,'default')            # plot isofill using 'default' template
```

```
x.show('template')                    # show the list of templates
t=x.createtemplate('test')            # create template 'test' from 'default'
    template
if x.istemplate(t):                  # test whether 't' is a template or not
   x.show('template')                 # show the list of templates

x.show('fillarea')                    # show the list of fillarea secondary
    objects
f=x.getfillarea('def37')              # get fillarea 'def37'
if x.issecondaryobject(f):           # check to see if it is a secondary object
   print 'Yes, this is a secondary object.'
   if x.isfillarea(f):                # check to see if it is a fill area
      print 'Yes, this is a fill area object.'
     f.list()                      # list the fillarea's attributes and values

a.levels=(220,225,230,235,240,250,260,270,280,290,300,310)   # change the
    isofill levels
x.clear()                          # clear the VCS Canvas
x.plot(a,t,s)                     # plot array using isofill 'a' and template 't'
a.list()                           # list isofill's attributes
a.fillareaindices=(3,4,7,9,11)    # set the indices
a.fillareaindices=(f,f,f,f,f,f)      # set the indices using the fillarea
    object
a.fillareaindices=(f,2,4,7)          # reset the indices using the fillarea
    object
a.fillareaindices=(7,f,f,f,8)        # resett the indices using the fillare
    object

f.color=44                           # change the fillarea object's color
f.style='hatch'                     # change the fillarea object's fill style

x.scriptobject(a,'test')             # save 'quick' isofill as a Python script
x.scriptobject(f,'test')             # save 'def37' fill area as a Python
    script

x.show('isofill')                    # show list of isofill
r=x.createisofill('test2','quick')   # create isofill 'test2'
x.show('isofill')                    # show list of isofill
x.removeobject(r)                    # remove isofill 'test2'
x.show('isofill')                    # show list of isofill

###############################################################################
# to see how x.update and x.mode work, see testisofill.py                     #
###############################################################################
#x.update()
#x.mode=1
```

```
   #x.mode=0

   print '************************************************'
   print '******                                    ******'
   print '******   I S O F I L L  C O M P L E T E D  ******'
   print '******                                    ******'
   print '************************************************'

if __name__=="__main__":
   exampleisofill()
```

Isoline Graphics Method Example:

```
#
# Example Isoline (Gi) module
#
###############################################################################
#                                                                             #
# Module:       exampleisoline module                                          #
#                                                                             #
# Copyright:    2000, Regents of the University of California                  #
#               This software may not be distributed to others without        #
#               permission of the author.                                      #
#                                                                             #
# Author:       Dean N. Williams, Lawrence Livermore National Laboratory      #
#               williams13@llnl.gov                                           #
#                                                                             #
# Description:  Example use of VCS's isoline graphics method.                  #
#                                                                             #
# Version:      1.0                                                           #
#                                                                             #
###############################################################################
#
#
#
###############################################################################
#                                                                             #
# Import: vcs and cdms modules.                                               #
#                                                                             #
###############################################################################
def exampleisoline():
   import vcs,cdms                    # import vcs and cdms
```

```
f=cdms.openDataset('clt.nc')         # open example file
clt=f.variables['clt']               # get variable clt
s=clt[0]                             # get clt data
x=vcs.init()                         # construct vcs canvas

x.plot(s,'default','isoline','quick',variable=clt)# plot slab the old way
x.geometry(450,337,0,0)              # change geometry and location
x.geometry(900,675,0,0)              # change geometry and location

x.show('isoline')                    # show list of isoline
a=x.getisoline('quick')              # get isoline 'quick'
if x.isgraphicsmethod(a):            # test object 'a' for graphics method
   print 'Yes, this is a graphics method'
   if x.isisoline(a):                # test object 'a' for isoline
      print 'Yes, this is an isoline graphics method'
     a.list()                       # list the isoline's attributes and values

a.script('test','w')                # save 'quick' isoline as a Python script

a.xticlabels('lon30','lon30')        # change the x-axis
a.xticlabels('','')                  # remove the x-axis
a.xticlabels('*')                    # put the x-axis back
a.datawc(-45.0, 45.0, -90.0, 90.0)   # change the region
a.datawc(1e20,1e20,1e20,1e20)        # put the region back

###############################################################################
# Set the isoline level vales                                                 #
###############################################################################
a.level=([20,0.0],[30,0],[50,0],[60,0])      # change the isoline values
a.level=[[20,0.0],[30,0],[50,0]]     # change the isoline values
a.level=((20,0.0),(30,0),(50,0),(60,0),(70,0)) # change the isoline values
a.level=(25,35,45,55)                # change the isoline values
a.level=[(22,33,44,55,66)]           # change the isoline values
a.level=[(23,32,45,50,76),]          # change the isoline values
a.level=[0]                          # same as a.level=(0,)
a.level=[[0,1e20]]                  # same as a.level=((0,1e20),), use default
    settings

###############################################################################
# Turn on and off the isoline level labels                                    #
###############################################################################
a.label='y'                          # same as a.label=1
a.label='n'                          # same as a.label=0

###############################################################################
# Set the line style and line color                                           #
```

```
#############################################################################
a.level=((20,0.0),(30,0),(50,0),(60,0),(70,0)) # change the isoline values
a.line=[1,3,0,4]                    # same as a.line=(1,3,0,4)
a.line=(['dash','long-dash','solid'])# same as a.line=([2,4,0])
a.line=[2,4,1,3,2,0]
a.linecolors=([22,33,44,55,66,77])   # change the line color
a.linecolors=(16,19,33,44)           # change the line color
a.linecolors=None                    # use the default line color
a.line=None                          # use the default line style, which is
    solid

#############################################################################
# Set the text font and text color                                          #
#############################################################################
a.label='y'                         # same as a.label=1
a.text=(1,2,3,4,5,6,7,8,9)           # select fonts from 1 through 9
a.text=[9,8,7,6,5,4,3,2,1]
a.text=([1,3,5,6,9,2])
a.textcolors=([22,33,44,55,66,77])   # set the text color
a.textcolors=(16,19,33,44)
a.textcolors=None                    # use default text color, black
a.text=None                          # use default font, 1

#############################################################################
# Create template 'test' from the default template                          #
#############################################################################
x.show('template')                   # show the list of templates
t=x.createtemplate('test')           # create template 'test' from 'default'
    template
if x.istemplate(t):                 # test whether 't' is a template or not
   x.show('template')                # show the list of templates

#############################################################################
# Create line object 'l' from the default line                              #
#############################################################################
x.show('line')
l=x.createline('test')
if x.issecondaryobject(l):          # check to see if it is a secondary object
   print 'Yes, this is a secondary object.'
   if x.isline(l):                   # check to see if it is a line object
      print 'Yes, this is a line object.'
      l.list()                       # list the line's attributes and values

x.clear()                           # clear the VCS Canvas
x.isoline(s,a,t)                    # plot the array using the template and
    isoline object
```

```
x.clear()                              # clear the VCS Canvas
x.plot(t,a,s)                          # plot again using the new way

###############################################################################
# Use the create line object 'l' from above and modify the line object  #
###############################################################################
a.line=[1,3,0,4]                       # same as a.line=(1,3,0,4)
a.line=([2,4,0])                       # same as a.line=(['dash', 'long-dash',
    'solid'])
a.line=(l,4,l,0)                       # use the line object
a.line=(l,3,4,2,0)
l.list()                             # list the line object attributes and values
l.color = 44                           # change the line color
l.type ='dash'                         # change the line type

###############################################################################
# Create the three types of text objects                                      #
###############################################################################
tc = x.createtextcombined('testc','std', 'testc','7left')
if x.istextcombined(tc):
   print '*** textcombined listings ***'
   tc.list()

tt = x.createtexttable('testt', 'default')
if x.istexttable(tt):
   print '*** texttable listings ***'
   tt.list()

to = x.createtextorientation('testo')
if x.istextorientation(to):
   print '*** textorientation listings ***'
   to.list()

###############################################################################
# Use the text objects in the isoline plot                                    #
###############################################################################
a.label='y'                            # make sure that the labels are turn on
a.text=([1,3,5,6,9,2])                 # set the font
a.text=([tc,tt,to,6,9,2])              # use the created text objects and fonts

###############################################################################
# Change the text object values                                               #
###############################################################################
tc.font = 3                            # changing isoline level 20
tc.height=15
tc.angle=180
```

```
    tc.color=242
    tt.font=2                          # changing isoline level 30
    tt.spacing=20
    to.height=15                       # changing isoline level 50
    to.path='down'

    a.text=None                        # use default font, which is font 1
    a.line=None                        # use default line, which is solid

    x.show('isoline')                  # show list of isoline
    r=x.createisoline('test2','quick')  # create isoline 'test2'
    x.show('isoline')                  # show list of isoline
    x.removeobject(r)                  # remove isoline 'test2'
    x.show('isoline')                  # show list of isoline

    ########################################################################
    # to see how x.update and x.mode work, see testisoline.py            #
    ########################################################################
    #x.update()
    #x.mode=1
    #x.mode=0
    print '************************************************'
    print '******                                    ******'
    print '******   I S O L I N E   C O M P L E T E D   ******'
    print '******                                    ******'
    print '************************************************'

if __name__=="__main__":
    exampleisoline()
```

Outfill Graphics Method Example:

```
#
# Example Outfill (Gfo) module
#
################################################################################
#                                                                              #
# Module:       exampleoutfill module                                           #
#                                                                              #
# Copyright:    2000, Regents of the University of California                   #
#               This software may not be distributed to others without         #
#               permission of the author.                                      #
#                                                                              #
```

```
# Author:       Dean N. Williams, Lawrence Livermore National Laboratory   #
#               williams13@llnl.gov                                        #
#                                                                          #
# Description:  Example use of VCS's outfill graphics method.              #
#                                                                          #
# Version:      1.0                                                        #
#                                                                          #
############################################################################
#
#
#
############################################################################
#                                                                          #
# Import: vcs and cdms modules.                                             #
#                                                                          #
############################################################################
def exampleoutfill():
   import vcs,cdms                      # import vcs and cdms

   f=cdms.openDataset('example.nc')         # open example file
   clt=f.variables['clt']               # get variable clt
   s=clt[0]                             # get clt data
   x=vcs.init()                         # construct vcs canvas

   x.plot(s,'default','outfill','quick',variable=clt)# plot slab the old way
   x.geometry(450,337,100,0)             # change the geometry and location

   f2=cdms.openDataset('examlpe.nc') # open surface data
   sft=f2.variables['sft']               # get surface sft
   s=sft[...]                           # get sft data

   x.clear()                            # clear the VCS Canvas
   x.plot(s,'default','outfill','quick',variable=sft)# plot the surface data

   x.show('outfill')                    # show list of outfill
   a=x.getoutfill('quick')              # get 'quick' outfill
   if x.isgraphicsmethod(a):             # test object 'a' for graphics method
      print 'Yes, this is a graphics method'
      if x.isoutfill(a):                # test object 'a' if outfill
         print 'Yes, this is an outfill graphics method'
        a.list()                       # list the outfill's attributes and values

   a.script('test','w')                # save 'quick' outfill as a Python script


   a.xticlabels('lon30','lon30')        # change the x-axis
```

```
a.xticlabels('','')                  # remove the x-axis
a.xticlabels('*')                    # put the x-axis
a.datawc(-45.0, 45.0, -90.0, 90.0)   # change the region
a.datawc(1e20,1e20,1e20,1e20)        # put the region back

a.fillareastyle='hatch'              # change the fill style to hatch
a.fillareaindex=11                   # change the hatch index pattern
a.fillareacolor=(77)                 # change the hatch color
a.fillareacolor=16                   # chnage the hatch color
a.fillareacolor=44                   # same as a.fillareacolor=(44)
a.fillareacolor=None                 # use the default hatch color (black)
a.outfill=([0])                      # set the outfill value
a.outfill=([1])                      # set the outfill value
a.outfill=([0,1])                    # set the outfill value
a.outfill=([0])                      # set the outfill value


x.clear()                            # clear the VCS Canvas
x.outfill(s,a,'default')             # plot outfill using 'default' template

x.show('template')                   # show the list of templates
t=x.createtemplate('test')           # create template 'test' from 'default'
    template
if x.istemplate(t):                 # test whether 't' is a template or not
   x.show('template')                # show the list of templates

x.clear()                            # clear the VCS Canvas
x.plot(t,a,s)                       # plot outfill template 't', outfill 'a',
    and array 's'
x.clear()                            # clear the VCS Canvas
x.outfill(a,s,t)                     # plot using outfill 'a', array 's', and
    template 't'

x.show('fillarea')                   # show the list of fillarea secondary
    objects
f=x.getfillarea('def37')             # get fillarea 'def37'
if x.issecondaryobject(f):          # check to see if it is a secondary object
   print 'Yes, this is a secondary object.'
   if x.isfillarea(f):               # check to see if it is a fill area
      print 'Yes, this is a fill area object.'
     f.list()                      # list the fillarea's attributes and values

a.fillareastyle=f
f.color=44                           # change the fillarea object's color
f.style='hatch'                     # change the fillarea object's fill style
```

```
    x.show('outfill')                    # show list of outfill
    r=x.createoutfill('test2','quick')   # create outfill 'test2'
    x.show('outfill')                    # show list of outfill
    x.removeobject(r)                    # remove outfill 'test2'
    x.show('outfill')                    # show list of outfill


    ######################################################################
    # to see how x.update and x.mode work, see testoutfill.py        #
    ######################################################################
    #x.update()
    #x.mode=1
    #x.mode=0
    print '************************************************'
    print '******                                        ******'
    print '******   O U T F I L L   C O M P L E T E D    ******'
    print '******                                        ******'
    print '************************************************'

if __name__=="__main__":
    exampleoutfill()
```

Outline Graphics Method Example:

```
#
# Example Outline (Go) module
#
##############################################################################
#                                                                            #
# Module:       exampleoutline module                                         #
#                                                                            #
# Copyright:    2000, Regents of the University of California                 #
#               This software may not be distributed to others without       #
#               permission of the author.                                    #
#                                                                            #
# Author:       Dean N. Williams, Lawrence Livermore National Laboratory     #
#               williams13@llnl.gov                                          #
#                                                                            #
# Description:  Example use of VCS's outline graphics method.                #
#                                                                            #
# Version:      1.0                                                          #
#                                                                            #
##############################################################################
```

```
#
#
#
##############################################################################
#                                                                            #
# Import: vcs and cdms modules.                                               #
#                                                                            #
##############################################################################
def exampleoutline():
   import vcs,cdms                      # import vcs and cdms

   f=cdms.openDataset('example.nc')     # open example file
   clt=f.variables['clt']               # get variable clt
   s=clt[0]                             # get clt data
   x=vcs.init()                         # construct vcs canvas

   x.plot(s,'default','outline','quick',variable=clt)# plot slab the old way
   x.geometry(450,337,100,0)            # change the geometry and location


   f2=cdms.openDataset('example.nc') # open surface data
   sft=f2.variables['sft']               # get variable sft
   s=sft[...]                           # get sft data
   x.clear()                            # clear the VCS Canvas
   x.plot(s,'default','outline','quick',variable=sft)# plot the surface data

   x.show('outline')                    # show list of outline
   a=x.getoutline('quick')              # get 'quick' outline
   if x.isgraphicsmethod(a):            # test object 'a' for graphics method
      print 'Yes, this is a graphics method'
      if x.isoutline(a):                # test object 'a' if outline
         print 'Yes, this is an outline graphics method'
        a.list()                       # list the isoline's attributes and values

   a.script('test','w')                # save 'quick' outline as a Python script

   a.xticlabels('lon30','lon30')        # change the x-axis
   a.xticlabels('','')                  # remove the x-axis
   a.xticlabels('*')                    # put the x-axis
   a.datawc(-45.0, 45.0, -90.0, 90.0)   # change the region
   a.datawc(1e20,1e20,1e20,1e20)        # put the region back

   a.line=0                             # same as 'solid', change the line style
   a.line=1                             # same as 'dash', change the line style
   a.line=2                             # same as 'dot', change the line style
   a.line=3                            # same as 'dash-dot', change the line style
```

```
a.line=4                          # same as 'long-dash', change the line style
a.linecolor=(77)                    # change the line color
a.linecolor=16                      # change the line color
a.linecolor=44                      # same as a.linecolor=(44)
a.linecolor=None                    # use the default line color, black
a.line=None                       # use default line style, solid black line

a.outline=([0])                     # set the outline value
a.outline=([1])                     # set the outline value
a.outline=([0,1])                   # set the outline value
a.outline=([0])                     # set the outline value

x.clear()                           # clear the VCS Canvas
x.outline(s,a,'default')            # plot outline using 'default' template

x.show('template')                  # show the list of templates
t=x.createtemplate('test')          # create template 'test' from 'default'
    template
if x.istemplate(t):                # test whether 't' is a template or not
   x.show('template')               # show the list of templates

x.clear()                           # clear the VCS Canvas
x.plot(t,a,s)                     # plot outline template 't', outline 'a',
    and array 's'
x.clear()                           # clear the VCS Canvas
x.outline(a,s,t)                    # plot using outline 'a', array 's', and
    template 't'

##########################################################################
# Create line object 'l' from the default line                           #
##########################################################################
x.show('line')
l=x.createline('test')
if x.issecondaryobject(l):          # check to see if it is a secondary object
   print 'Yes, this is a secondary object.'
   if x.isline(l):                   # check to see if it is a line object
      print 'Yes, this is a line object.'
      l.list()                       # list the line's attributes and values

##########################################################################
# Use the create line object 'l' from above and modify the line object  #
##########################################################################
a.line=l                            # use the line object
l.list()                          # list the line object attributes and values
l.color = 44                        # change the line color
l.type ='dash'                      # change the line type
```

```
    x.show('outline')                    # show list of outline
    r=x.createoutline('test2','quick')   # create outline 'test2'
    x.show('outline')                    # show list of outline
    x.removeobject(r)                    # remove outline 'test2'
    x.show('outline')                    # show list of outline


    ##########################################################################
    # to see how x.update and x.mode work, see testoutline.py              #
    ##########################################################################
    #x.update()
    #x.mode=1
    #x.mode=0
    print '************************************************'
    print '******                                        ******'
    print '******   O U T F I L L   C O M P L E T E D    ******'
    print '******                                        ******'
    print '************************************************'

if __name__=="__main__":
    exampleoutfill()
```

Scatter Graphics Method Example:

```
#
# Example Scatter (GSp) module
#
###############################################################################
#                                                                             #
# Module:       examplescatter module                                          #
#                                                                             #
# Copyright:    2000, Regents of the University of California                  #
#               This software may not be distributed to others without        #
#               permission of the author.                                      #
#                                                                             #
# Author:       Dean N. Williams, Lawrence Livermore National Laboratory      #
#               williams13@llnl.gov                                           #
#                                                                             #
# Description:  Example use of VCS's scatter graphics method.                  #
#                                                                             #
# Version:      1.0                                                           #
```

```
#                                                                              #
################################################################################
#
#
#
################################################################################
#                                                                              #
# Import: vcs and cu modules.                                                   #
#                                                                              #
################################################################################
def examplescatter():
   import vcs,cu                        # import vcs, cu

   f=cu.open('clt2.nc')                 # open clt file
   u=f.getslab('u')                     # get slab u
   v=f.getslab('v')                     # get slab v

   x=vcs.init()                         # construct vcs canvas

   x.plot(u, v, 'default','scatter','quick')# plot slab the old way
   x.geometry(450,337,100,0)            # change the geometry and location

   a=x.getscatter('quick')              # get 'quick' scatter
   if x.isgraphicsmethod(a):            # test object 'a' for graphics method
      print 'Yes, this is a graphics method'
      if x.isscatter(a):                # test object 'a' if scatter
         print 'Yes, this is an scatter graphics method'
        a.list()                       # list the scatter's attributes and values

   a.script('test','w')                # save 'quick' scatter as a Python script


   a.xticlabels('','')                  # remove the x-axis
   a.xticlabels('*')                    # put the x-axis

   ###############################################################################
   # Change the scatter marker type                                              #
   ###############################################################################
   a.marker=1                           # same as a.marker='dot'
   a.marker=2                           # same as a.marker='plus'
   a.marker=3                           # same as a.marker='star'
   a.marker=4                           # same as a.marker='circle'
   a.marker=5                           # same as a.marker='cross'
   a.marker=6                           # same as a.marker='diamond'
   a.marker=7                           # same as a.marker='triangle_up'
   a.marker=8                           # same as a.marker='triangle_down'
```

```
a.marker=9                          # same as a.marker='triangle_left'
a.marker=10                         # same as a.marker='triangle_right'
a.marker=11                         # same as a.marker='square'
a.marker=12                         # same as a.marker='diamond_fill'
a.marker=13                         # same as a.marker='triangle_up_fill'
a.marker=14                         # same as a.marker='triangle_down_fill'
a.marker=15                         # same as a.marker='triangle_left_fill'
a.marker=16                         # same as a.marker='triangle_right_fill'
a.marker=17                         # same as a.marker='square_fill'

###############################################################################
# Change the scatter marker size                                              #
###############################################################################
a.markersize=5
a.markersize=55
a.markersize=100
a.markersize=300
a.markersize=15

###############################################################################
# Change the scatter marker color                                             #
###############################################################################
a.markercolor=(77)
a.markercolor=16
a.markercolor=44                    # same as a.markercolor=(44)

###############################################################################
# Change the scatter settings to default                                      #
###############################################################################
a.markercolor=None
a.markersize=None
a.marker=None

a.marker=1                          # same as a.marker='dot'

x.clear()                           # clear the VCS Canvas
x.scatter(u, v, a,'default')        # plot scatter using 'default' template

x.show('template')                  # show the list of templates
t=x.createtemplate('test')          # create template 'test' from 'default'
    template
if x.istemplate(t):                 # test whether 't' is a template or not
   x.show('template')               # show the list of templates

x.clear()                           # clear the VCS Canvas
x.plot(t,a,u,v)                     # plot scatter template 't', outline 'a',
```

```
    and arrays 'u':'v'
x.clear()                          # clear the VCS Canvas
x.scatter(a,u,v,t)                 # plot using outline 'a', array 'u':'v',
    and template 't'

x.show('marker')                  # show the list of marker secondary objects
m=x.getmarker('red')                # get marker 'red'
if x.issecondaryobject(m):         # check to see if it is a secondary object
   print 'Yes, this is a secondary object.'
   if x.ismarker(m):                # check to see if it is a fill area
      print 'Yes, this is a marker object.'
      m.list()                      # list the marker's attributes and values

##############################################################################
# Use the create marker object 'm' from above and modify the line object  #
##############################################################################
a.marker=m                         # use the marker object
m.list()                           # list the marker object attributes and
    values
m.color = 44                       # change the marker color
m.type ='square'                   # change the marker type
m.size=20                          # change the marker size

x.show('scatter')                  # show list of scatter
r=x.createscatter('test2','quick')   # create scatter 'test2'
x.show('scatter')                  # show list of scatter
x.removeobject(r)                  # remove scatter 'test2'
x.show('scatter')                  # show list of scatter

##########################################################################
# to see how x.update and x.mode work, see testoutline.py           #
##########################################################################
#x.update()
#x.mode=1
#x.mode=0
print '*************************************************'
print '******                                         ******'
print '******   S C A T T E R   C O M P L E T E D   ******'
print '******                                         ******'
print '*************************************************'

if __name__=="__main__":
   examplescatter()
```

Vector Graphics Method Example:

```
#
# Example Vector (Gv) module
#
#############################################################################
#                                                                           #
# Module:       examplevector module                                        #
#                                                                           #
# Copyright:    2000, Regents of the University of California               #
#               This software may not be distributed to others without      #
#               permission of the author.                                   #
#                                                                           #
# Author:       Dean N. Williams, Lawrence Livermore National Laboratory    #
#               williams13@llnl.gov                                         #
#                                                                           #
# Description:  Used to test VCS's vector graphics method.                  #
#                                                                           #
# Version:      1.0                                                         #
#                                                                           #
#############################################################################
#
#
#
#############################################################################
#                                                                           #
# Import: vcs and cu modules.                                               #
#                                                                           #
#############################################################################
def examplevector():
   import vcs,cu                        # import vcs and cu

   f=cu.open('clt.nc')                  # open clt file
   u=f.getslab('u',':',':',-10.,0.,0,10)# get slab u
   v=f.getslab('v',':',':',-10.,0.,0,10)# get slab v
   x=vcs.init()                         # construct vcs canvas

   x.plot(u, v, 'default','vector','quick')     # plot slab the old way
   x.geometry(450,337,100,0)            # change the geometry and location

   a=x.getvector('quick')               # get 'quick' vector
   if x.isgraphicsmethod(a):            # test object 'a' for graphics method
      print 'Yes, this is a graphics method'
      if x.isvector(a):                 # test object 'a' if vector
         print 'Yes, this is an vector graphics method'
```

```
      a.list()                        # list the vector's attributes and values

a.script('test','w')                  # save 'quick' vector as a Python script


a.xticlabels('','')                   # remove the x-axis
a.xticlabels('*')                     # put the x-axis

###############################################################################
# Change the vector scale                                                     #
###############################################################################
a.scale=2.0
a.scale=5.0
a.scale=1.5
a.scale=0.0
a.scale=-1.5
a.scale=-2.0
a.scale=-5.0

###############################################################################
# Change the vector typeiiiiiii                                               #
###############################################################################
a.type=0                              # same as a.type = 'arrows'
a.type=1                              # same as a.type = 'barbs'
a.type=2                              # same as a.type = 'solidarrows'

###############################################################################
# Change the vector reference                                                 #
###############################################################################
a.reference=10.
a.reference=100.
a.reference=4.
a.reference=5.

###############################################################################
# Change the vector alignment                                                 #
###############################################################################
a.alignment='head'                    # same as a.alignment=0
a.alignment='center'                  # same as a.alignment=1
a.alignment='tail'                    # same as a.alignment=2

###############################################################################
# Change the vector line                                                      #
###############################################################################
a.line=0                              # same as 'solid'
a.line=1                              # same as 'dash'
```

```
a.line=2                              # same as 'dot'
a.line=3                              # same as 'dash-dot'
a.line=4                              # same as 'long-dash'
a.line=None                           # use default line

###############################################################################
# Change the vector line color                                                #
###############################################################################
a.linecolor=(77)
a.linecolor=16
a.linecolor=44  # same as a.color=(44)
a.linecolor=None

x.clear()                             # clear the VCS Canvas
x.vector(u, v, a,'default')           # plot vector using 'default' template

x.show('template')                    # show the list of templates
t=x.createtemplate('test')            # create template 'test' from 'default'
    template
if x.istemplate(t):                   # test whether 't' is a template or not
   x.show('template')                 # show the list of templates

x.clear()                             # clear the VCS Canvas
x.plot(t,a,u,v)                       # plot vector template 't', outline 'a',
    and arrays 'u':'v'
x.clear()                             # clear the VCS Canvas
x.vector(a,u,v,t)                     # plot using outline 'a', array 'u':'v',
    and template 't'

x.show('line')                        # show the list of line secondary objects
l=x.getline('red')                    # get line 'red'
if x.issecondaryobject(l):            # check to see if it is a secondary object
   print 'Yes, this is a secondary object.'
   if x.isline(l):                    # check to see if it is a line
      print 'Yes, this is a line object.'
      l.list()                        # list the line's attributes and values

###############################################################################
# Use the create line object 'm' from above and modify the line object    #
###############################################################################
a.line=l                              # use the line object
l.list()                           # list the line object attributes and values
l.color = 44                          # change the line color
l.style ='square'                     # change the line type
l.width=4                             # change the line size
```

```
    x.show('vector')                    # show list of vector
    r=x.createvector('test2','quick')    # create vector 'test2'
    x.show('vector')                    # show list of vector
    x.removeobject(r)                   # remove vector 'test2'
    x.show('vector')                    # show list of vector

    ###############################################################################
    # to see how x.update and x.mode work, see testoutline.py                     #
    ###############################################################################
    #x.update()
    #x.mode=1
    #x.mode=0
    print '************************************************'
    print '******                                        ******'
    print '******   V E C T O R   C O M P L E T E D   ******'
    print '******                                        ******'
    print '************************************************'

if __name__=="__main__":
    examplevector()
```

XvsY Graphics Method Example:

```
#
# Example XvsY (GXY) module
#
###############################################################################
#                                                                             #
# Module:       examplexvsy module                                            #
#                                                                             #
# Copyright:    2000, Regents of the University of California                  #
#               This software may not be distributed to others without        #
#               permission of the author.                                     #
#                                                                             #
# Author:       Dean N. Williams, Lawrence Livermore National Laboratory      #
#               williams13@llnl.gov                                           #
#                                                                             #
# Description:  Example use of VCS's XvsY graphics method.                    #
#                                                                             #
# Version:      1.0                                                           #
#                                                                             #
###############################################################################
```

```
#
#
#
##############################################################################
#                                                                            #
# Import: vcs and cu modules.                                                #
#                                                                            #
##############################################################################
def examplexvsy():
   import vcs,cu                        # import vcs and cu

   f=cu.open('clt.nc')                  # open clt file
   u=f.getslab('u')                     # get slab u
   v=f.getslab('v')                     # get slab v
   x=vcs.init()                         # construct vcs canvas

   x.plot(u, v, 'default','xvsy','quick')       # plot slabs the old way
   x.geometry(450,337,100,0)            # change the geometry and location

   a=x.getxvsy('quick')                 # get 'quick' xvsy
   if x.isgraphicsmethod(a):            # test object 'a' for graphics method
      print 'Yes, this is a graphics method'
      if x.isxvsy(a):                   # test object 'a' if xvsy
         print 'Yes, this is an xvsy graphics method'
         a.list()                       # list the xvsy's attributes and values

   a.script('test','w')                 # save 'quick' xvsy as a Python script

   a.xticlabels('','')                  # remove the x-axis
   a.xticlabels('*')                    # put the x-axis

   ############################################################################
   # Change the xvsy line                                                     #
   ############################################################################
   a.line=0                             # same as 'solid'
   a.line=1                             # same as 'dash'
   a.line=2                             # same as 'dot'
   a.line=3                             # same as 'dash-dot'
   a.line=4                             # same as 'long-dash'

   ############################################################################
   # Change the xvsy line color                                               #
   ############################################################################
   a.linecolor=(77)
   a.linecolor=16
   a.linecolor=44                       # same as a.color=(44)
```

```
a.linecolor=None

###############################################################################
# Change the xvsy marker                                                      #
###############################################################################
a.marker=1                          # Same as a.marker='dot'
a.marker=2                          # Same as a.marker='plus'
a.marker=3                          # Same as a.marker='star'
a.marker=4                          # Same as a.marker='circle'
a.marker=5                          # Same as a.marker='cross'
a.marker=6                          # Same as a.marker='diamond'
a.marker=7                          # Same as a.marker='triangle_up'
a.marker=8                          # Same as a.marker='triangle_down'
a.marker=9                          # Same as a.marker='triangle_left'
a.marker=10                         # Same as a.marker='triangle_right'
a.marker=11                         # Same as a.marker='square'
a.marker=12                         # Same as a.marker='diamond_fill'
a.marker=13                         # Same as a.marker='triangle_up_fill'
a.marker=14                         # Same as a.marker='triangle_down_fill'
a.marker=15                         # Same as a.marker='triangle_left_fill'
a.marker=16                         # Same as a.marker='triangle_right_fill'
a.marker=17                         # Same as a.marker='square_fill'
a.marker=None                       # Draw no markers

###############################################################################
# Change the xvsy marker color                                                #
###############################################################################
a.marker='dot'
a.markercolor=16
a.markercolor=44                    # same as a.markercolor=(44)
a.markercolor=None

###############################################################################
# Change the xvsy marker size                                                 #
###############################################################################
a.markersize=5
a.markersize=55
a.markersize=10
a.markersize=100
a.markersize=300
a.markersize=None

x.clear()                           # clear the VCS Canvas
x.xvsy(u, v, a,'default')           # plot xvsy using 'default' template

x.show('template')                  # show the list of templates
```

```
t=x.createtemplate('test')          # create template 'test' from 'default'
    template
if x.istemplate(t):                 # test whether 't' is a template or not
   x.show('template')               # show the list of templates

x.clear()                           # clear the VCS Canvas
x.plot(t,a,u,v)                     # plot xvsy template 't', outline 'a', and
    arrays 'u':'v'
x.clear()                           # clear the VCS Canvas
x.xvsy(a,u,v,t)                     # plot using outline 'a', array 'u':'v',
    and template 't'

x.show('line')                      # show the list of line secondary objects
l=x.getline('red')                  # get line 'red'
if x.issecondaryobject(l):          # check to see if it is a secondary object
   print 'Yes, this is a secondary object.'
   if x.isline(l):                  # check to see if it is a line
      print 'Yes, this is a line object.'
      l.list()                      # list the line's attributes and values

##############################################################################
# Use the create line object 'm' from above and modify the line object       #
##############################################################################
a.line=l                            # use the line object
l.list()                            # list the line object attributes and values
l.color = 44                        # change the line color
l.style ='dot'                      # change the line type
l.width=4                           # change the line size

x.show('marker')                    # show the list of marker secondary objects
m=x.getmarker('red')                # get marker 'red'
if x.issecondaryobject(m):          # check to see if it is a secondary object
   print 'Yes, this is a secondary object.'
   if x.ismarker(m):                # check to see if it is a fill area
      print 'Yes, this is a marker object.'
      m.list()                      # list the marker's attributes and values

##############################################################################
# Use the create marker object 'm' from above and modify the line object     #
##############################################################################
a.marker=m                          # use the marker object
m.list()                            # list the marker object attributes and
    values
m.color = 44                        # change the marker color
m.type ='square'                    # change the marker type
m.size=20                           # change the marker size
```

```
    x.show('xvsy')                      # show list of xvsy
    r=x.createxvsy('test2','quick')     # create xvsy 'test2'
    x.show('xvsy')                      # show list of xvsy
    x.removeobject(r)                   # remove xvsy 'test2'
    x.show('xvsy')                      # show list of xvsy

    ##########################################################################
    # to see how x.update and x.mode work, see testoutline.py              #
    ##########################################################################
    #x.update()
    #x.mode=1
    #x.mode=0
    print '*********************************************'
    print '******                                  ******'
    print '******   X v s Y   C O M P L E T E D    ******'
    print '******                                  ******'
    print '*********************************************'

if __name__=="__main__":
    examplexvsy()
```

Xyvsy Graphics Method Example:

```
#
# Example Xyvsy (GXy) module
#
#############################################################################
#                                                                           #
# Module:       examplexyvsy module                                          #
#                                                                           #
# Copyright:    2000, Regents of the University of California               #
#               This software may not be distributed to others without      #
#               permission of the author.                                   #
#                                                                           #
# Author:       Dean N. Williams, Lawrence Livermore National Laboratory    #
#               williams13@llnl.gov                                         #
#                                                                           #
# Description:  Example use of VCS's Xyvsy graphics method.                 #
#                                                                           #
# Version:      1.0                                                         #
#                                                                           #
```

```
###############################################################################
#
#
#
###############################################################################
#                                                                             #
# Import: vcs and cu modules.                                                  #
#                                                                             #
###############################################################################
def examplexyvsy():
   import vcs,cu                        # import vcs and cu

   f=cu.open('clt.nc')                  # open clt file
   u=f.getslab('u')                     # get slab u
   x=vcs.init()                         # construct vcs canvas

   x.plot(u, 'default','xyvsy','quick') # plot slab the old way
   x.geometry(450,337,100,0)            # change the geometry and location

   a=x.getxyvsy('quick')                # get 'quick' xyvsy
   if x.isgraphicsmethod(a):            # test object 'a' for graphics method
      print 'Yes, this is a graphics method'
      if x.isxyvsy(a):                  # test object 'a' if xyvsy
         print 'Yes, this is an xyvsy graphics method'
         a.list()                       # list the xyvsy's attributes and values

   a.script('test','w')                 # save 'quick' xyvsy as a Python script

   a.xticlabels('','')                  # remove the x-axis
   a.xticlabels('*')                    # put the x-axis

   ############################################################################
   # Change the xyvsy line                                                    #
   ############################################################################
   a.line=0                             # same as 'solid'
   a.line=1                             # same as 'dash'
   a.line=2                             # same as 'dot'
   a.line=3                             # same as 'dash-dot'
   a.line=4                             # same as 'long-dash'

   ############################################################################
   # Change the xyvsy line color                                              #
   ############################################################################
   a.linecolor=(77)
   a.linecolor=16
   a.linecolor=44                       # same as a.color=(44)
```

```
a.linecolor=None

###############################################################################
# Change the xyvsy marker                                                     #
###############################################################################
a.marker=1                          # Same as a.marker='dot'
a.marker=2                          # Same as a.marker='plus'
a.marker=3                          # Same as a.marker='star'
a.marker=4                          # Same as a.marker='circle'
a.marker=5                          # Same as a.marker='cross'
a.marker=6                          # Same as a.marker='diamond'
a.marker=7                          # Same as a.marker='triangle_up'
a.marker=8                          # Same as a.marker='triangle_down'
a.marker=9                          # Same as a.marker='triangle_left'
a.marker=10                         # Same as a.marker='triangle_right'
a.marker=11                         # Same as a.marker='square'
a.marker=12                         # Same as a.marker='diamond_fill'
a.marker=13                         # Same as a.marker='triangle_up_fill'
a.marker=14                         # Same as a.marker='triangle_down_fill'
a.marker=15                         # Same as a.marker='triangle_left_fill'
a.marker=16                         # Same as a.marker='triangle_right_fill'
a.marker=17                         # Same as a.marker='square_fill'
a.marker=None                       # Draw no markers

###############################################################################
# Change the xyvsy marker color                                               #
###############################################################################
a.marker='dot'
a.markercolor=16
a.markercolor=44                    # same as a.markercolor=(44)
a.markercolor=None

###############################################################################
# Change the xyvsy marker size                                                #
###############################################################################
a.markersize=5
a.markersize=55
a.markersize=10
a.markersize=100
a.markersize=300
a.markersize=None

x.clear()                           # clear the VCS Canvas
x.xyvsy(u, a,'default')              # plot xyvsy using 'default' template

x.show('template')                  # show the list of templates
```

```
t=x.createtemplate('test')          # create template 'test' from 'default'
    template
if x.istemplate(t):                 # test whether 't' is a template or not
   x.show('template')               # show the list of templates

x.clear()                           # clear the VCS Canvas
x.plot(t,a,u)                       # plot xyvsy template 't', outline 'a',
    and arrays 'u':'v'
x.clear()                           # clear the VCS Canvas
x.xyvsy(a,u,t)                      # plot using outline 'a', array 'u':'v',
    and template 't'

x.show('line')                      # show the list of line secondary objects
l=x.getline('red')                  # get line 'red'
if x.issecondaryobject(l):          # check to see if it is a secondary object
   print 'Yes, this is a secondary object.'
   if x.isline(l):                  # check to see if it is a line
      print 'Yes, this is a line object.'
      l.list()                      # list the line's attributes and values

###############################################################################
# Use the create line object 'm' from above and modify the line object    #
###############################################################################
a.line=l                            # use the line object
l.list()                            # list the line object attributes and values
l.color = 44                        # change the line color
l.style ='dot'                      # change the line type
l.width=4                           # change the line size

x.show('marker')                    # show the list of marker secondary objects
m=x.getmarker('red')                # get marker 'red'
if x.issecondaryobject(m):          # check to see if it is a secondary object
   print 'Yes, this is a secondary object.'
   if x.ismarker(m):                # check to see if it is a fill area
      print 'Yes, this is a marker object.'
      m.list()                      # list the marker's attributes and values

###############################################################################
# Use the create marker object 'm' from above and modify the line object  #
###############################################################################
a.marker=m                          # use the marker object
m.list()                            # list the marker object attributes and
    values
m.color = 44                        # change the marker color
m.type ='square'                    # change the marker type
m.size=20                           # change the marker size
```

```
    x.show('xyvsy')                     # show list of xyvsy
    r=x.createxyvsy('test2','quick')    # create xyvsy 'test2'
    x.show('xyvsy')                     # show list of xyvsy
    x.removeobject(r)                   # remove xyvsy 'test2'
    x.show('xyvsy')                     # show list of xyvsy

    ######################################################################
    # to see how x.update and x.mode work, see testoutline.py          #
    ######################################################################
    #x.update()
    #x.mode=1
    #x.mode=0
    print '*********************************************'
    print '******                                 ******'
    print '******   X y v s y   C O M P L E T E D   ******'
    print '******                                 ******'
    print '*********************************************'

if __name__=="__main__":
    examplexyvsy()
```

Yxvsx Graphics Method Example:

```
#
# Example Yxvsx (GYx) module
#
##############################################################################
#                                                                            #
# Module:       exampleyxvsx module                                           #
#                                                                            #
# Copyright:    2000, Regents of the University of California                 #
#               This software may not be distributed to others without       #
#               permission of the author.                                    #
#                                                                            #
# Author:       Dean N. Williams, Lawrence Livermore National Laboratory     #
#               williams13@llnl.gov                                          #
#                                                                            #
# Description:  Example use of VCS's Yxvsx graphics method.                  #
#                                                                            #
# Version:      1.0                                                          #
#                                                                            #
```

```
###############################################################################
#
#
#
###############################################################################
#                                                                             #
# Import: vcs and cu modules.                                                  #
#                                                                             #
###############################################################################
def exampleyxvsx():
   import vcs,cu                        # import vcs and cu

   f=cu.open('clt.nc')                  # open clt file
   u=f.getslab('u')                     # get slab u
   x=vcs.init()                         # construct vcs canvas

   x.plot(u, 'default','yxvsx','quick') # plot slab the old way
   x.geometry(450,337,100,0)            # change the geometry and location

   a=x.getyxvsx('quick')                # get 'quick' yxvsx
   if x.isgraphicsmethod(a):            # test object 'a' for graphics method
      print 'Yes, this is a graphics method'
      if x.isyxvsx(a):                  # test object 'a' if yxvsx
         print 'Yes, this is an yxvsx graphics method'
         a.list()                       # list the yxvsx's attributes and values

   a.script('test','w')                 # save 'quick' yxvsx as a Python script

   a.xticlabels('','')                  # remove the x-axis
   a.xticlabels('*')                    # put the x-axis

   ############################################################################
   # Change the yxvsx line                                                    #
   ############################################################################
   a.line=0                             # same as 'solid'
   a.line=1                             # same as 'dash'
   a.line=2                             # same as 'dot'
   a.line=3                             # same as 'dash-dot'
   a.line=4                             # same as 'long-dash'

   ############################################################################
   # Change the yxvsx line color                                              #
   ############################################################################
   a.linecolor=(77)
   a.linecolor=16
   a.linecolor=44                       # same as a.color=(44)
```

```
a.linecolor=None

###############################################################################
# Change the yxvsx marker                                                     #
###############################################################################
a.marker=1                          # Same as a.marker='dot'
a.marker=2                          # Same as a.marker='plus'
a.marker=3                          # Same as a.marker='star'
a.marker=4                          # Same as a.marker='circle'
a.marker=5                          # Same as a.marker='cross'
a.marker=6                          # Same as a.marker='diamond'
a.marker=7                          # Same as a.marker='triangle_up'
a.marker=8                          # Same as a.marker='triangle_down'
a.marker=9                          # Same as a.marker='triangle_left'
a.marker=10                         # Same as a.marker='triangle_right'
a.marker=11                         # Same as a.marker='square'
a.marker=12                         # Same as a.marker='diamond_fill'
a.marker=13                         # Same as a.marker='triangle_up_fill'
a.marker=14                         # Same as a.marker='triangle_down_fill'
a.marker=15                         # Same as a.marker='triangle_left_fill'
a.marker=16                         # Same as a.marker='triangle_right_fill'
a.marker=17                         # Same as a.marker='square_fill'
a.marker=None                       # Draw no markers

###############################################################################
# Change the yxvsx marker color                                               #
###############################################################################
a.marker='dot'
a.markercolor=16
a.markercolor=44                    # same as a.markercolor=(44)
a.markercolor=None

###############################################################################
# Change the yxvsx marker size                                                #
###############################################################################
a.markersize=5
a.markersize=55
a.markersize=10
a.markersize=100
a.markersize=300
a.markersize=None

x.clear()                           # clear the VCS Canvas
x.yxvsx(u, a,'default')              # plot yxvsx using 'default' template

x.show('template')                  # show the list of templates
```

```
t=x.createtemplate('test')          # create template 'test' from 'default'
    template
if x.istemplate(t):                 # test whether 't' is a template or not
   x.show('template')               # show the list of templates

x.clear()                           # clear the VCS Canvas
x.plot(t,a,u)                       # plot yxvsx template 't', outline 'a',
    and arrays 'u':'v'
x.clear()                           # clear the VCS Canvas
x.yxvsx(a,u,t)                      # plot using outline 'a', array 'u':'v',
    and template 't'

x.show('line')                      # show the list of line secondary objects
l=x.getline('red')                  # get line 'red'
if x.issecondaryobject(l):          # check to see if it is a secondary object
   print 'Yes, this is a secondary object.'
   if x.isline(l):                  # check to see if it is a line
      print 'Yes, this is a line object.'
      l.list()                      # list the line's attributes and values

###############################################################################
# Use the create line object 'm' from above and modify the line object    #
###############################################################################
a.line=l                            # use the line object
l.list()                            # list the line object attributes and values
l.color = 44                        # change the line color
l.style ='dot'                      # change the line type
l.width=4                           # change the line size

x.show('marker')                    # show the list of marker secondary objects
m=x.getmarker('red')                # get marker 'red'
if x.issecondaryobject(m):          # check to see if it is a secondary object
   print 'Yes, this is a secondary object.'
   if x.ismarker(m):                # check to see if it is a fill area
      print 'Yes, this is a marker object.'
      m.list()                      # list the marker's attributes and values

###############################################################################
# Use the create marker object 'm' from above and modify the line object  #
###############################################################################
a.marker=m                          # use the marker object
m.list()                            # list the marker object attributes and
    values
m.color = 44                        # change the marker color
m.type ='square'                    # change the marker type
m.size=20                           # change the marker size
```

```
    x.show('yxvsx')                     # show list of yxvsx
    r=x.createyxvsx('test2','quick')    # create yxvsx 'test2'
    x.show('yxvsx')                     # show list of yxvsx
    x.removeobject(r)                   # remove yxvsx 'test2'
    x.show('yxvsx')                     # show list of yxvsx

    ######################################################################
    # to see how x.update and x.mode work, see testoutline.py       #
    ######################################################################
    #x.update()
    #x.mode=1
    #x.mode=0
    print '*********************************************'
    print '******                                 ******'
    print '******   Y x v s x   C O M P L E T E D  ******'
    print '******                                 ******'
    print '*********************************************'

if __name__=="__main__":
    exampleyxvsx()
```

Colormap Example:

```
#
# Example Colormap (Cp) module
#
###############################################################################
#                                                                             #
# Module:       examplecolormap module                                        #
#                                                                             #
# Copyright:    2000, Regents of the University of California                  #
#               This software may not be distributed to others without        #
#               permission of the author.                                     #
#                                                                             #
# Author:       Dean N. Williams, Lawrence Livermore National Laboratory      #
#               williams13@llnl.gov                                           #
#                                                                             #
# Description:  Example use of VCS's color map.                               #
#                                                                             #
# Version:      1.0                                                           #
#                                                                             #
###############################################################################
```

```
#
#
#
##############################################################################
#                                                                            #
# Import: vcs and cu modules.                                                #
#                                                                            #
##############################################################################
def examplecolormap():
   import vcs,cu                        # import vcs and cu

   f=cu.open('example.nc')              # open clt file
   s=f.getslab('clt')                   # get slab clt
   x=vcs.init()                         # construct vcs canvas

   x.plot(s,'default','isofill','quick')# plot slab the old way
   x.geometry(450,337,0,0)              # change the geometry

   x.setcolormap("AMIP")                # change the colormap

   ###############################################################
   # Change the color map cell 31.                               #
   ###############################################################
   x.setcolorcell(31,0,0,0)
   x.setcolorcell(31,100,0,0)
   x.setcolorcell(31,0,100,0)
   x.setcolorcell(31,0,0,100)
   x.setcolorcell(31,100,100,100)
   x.setcolorcell(31,70,70,70)

   ###################################################################
   # NOTE:                                                           #
   #       The colormapgui command will only work if the display     #
   #       depth is set to 8-bit pseudo color. The next release      #
   #       will allow for the colormap's graphical user interface    #
   #       (GUI) and the animation's GUI to work in 16-, 24-, or     #
   #       32-bit True color visual classes.                         #
   ###################################################################
   x.colormapgui()


   print '*************************************************'
   print '******                                     ******'
   print '******   C O L O R M A P   C O M P L E T E D   ******'
   print '******                                     ******'
   print '*************************************************'
```

```
if __name__=="__main__":
   examplecolormap()
```

Hardcopy Example:

```
#
# Example Hardcopy module
#
##############################################################################
#                                                                            #
# Module:       examplehardcopy module                                       #
#                                                                            #
# Copyright:    2000, Regents of the University of California                 #
#               This software may not be distributed to others without       #
#               permission of the author.                                    #
#                                                                            #
# Author:       Dean N. Williams, Lawrence Livermore National Laboratory     #
#               williams13@llnl.gov                                          #
#                                                                            #
# Description:  Exmaple use of VCS's hard copy.                              #
#                                                                            #
# Version:      1.0                                                          #
#                                                                            #
##############################################################################
#
#
#
##############################################################################
#                                                                            #
# Import: vcs and cu modules.                                                #
#                                                                            #
##############################################################################
def examplehardcopy():
   import vcs,cu                        # import vcs and cu

   f=cu.open('clt.nc')                  # open clt file
   s=f.getslab('clt')                   # get slab clt
   x=vcs.init()                         # construct vcs canvas

   x.plot(s,'default','isofill','quick', bg=1) # plot slab the old way, but in
       background
```

```
x.gif('test')                          # generate gif file
x.postscript('test')                   # generate postscript file
x.cgm('test')                          # generate cgm file
x.raster('test')                     # generate a raster file

x.setcolormap("AMIP")                  # change the colormap
x.clear()                              # clear all segments

t1=x.createtemplate('test1','AMIP_1of2') # create template 'test' from AMIPDUD
t2=x.createtemplate('test2','AMIP_2of2') # create template 'test' from AMIPDUD
isof=x.createisofill('test')           # create isofill graphics method from
    default
isol=x.createisoline('test')           # create isoline graphics method from
    default

#########################################################################
# draw isofill plot on top, then an isoline plot on the bottom       #
#########################################################################
x.plot(s,t1,isof, bg=1)                # generate isofill plot in background
x.plot(s,t2, isol, bg=1)               # generate isoline plot in background
x.gif('test2.gif')                     # generate gif file
x.postscript('test2.ps')               # generate postscript file
x.cgm('test2.cgm')                     # generate cgm file

x.clear()                              # clear all segments

#########################################################################
# draw isofill plot, then overlay an isoline plot                      #
#########################################################################
x.plot(s, isof, bg=1)                 # generate isofill plot
x.plot(s, isol, 'default', bg=1)      # generate isoline plot
x.gif('test3.gif')                     # generate gif file

x.clear()                              # clear all segments
x.setcolormap("default")               # change colormap to default
x.plot(s,bg=1)                         # plot boxfill
x.postscript('test4.ps')               # create a postscript file
x.('test4.ps')                # generate a gif file from the postscript file

print '*****************************************************'
print '******                                          ******'
print '******   H A R D C O P Y   C O M P L E T E D    ******'
print '******                                          ******'
print '*****************************************************'
```

```
if __name__=="__main__":
   examplehardcopy()
```

Picture Template Example:

```
#
# Example Picture Template (P) module
#
##############################################################################
#                                                                            #
# Module:       exampletemplate module                                       #
#                                                                            #
# Copyright:    2000, Regents of the University of California                 #
#               This software may not be distributed to others without       #
#               permission of the author.                                    #
#                                                                            #
# Author:       Dean N. Williams, Lawrence Livermore National Laboratory     #
#               williams13@llnl.gov                                          #
#                                                                            #
# Description:  Example use of VCS's template object.                        #
#                                                                            #
# Version:      1.0                                                          #
#                                                                            #
##############################################################################
#
#
#
##############################################################################
#                                                                            #
# Import: vcs and cu modules.                                                #
#                                                                            #
##############################################################################
def exampletemplate():
   import vcs,cu                        # import vcs and cu

   f=cu.open('clt.nc')                  # open clt file
   s=f.getslab('clt')                   # get slab clt
   x=vcs.init()                         # construct vcs canvas

   x.show('template')                   # show the list of templates
   t=x.createtemplate('test','AMIP')    # create template 'test' from AMIP
   x.show('template')                   # show the list of templates
```

```
    t.script('test','w')                # save test template as a Python script

    g=x.createisofill('test')           # create isofill 'test' from 'default'
        isofill

    x.plot(g,s,t)                       # make isofill plot
#   x.isofill(g,s,t)                    # make isofill plot


    ###############################################################################
    # Show the many different ways to show the template members (attributes)    #
    # and their values.                                                         #
    ###############################################################################
    t.list()                # list the templates members
    t.list('text')          # list only text members, same as t.list('Pt')
    t.list('format')        # list only format members, same as t.list('Pf')
    t.list('xtickmarks')    # list only xtickmarks members, same as t.list('Pxt')
    t.list('ytickmarks')    # list only ytickmarks members, same as t.list('Pyt')
    t.list('xlabels')       # list only xlabels members, same as t.list('Pxl')
    t.list('ylabels')       # list only ylabels members, same as t.list('Pyl')
    t.list('boxeslines')    # list only boxeslines members, same as t.list('Pbl')
    t.list('legend')        # list only legend member, same as t.list('Pls')
    t.list('data')          # list only data member, same as t.list('Pds')
    t.list('file')          # list only file member and its values
    t.file.list()           # list only file member and its values
    t.list('mean')          # list only mean member and its values
    t.mean.list()           # list only mean member and its values

    ###############################################################################
    # The screen x and y positions on the screen are normalized between 0 and 1 #
    # for both the x and y axis.                                                #
    ###############################################################################
    t.mean.priority = 0     # remove the "Mean" text from the plot
    t.mean.priority = 1     # re-display the "Mean" text on the plot
    t.mean.x=0.5            # move the "Mean" text to x-axis center
    t.mean.y=0.5            # move the "Mean" text to y-axis center

    ###############################################################################
    # Position the data in front of the "Mean" text, then move the "Mean" text  #
    # in front of the data.                                                     #
    ###############################################################################
    t.data.priority = 2
    t.data.priority=3

    ###############################################################################
    # Change the font representation for the "Mean" text.                       #
```

```
    # You can set the text by using text objects or text names.                 #
    ###############################################################################
    tt = x.createtexttable('test')
    to = x.createtextorientation('test')
    t.mean.texttable = tt                # set texttable by using texttable object
    t.mean.textorientation = 'test'       # set textorientation by using
        textorientation name
    t.mean.list()                       # show the mean member and their new values
    tt.font=2                             # change the font
    to.height=40                          # change the height

    ###############################################################################
    # Change the legend space.                                                    #
    ###############################################################################
    t.legend.list()                       # list the legend members
    x.mode=0                              # turn the automatic update off
    t.legend.x1=0.85
    t.legend.y1=0.90
    t.legend.x2=0.95
    t.legend.y2=0.16
    x.update()

    print '*****************************************************'
    print '******                                          ******'
    print '******   T E M P L A T E   C O M P L E T E D    ******'
    print '******                                          ******'
    print '*****************************************************'

if __name__=="__main__":
    exampletemplate()
```

Simple Animation Example:

```
#
# Simple animation module
#
###############################################################################
#                                                                             #
# Module:       simpleanimation module                                        #
#                                                                             #
# Copyright:    2000, Regents of the University of California                  #
#               This software may not be distributed to others without        #
```

```
#               permission of the author.                                    #
#                                                                            #
# Author:       Dean N. Williams, Lawrence Livermore National Laboratory     #
#               williams13@llnl.gov                                          #
#                                                                            #
# Description:  Simple animation example.                                    #
#               NOTE:                                                        #
#                     The animation module will only work if the display     #
#                     depth is set to 8-bit pseudo color. The next release   #
#                     will allow for the animation's graphical user          #
#                     interface (GUI) and the colormap's GUI to display      #
#                     in 16-, 24-, or 32-bit True color visual classess.     #
#                                                                            #
# Version:      1.0                                                          #
#                                                                            #
##############################################################################
#
#
#
##############################################################################
#                                                                            #
# Import: vcs  and cdms modules.                                             #
#                                                                            #
##############################################################################
def simpleanimation():
   import vcs,cdms                      # import vcs and cdms

   ########################################################################
   # See the CDMS document on how to ingest data. Also see the CU and     #
   # the Numeric documents on alternative ways to import data into VCS    #
   ########################################################################

   f=cdms.openDataset('example.nc')     # open example file
   clt=f.variables['clt']               # get variable clt
   s=clt[0]                            # get clt data

   ########################################################################
   # At the moment, VCS can only animate one variable at a time.          #
   ########################################################################
   x=vcs.init()                 # initialize vcs
   x.plot(s,variable=clt)       # plot data using default values
   x.animate()

   print '*************************************************************'
   print '******                                                 ******'
   print '******   S I M P L E   A N I M A T I O N   C O M P L E T E D   ******'
```

```
    print '******                                                              ******'
    print '************************************************************************'

if __name__=="__main__":
    simpleanimation()
```

CHAPTER 9

Quick Reference Guides

Commands and Their Usage

Command	Usage
Initialization	
init	a = vcs.init()
Help	
help	vcs.help()
objecthelp	a.objecthelp(b)
Canvas	
mode	a.mode=1
update	a.update()
open	a.open()
close	a.close()
flushcanvas	a.flushcanvas()
refreshcanvas	a.refreshcanvas()
portrait	a.portrait()
landscape	a.landscape()
page	a.page()
geometry	a.geometry(450, 337,100, 100)
Printing and Saving Graphics	
printer	a.printer('printer_name')
gif	a.gif('gif_file_name')

Command	Usage
postscript	a.postscript('postscript_file_name')
cgm	a.cgm('cgm_file_name')
raster	a.raster('raster_file_name')
pstogif	a.pstogif('postscript_file_name')
Plot and Clear Commands	
plot	a.plot(array)
boxfill	a.boxfill(array)
continents	a.continents(array)
isofill	a.isofill(array)
isoline	a.isoline(array)
outfill	a.outfill(array)
outline	a.outline(array)
scatter	a.scatter(array)
vector	a.vector(array)
xvsy	a.xvsy(array)
xyvsy	a.xyvsy(array)
yxvsx	a.yxvsx(array)
clear	a.clear()
Query Functions	
graphicsmethodname	a.graphicsmethodname('box')
getcontinentstype	a.getcontinentstype()
isgraphicsmethod	a.isgraphicsmethod()
isboxfill	a.isboxfill()
iscontinents	a.iscontinents()
isisofill	a.isisofill()
isisoline	a.isisoline()
isoutfill	a.isoutfill()
isoutline	a.isoutline()
isvector	a.isvector()

Command	Usage
isxvsy	a.isxvsy()
isxyvsy	a.isxyvsy()
isyxvsx	a.isyxvsx()
istemplate	a.istemplate()
issecondaryobject	a.issecondaryobject()
isfillarea	a.isfillarea()
isline	a.isline()
ismarker	a.ismarker()
istextcombined	a.istextcombined()
istextorientation	a.istextorientation()
istexttable	a.istexttable()
List Available Templates, Graphics Methods and Secondary Objects	
listelements	a.listelements()
show	a.show('boxfill')
list	box.list() con.list isol.list() isof.list() outl.list() outf.list() sct.list() vc.list() xy.list() tpl.list() fa.list() tt.list() to.list() tc.list()

Command	Usage
Graphics Methods Objects	
Boxfill	
createboxfill	box=a.createboxfill('new')
getboxfill	box=a.getboxfill()
Continents	
createcontinents	con=a.createcontinents('new')
getcontinents	con=a.getcontinents()
Isofill	
createisofill	iso=a.createisofill('new')
getisofill	iso=a.getisofill()
Isoline	
createisoline	iso=a.createisoline('new')
getisoline	iso=a.getisoline()
Outfill	
createoutfill	out=a.createoutfill('new')
getoutfill	out=a.getoutfill()
Outline	
createoutline	out=a.createoutline('new')
getoutline	out=a.getoutline()
Scatter	
createscatter	scr=a.createscatter('new')
getscatter	scr=a.getscatter()
Vector	
createvector	vc=a.createvector('new')
getvector	vc=a.getvector()
XvsY	
createxvsy	xy=a.createxvsy('new')

Command	Usage
getxvsy	xy=a.getxvsy()
Xyvsy	
createxyvsy	xy=a.createxyvsy(‘new’)
getxyvsy	xy=a.getxyvsy()
Yxvsx	
createyxvsx	yx=a.createyxvsx(‘new’)
getyxvsx	yx=a.createyxvsx()
Picture Template	
createtemplate	tpl=a.createtemplate(‘new’)
gettemplate	tpl=gettemplate()
Secondary Objects	
Colormap Commands	
setcolormap	a.setcolormap("colormap_name")
setcolorcell	.setcolorcell(16,100,100,100)
colormapgui	a.colormapgui()
Fill Area	
createfillarea	fa=a.createfillarea(’new’)
getfillarea	fa=a.getfillarea()
Line	
createline	ln=a.createline(‘new’)
getline	ln=a.getline()
marker	
createmarker	mk=a.createmarker(‘new’)
getmarker	mk=a.getlmarker()
Text-Combined	
createtextcombined	tc=a.createtextcombined(‘new’)
gettextcombined	tc=a.gettextcombined()

Command	Usage
Text-Orientation	
createtextorientation	to.createtextorientation('new')
gettextorientation	to=a.gettextorientation()
Text-Table	
createtexttable	tt=a.createtexttable('new')
gettexttable	tt=a.gettexttable()
Remove Objects	
removeobject	a.removeobject(a)
Set Default Picture Template and Graphics Methods	
set	a.set('isofill','some_isofill_name')
Set Continents Type	
setcontinentstype	a.setcontinentstype(3)
Animation	
animate	a.animate()
Flush	
flush	a.flush()
Grid	
grid	a.grid(12,24, -90,90, -180,180)
resetgrid	a.resetgrid()

VCS Cheat Sheet

init	graphicsmethodname	createisoline	fillareaobject
help	getcontinentstype	getisoline	createline
objecthelp	isgraphicsmethod	isolineobject	getline
mode	isboxfill	createoutfill	lineobject
update	iscontinents	getoutfill	marker
open	isisofill	outfillobject	createmarker
close	isisoline	createoutline	getmarker
flushcanvas	isoutfill	getoutline	markerobject
refreshcanvas	isoutline	outlineobject	createtextcombined
portrait	isvector	createscatter	gettextcombined
landscape	isxvsy	getscatter	textcombinedobject
page	isxyvsy	scatterobject	createtextorientation
geometry	isyxvsx	createvector	gettextorientation
printer	istemplate	getvector	textorientationobject
gif	issecondaryobject	vectorobject	createtexttable
postscript	isfillarea	createxvsy	gettexttable
cgm	isline	getxvsy	texttableobject
raster	ismarker	xvsyobject	removeobject
pstogif	istextcombined	createxyvsy	set
plot	istextorientation	getxyvsy	setcontinentstype
boxfill	istexttable	xyvsyobject	animate
continents	listelements	createyxvsx	flush
isofill	show	getyxvsx	grid
isoline	createboxfill	yxvsxobject	resetgrid
outfill	getboxfill	createtemplate	list
outline	boxfillobject	gettemplate	
scatter	createcontinents	templateobject	
vector	getcontinents	setcolormap	
xvsy	continentsobject	setcolorcell	
xyvsy	createisofill	colormapgui	
yxvsx	getisofill	createfillarea	
clear	isofillobject	getfillarea	

CHAPTER 10

Fonts, Lines, Markers, and Patterns

Pictures usually contain some sort of textual information. Therefore, it is necessary to specify the type of font needed for a particular plot. This section will textually show the 9 different fonts.

Lines can be used to draw from one point to another or to outline information. This section will visually show the 5 different line types.

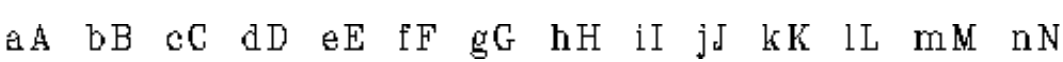
Markers can be used to identify points of data. This section will visually show the 17 different marker types.

Patterns can be used to fill in areas between data. This section will visually show the 20 different patterns available.

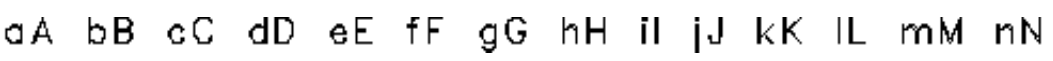
Fonts

Font 1:

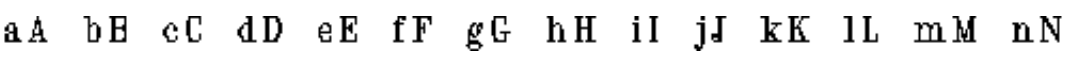
Font 2:



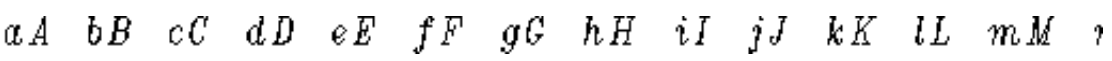
Font 3:



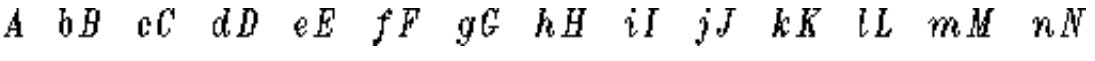
Font 4:



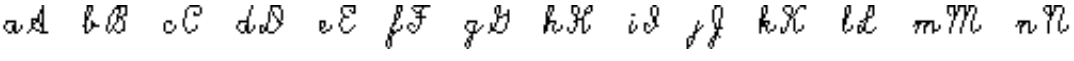
Font 5:



Font 6:

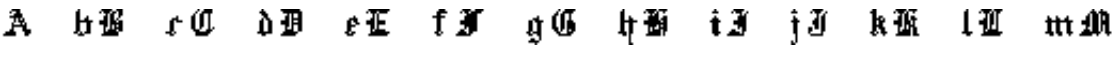


Font 7:



Font 8:

Font 9:



Line Styles

Line 1 (Solid):

Line 2 (Dashed):

Line 3 (Dotted):

Line 4 (Dashed-Dotted):

Line 5 (Long-Dashed):

Markers

Marker 1 (Dot):

Marker 2 (Plus):

Marker 3 (Star):

Marker 4 (Circle):

Marker 5 (Cross):

Marker 6 (Diamond):

Marker 7 (Triangle-Up):

Marker 8 (Triangle-Down):

Marker 9 (Triangle-Left):

Marker 10 (Triangle-Right):

Marker 11 (Square):

Marker 12 (Diamond-Fill):

Marker 13 (Triangle-Up-Fill):

Marker 14 (Triangle-Down-Fill):

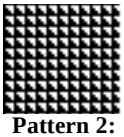
Marker 15 (Triangle-Left-Fill):

Marker 16 (Triangle-Right-Fill):

Marker 17 (Square-Fill):

Patterns

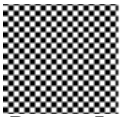
Pattern 1:



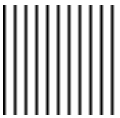
Pattern 2:

Pattern 3:

Pattern 4:

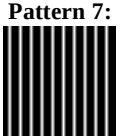


Pattern 5:

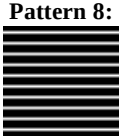


Pattern 6:

Pattern 7:



Pattern 8:

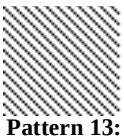


Pattern 9:

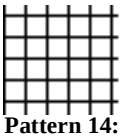
Pattern 10:

Pattern 11:

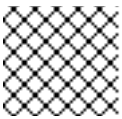
Pattern 12:



Pattern 13:



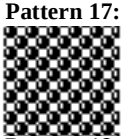
Pattern 14:



Pattern 15:

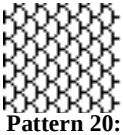
Pattern 16:

Pattern 17:

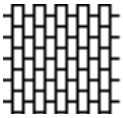


Pattern 18:

Pattern 19:



Pattern 20:



CHAPTER 11

The VCS Module

The VCS module is designed to provide access to the VCS graphical interface.

The VCS Graphics Module

VCS is CDAT's interface to a library for two-dimensional graphics that has been particularly customized for climate data but which in fact can be used by many scientific applications. VCS has a rich set of facilities that allow you to easily make sophisticated presentation graphics.

VCS as a module has the principal function init() which returns a Canvas object. The rest of the commands are methods on the Canvas object. Up to 8 canvases can be active at one time.

Command	Description	Examples
init()	Initializes and creates a VCS object, reads in the initial.attribute file and set up the VCS default values. Note: The number of VCS Canvases is limited to eight.	import vcs x=vcs.**init**()
help()	Gives insight to other VCS functions by providing a description and at least one example.	import vcs vcs.**help**() vcs.**help**('init')
objecthelp()	Print out information on VCS objects. See example on its use.	import vcs x=vcs.init() ln=a.createline('red') x.**objecthelp**(ln)
plot()	Plot an array(s) of data given a template and graphics method.	import vcs f=cu.open('clt.nc') s=f.getslab('clt') x=vcs.init() x.**plot**(s)
clear()	This routine will remove all plots on the VCS Canvas.	import vcs f=cu.open('clt.nc') s=f.getslab('clt') x=vcs.init() x.plot(s) x.**clear**()
mode()	Updating of the graphical displays on the VCS Canvas can be deferred until a later time.	import vcs x=vcs.init() x.**mode**=0

update()	If a series of commands are given to VCS and the Canvas Mode is set to manual, then use this function to update the VCS Canvas manually.	import vcs f=cu.open('clt.nc') s=f.getslab('clt') x=vcs.init() x.mode=0 x.plot(s'default','boxfill','quick') box=x.getboxfill('quick') box.color_1=100 box.xticlabels('lon30','lon30') box.xticlabels('','') box.datawc(1e20,1e20,1e20,1e20) box.datawc(-45.0, 45.0, -90.0, 90.0) x.**update**()
open()	Open VCS Canvas object. This routine really just manages the VCS canvas. It will popup the VCS Canvas for viewing. It can be used to display only the VCS Canvas.	import vcs x=vcs.init() x.**open**()
close()	Close the VCS Canvas. It will remove the VCS Canvas object from the screen, but not deallocate it.	import vcs x=vcs.init() x.plot(s) a.**close**()
portrait()	Change the VCS Canvas orientation to Portrait.	import vcs f=cu.open('clt.nc') s=f.getslab('clt') x=vcs.init() x.plot(s) x.**portrait**()

landscape()	Change the VCS Canvas orientation to Landscape.	import vcs f=cu.open('clt.nc') s=f.getslab('clt') x=vcs.init() x.plot(s) x.**landscape**()
page()	Change the VCS Canvas orientation to either 'portrait' or 'landscape'.	import vcs f=cu.open('clt.nc') s=f.getslab('clt') x=vcs.init() x.plot(s) x.**page**()
geometry()	The geometry command is used to set the size and position of the VCS canvas.	import vcs f=cu.open('clt.nc') s=f.getslab('clt') x=vcs.init() x.plot(s) x.**geometry**(450, 337,100, 100)
printer()	his function creates a temporary cgm file and then sends it to the specified printer.	import vcs f=cu.open('clt.nc') s=f.getslab('clt') x=vcs.init() x.plot(s) x.**printer**('printer_name')
gif()	Generate a gif file	import vcs f=cu.open('clt.nc') s=f.getslab('clt') x=vcs.init() x.plot(s) x.**gif**('example')

postscript()	Generate a postscript file.	import vcs f=cu.open('clt.nc') s=f.getslab('clt') x=vcs.init() x.plot(s) x.**postscript**('example')
cgm()	Generate a cgm file.	import vcs f=cu.open('clt.nc') s=f.getslab('clt') x=vcs.init() x.plot(s) x.**cgm**('example')
raster()	Generate a raster file.	import vcs f=cu.open('clt.nc') s=f.getslab('clt') x=vcs.init() x.plot(s) x.**raster**('example')
pstogif()	Converts a postscript file to a gif file.	import vcs x=vcs.init() x.**pstogif**('filename.ps')
boxfill()	Generate a boxfill plot given the data, boxfill graphics method, and template.	import vcs f=cu.open('clt.nc') s=f.getslab('clt') x=vcs.init() x.**boxfill**(s)
continents()	Generate a continents plot given the continents graphics method, and template.	import vcs f=cu.open('clt.nc') s=f.getslab('clt') x=vcs.init() x.**continents**(s)

isofill()	Generate a isofill plot given the data, isofill graphics method, and template.	import vcs f=cu.open('clt.nc') s=f.getslab('clt') x=vcs.init() x.**isofill**(s)
isoline()	Generate a isoline plot given the data, isoline graphics method, and template.	import vcs f=cu.open('clt.nc') s=f.getslab('clt') x=vcs.init() x.**isoline**(s)
outfill()	Generate a outfill plot given the data, outfill graphics method, and template.	import vcs f=cu.open('clt.nc') s=f.getslab('clt') x=vcs.init() x.**outfill**(s)
outline()	Generate a outline plot given the data, outline graphics method, and template.	import vcs f=cu.open('clt.nc') s=f.getslab('clt') x=vcs.init() x.**outline**(s)
scatter()	Generate a scatter plot given the data, scatter graphics method, and template.	import vcs f=cu.open('clt.nc') s=f.getslab('clt') x=vcs.init() x.**scatter**(s)

vector()	Generate a vector plot given the data, vector graphics method, andtemplate.	import vcs f=cu.open('clt.nc') s=f.getslab('clt') x=vcs.init() x.**vector**(s)
xvsy()	Generate a XvsY plot given the data, XvsY graphics method, and template.	import vcs f=cu.open('clt.nc') s=f.getslab('clt') x=vcs.init() x.**xvsy**(s)
xyvsy()	Generate a Xyvsy plot given the data, Xyvsy graphics method, and template.	import vcs f=cu.open('clt.nc') s=f.getslab('clt') x=vcs.init() x.**xyvsy**(s)
yxvsx()	Generate a Yxvsx plot given the data, Yxvsx graphics method, and template.	import vcs f=cu.open('clt.nc') s=f.getslab('clt') x=vcs.init() x.**yxvsx**(s)
graphicsmethod-name()	Returns the grapics method's type string: boxfill, isofill, isoline, out-fill, outline, continents, scatter, vector, xvsy, xyvsy, or yxvsx.	import vcs x=vcs.init() box=x.getboxfill() x.**graphicsmethodname**(box)
getcontinentstype()	Return continents type from VCS. Remember the value can only be between 0 and 11.	import vcs x=vcs.init() cont_type = x.**getcontinentstype**()

isgraphicsmethod()	Indicates if the entered argument is one of the following graphics methods: boxfill, isofill, isoline, outfill, outline, continents, scatter, vector, xvsy, xyvsy, yxvsx.	import vcs x=vcs.init() box=x.getboxfill('quick') x.**isgraphicsmethod**(box)
isboxfill()	Check to see if an object is a VCS primary boxfill graphics method object.	import vcs x=vcs.init() box=x.getboxfill("quick") x.**isboxfill**(box)
iscontinents()	Check to see if an object is a VCS primary continents graphics method.	import vcs x=vcs.init() con=x.getcontinents("quick") x.**iscontinents**(con)
isisofill()	Check to see if an object is a VCS primary isofill graphics method.	import vcs x=vcs.init() iso=x.getisofill("quick") x.**isisofill**(iso)
isisoline()	Check to see if an object is a VCS primary isoline graphics method.	import vcs x=vcs.init() iso=x.getisoline("quick" x.**isisoline**(iso)
isoutfill()	Check to see if this object is a VCS primary outfill graphics method.	import vcs x=vcs.init() out=x.getoutfill("quick") x.**isoutfill**(out)
isoutline()	Check to see if an object is a VCS primary outline graphics method.	import vcs x=vcs.init() out=x.getoutline("quick") x.**isoutline**(out)

isvector()	Check to see if an object is a VCS primary vector graphics method.	import vcs x=vcs.init() vec=x.getvector("quick") x.**isvector**(vec)
isxvsy()	Check to see if an object is a VCS primary xvsy graphics method.	import vcs x=vcs.init() xy=x.getxvsy("quick") x.**isxvsy**(xy)
isxyvsy()	Check to see if an object is a VCS primary Xyvsy graphics method.	import vcs x=vcs.init() xyy=x.getxyvsy("quick") x.**isxyvsy**(xyy)
isyxvsx()	Check to see if an object is a VCS primary yxvsx graphics method.	import vcs x=vcs.init() yxx=x.getyxvsx("quick") x.**isyxvsx**(yxx)
istemplate()	Indicates if the entered argument a template.	import vcs x=vcs.init() templt=x.gettemplate('quick') x.**istemplate**(templt)
issecondaryobject()	Query to see if an object is a secondary object.	import vcs x=vcs.init() line=x.getline() x.**issecondaryobject**(line)
isfillarea()	Check to see if an object is a VCS secondary fillarea.	import vcs x=vcs.init() fa=x.getfillarea("def37") x.**isfillarea**(fa)
isline()	Check to see if this object is a VCS secondary line.	import vcs x=vcs.init() ln=x.getline() x.**isline**(ln)

ismarker()	Check to see if this object is a VCS secondary marker.	import vcs x=vcs.init() ln=x.getmarker() x.**ismarker**(ln)
istextcombined()	Check to see if an object is a VCS secondary text combined.	import vcs x=vcs.init() tc=x.gettextcombined("std", "7left") x.**istextcombined**(tc)
istextorientation()	Check to see if an object is a VCS secondary text orientation.	import vcs x=vcs.init() to=x.gettextorientation("7left") x.**istextorientation**(to)
istexttable()	Check to see if an object is a VCS secondary text table.	import vcs x=vcs.init() tt=x.gettexttable("std") x.**istexttable**(tt)
listelements()	Returns a Python list of all the VCS class objects.	import vcs x=vcs.init() x.**listelements**()
show()	Show the list of VCS primary and secondary class objects.	import vcs x=vcs.init() x.**show**('boxfill') x.**show**('isofill') x.**show**('template') x.**show**('line')
createboxfill()	Create a new boxfill graphics method given the name and the existingboxfill graphics method to copy the attributes from.	import vcs x=vcs.init() box=x.**createboxfill**('example','quick')
getboxfill()	his function will create a boxfill class object from an existing VCS boxfill graphics method.	import vcs x=vcs.init() box2=x.**getboxfill**('quick')

createcontinents()	Create a new continents graphics method given the the name and the existing continents graphics method to copy the attributes from.	import vcs x=vcs.init() con=x.**createcontinents**('example','quick')
getcontinents()	This function will create a continents class object from an existing VCS continents graphics method.	import vcs x=vcs.init() con2=x.**getcontinents**('quick')
createisofill()	Create a new isofill graphics method given the the name and the existing isofill graphics method to copy the attributes from.	import vcs x=vcs.init() iso=x.**createisofill**('example','quick')
getisofill()	This function will create a isofill class object from an existing VCS isofill graphics method.	import vcs x=vcs.init() iso2=x.**getisofill**('quick')
createisoline()	Create a new isoline graphics method given the the name and the existing isoline graphics method to copy the attributes from.	import vcs x=vcs.init() iso=x.**createisoline**('example','quick')
getisoline()	This function will create a isoline class object from an existing VCS isoline graphics method.	import vcs x=vcs.init() iso2=x.**getisoline**('quick')
createoutfill()	Create a new outfill graphics method given the the name and the existing outfill graphics method to copy the attributes from.	import vcs x=vcs.init() out=x.**createoutfill**('example','quick')
getoutfill()	This function will create a outfill class object from an existing VCS outfill graphics method.	import vcs x=vcs.init() out2=x.**getoutfill**('quick')
createoutline()	Create a new outline graphics method given the the name and the existing outline graphics method to copy the attributes from.	import vcs x=vcs.init() out=x.**createoutline**('example','quick')

getoutline()	This function will create a outline class object from an existing VCS outline graphics method.	import vcs x=vcs.init() out2=x.**getoutline**('quick')
createscatter()	Create a new scatter graphics method given the the name and the existing mscatter graphics method to copy the attributes from.	import vcs x=vcs.init() sct=x.**createscatter**('example','quick')
getscatter()	This function will create a scatter class object from an existing VCS scatter graphics method.	import vcs x=vcs.init() sct2=x.**getscatter**('quick')
createvector()	Create a new vector graphics method given the the name and the existing vector graphics method to copy the attributes from.	import vcs x=vcs.init() vec=x.**createvector**('example','quick')
getvector()	This function will create a vector class object from an existing VCS vector graphics method.	import vcs x=vcs.init() vec2=x.**getvector**('quick')
createxvsy()	Create a new XvsY graphics method given the the name and the existing XvsY graphics method to copy the attributes from.	import vcs x=vcs.init() xy=x.**createxvsy**('example','quick')
getxvsy()	This function will create a XvsY class object from an existing VCS XvsY graphics method.	import vcs x=vcs.init() xy2=x.**getxvsy**('quick')
createxyvsy()	Create a new Xyvsy graphics method given the the name and the existing Xyvsy graphics method to copy the attributes from.	import vcs x=vcs.init() xyy=x.**createxyvsy**('example','quick')
getxyvsy()	This function will create a Xyvsy class object from an existing VCS Xyvsy graphics method.	import vcs x=vcs.init() xyy2=x.getxyvsy('quick')

createyxvsx()	Create a new Yxvsx graphics method given the the name and the existing Yxvsx graphics method to copy the attributes from.	import vcs x=vcs.init() yxx=x.**createyxvsx**('example','quick')
getyxvsx()	This function will create a Yxvsx class object from an existing VCS Yxvsx graphics method.	import vcs x=vcs.init() yxx2=x.**getyxvsx**('quick')
createtemplate()	Create a new template given the the name and the existing template to copy the attributes from.	import vcs x=vcs.init() con=x.**createtemplate**('example','quick')
gettemplate()	his function will create a template class object from an existing VCS template.	import vcs x=vcs.init() **templt2=x.gettemplate**('quick')
setcolormap()	This routine will change the VCS color map.	import vcs x=vcs.init() x.**setcolormap**("AMIP")
setcolorcell()	Set a individual color cell in the active colormap. If default is the active colormap, then return an error string.	import vcs x=vcs.init() x.setcolormap("AMIP") a.**setcolorcell**(11,0,0,0) x.**setcolorcell**(21,100,0,0)
colormapgui()	Run the VCS colormap interface. The colormapgui command is used to bring up the VCS colormap interface. The interface is used to select, create, change, or remove colormaps. Note: The colormapgui GUI will only work for 8-bit 'Pseudo Color'.	import vcs x=vcs.init() x.**colormapgui**()

createfillarea()	Create a new fillarea secondary method given the the name and the existing fillarea secondary method to copy the attributes from.	import vcs x=vcs.init() fa=x.**createfillarea**('example','black')
getfillarea()	This function will create a fillarea class object from an existing VCS fillarea secondary method.	import vcs x=vcs.init() fa2=x.**getfillarea**('quick')
createline()	Create a new line secondary method given the the name and the existing line secondary method to copy the attributes from.	import vcs x=vcs.init() ln=x.**createline**('example','black')
getline()	This function will create a line class object from an existing VCS line secondary method.	import vcs x=vcs.init() ln2=x.**getline**('quick')
createmarker()	Create a new marker secondary method given the the name and the existing marker secondary method to copy the attributes from.	import vcs x=vcs.init() mrk=x.**createmarker**('example','black')
getmarker()	This function will create a marker class object from an existing VCS marker secondary method.	import vcs x=vcs.init() mrk2=x.**getmarker**('quick')
createtextcombined()	Create a new textcombined secondary method given the the names and the existing texttable and textorientation secondary methods to copy the attributes from. I	import vcs x=vcs.init() tc=x.**createtextcombined**('example1','std','example1','7left')
gettextcombined()	This function will create a textcombined class object from an existing VCS texttable secondary method and an existing VCS textorientation secondary method.	import vcs x=vcs.init() tc2=x.**gettextcombined**('std','7left')

createtextorienta-tion()	Create a new textorientation secondary method given the the name and he existing textorientation secondary method to copy the attributes from.	import vcs x=vcs.init() to=x.**createtextorientation**('example1')
gettextorientation()	This function will create a textorientation class object from an existing VCS textorientation secondary method.	import vcs x=vcs.init() to2=x.**gettextorientation**('quick')
createtexttable()	Create a new texttable secondary method given the the name and the existing texttable secondary method to copy the attributes from.	import vcs x=vcs.init() tt=x.**createtexttable**('example','black')
gettexttable()	This function will create a texttable class object from an existing VCS texttable secondary method.	import vcs x=vcs.init() tt2=x.**gettexttable**('quick')
removeobject()	This function allows the user to remove these objects from the appropriate class list. Note, To remove the object completely from Python, remember to use the "del" function. Also note, The user is not allowed to remove a "default" class object.	import vcs x=vcs.init() line=x.getline('red') x.**removeobject**(line) iso=x.createisoline('example') x.**removeobject**(iso)
setcontinentstype()	One has the option of using continental maps that are predefined or that are user-defined. Predefined continental maps are either internal to VCS or are specified by external files. (The continents type number ranges from 0 to 11.)	import vcs x=vcs.init() f=cu.open('clt.nc') s=f.getslab('clt') x.**setcontinentstype**(3) x.plot(array,'default','isofill','quick')

set()	Set the default VCS primary class objects: template and graphics methods.	import vcs x=vcs.init() f=cu.open('clt.nc') s=f.getslab('clt') x.**set**('isofill','quick') x.plot(s)
animate()	Animate the contents of the VCS Canvas. Currently, only one display can be shown in the VCS Canvas for the animation to work. This function pops up the animation GUI. Note: The animation GUI will only work for 8-bit 'Pseudo Color'.	import vcs x=vcs.init() f=cu.open('clt.nc') s=f.getslab('clt') x.plot(s) x.**animate**()
flush()	The flush command executes all buffered X events in the que.	import vcs f=cu.open('clt.nc') s=f.getslab('clt') x=vcs.init() x.plot(s) x.**flush**()
grid()	Set the default plotting region for variables that have more dimension values than the graphics method. This will also be used for animating plots over the third and fourth dimensions.	import vcs f=cu.open('clt.nc') s=f.getslab('clt') x=vcs.init() x.**grid**(12,24, -70,70, -150,150) x.plot(s)
resetgrid()	Set the plotting region to default values. That is, let the variable's dimension values determine the grid.	import vcs f=cu.open('clt.nc') s=f.getslab('clt') x=vcs.init() x.**resetgrid**() x.plot(s)

Index