January 12, 2007

DRAWxtl V4.3

A program to make ball-and-stick, or polyhedral crystal structure drawings

by

Larry Finger, Martin Kroeker and Brian Toby

This program reads a basic description of the crystal structure, which includes unit-cell parameters, space group, atomic coordinates, thermal parameters or a Fourier map, and outputs a geometry object that contains polyhedra, planes, lone-pair cones, spheres or ellipsoids, bonds, iso-surface Fouier contours and the unit-cell boundary. Since version 4.1, the program now contains a real-time display that renders the structure object using a series of openGL calls using the glut-programming interface. As before, scene description files for the popular POV ray-tracer program, or a Virtual Reality Modeling Language (VRML) file are produced. The POV scene file can be used as is, but it can also be edited to make use of the more sophisticated features of POV like transparency, backgrounds or light effects. The VRML version is suitable for use with WWW browsers such as FireFox, and can be used to make displays suitable for interactive examination by users. Source URL's for this and associated programs are given in Appendix A. The original version of this program was a module for the visualization software of Advanced Visual Systems, Inc.; however, that program has not been maintained for several years. That code was written by Larry Finger (Larry.Finger@lwfinger.net), Geophysical Laboratory, Carnegie Institution of Washington, 5251 Broad Branch Road, N.W., Washington, DC 20015-1305. The original POV and VRML modifications were proposed and coded by Martin Kroeker (martin@ruby.chemie.uni-freiburg.de), Institut für Anorganische und Analytische Chemie, Universität, Freiburg, Germany. Addition of the Fourier-contour code was accomplished by Brian Toby (brian.toby@ANL.gov), Advanced Photon Source, Argonne National Laboratory, 9700 South Cass Ave., Argonne, IL 60439 USA.

The literature reference for DRAWxtl is: Larry W. Finger, Martin Kroeker, and Brian H. Toby (2007) DRAWxtl, an open-source computer program to produce crystal-structure drawings, J. Applied Crystallography, V40, pp. 188-192.

.

Program Features:
DRAWxtl can make ball-and-stick, polyhedral, or mixed diagrams, with atoms represented as spheres or thermal ellipsoids. For the latter type, anisotropic coefficients can be input as Uij, Bij, or βij, depending upon what is available. New with V5.3 is the ability to plot modulated and composite crystals up to 3-D modulations. All of the standard modulation functions defined in the CIF definitions can be handled, except for the rigid-body terms. At present, modulated structures may only be input with a CIF description of the structure.

Polyhedral diagrams can be made with polyhedra of any desired shape, not just tetrahedra or octahedra. For zeolite drawings, it is also possible to make the 'polyhedron' of tetrahedral ions about the center of the cages.

The program can read structural input from CIF, CSD (the Cambridge Structure Database), GSAS, SCHAKAL, and SHELX formats, as well as its own native format. In this way, the crystallographic data for complicated structures do not have to be retyped. Fourier maps may be input in the ‘grd’ format from GSAS, the ‘stf’ format of JANA2000, the ‘w2k’ from WIEN2K, the ‘vsp’ format from VASP, or the ‘flp’ format of FullProf. Alternatively, if structure factors are available in CIF format, DRAWxtl will compute the Fourier map.

The program also outputs a table of bond distances from each of the atoms in the asymmetric unit to all other atoms. All symmetry operators are used in this calculation, with distances less than a specified limit (defaults to 3.5 Å) written to the listing file. This output can be used to check that input parameters are valid. For modulated structures, the minimum, maximum and average values for the bond length are calculated and listed. For anisotropic atoms, the eigenvalues (RMS amplitudes) and eigenvectors for the ellipsoids are computed and listed.

This program is written in highly portable C, with a little C++) and will run on a wide variety of machines including 386- to Pentium-based PC's, Macintoshes, high-end workstations, and Linux systems. The only requirements are a graphical-windows based system, a floating-point coprocessor, C and C++ compilers, and libraries for openGL, and GLUT. The developers use Microsoft Visual C 6.0 for Windows and gcc for Linux. For those people that do not wish to use a windowing system, it is possible to change one line in the main header file and build a version that has the behavior of V3.2 and earlier.

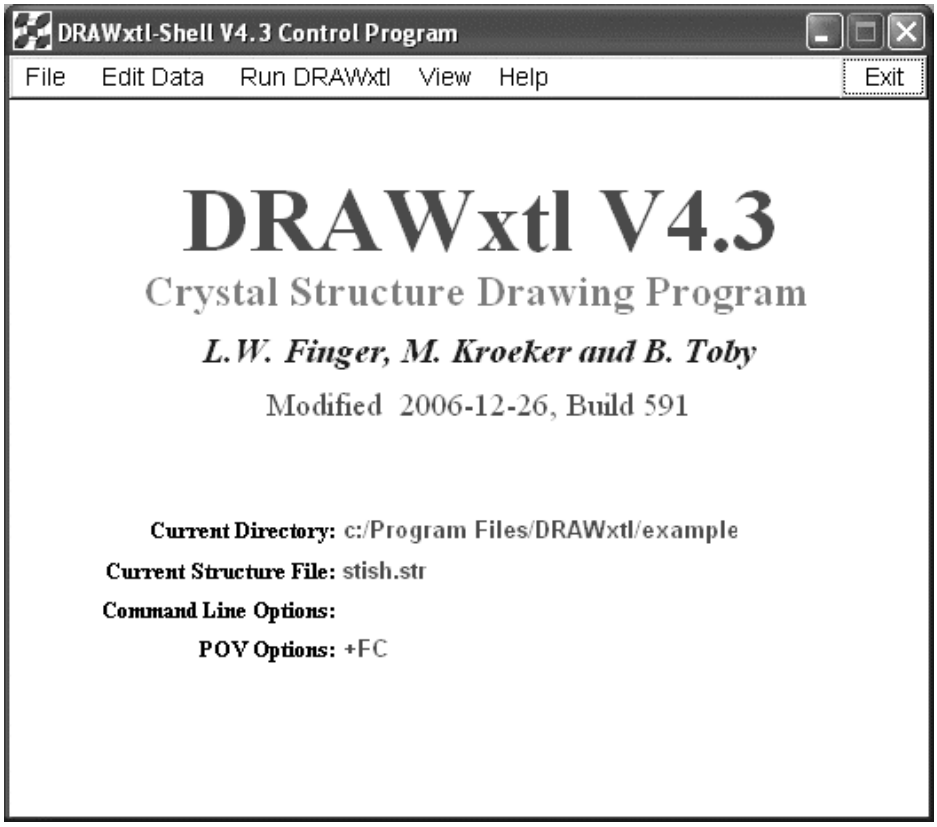
DRAWxtl is open-source software, and is free to anyone. We hold copyright on the code, and like any other piece of intellectual property, we ask that you respect our rights.

Program Operation:

This command-line version reads a text file of instructions that control the drawing produced. As in earlier versions, the program produces the POV and VRML output files, which are then read by the companion renderers/viewers to produce the hard copy output, or files suitable for viewing with a Web browser. The program also has the ability to go directly from the main screen to an Encapsulated Postscript file; thereby bypassing the external hard-copy program step for users that do not have POV installed. Although the program may be invoked from a command-line terminal screen, many users find it convenient to use a shell program that enables point-and-click access to the various steps used in preparing a crystal-structure drawing. Note: This shell is merely a front-end for batch operations. If you want a full graphical-user interface, please download V5.3 from http://www.lwfinger.net/drawxtl.

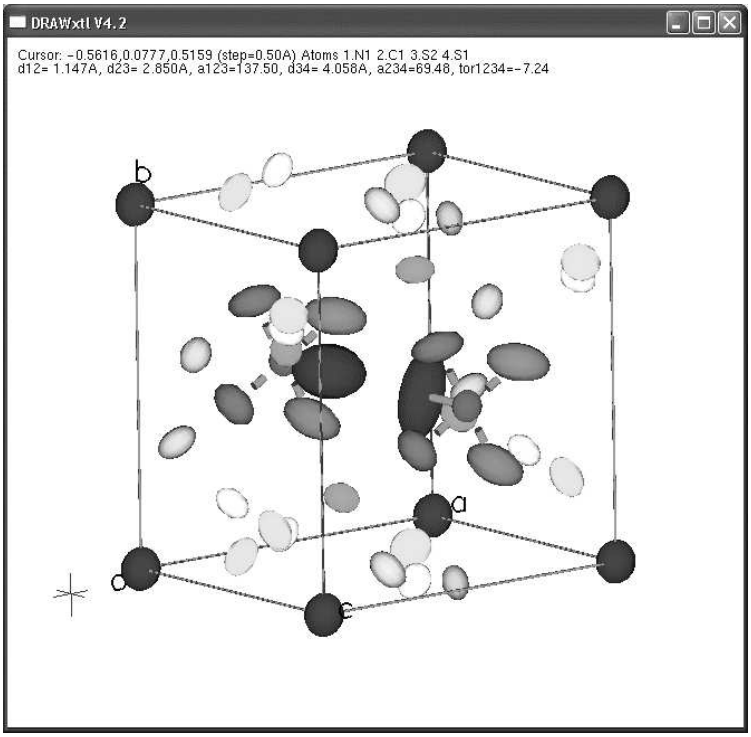
Figure 1 shows the window provided by **DRAW_shell43** running on Windows. With the exception of minor difference in the graphics, operation under Linux or OS X is identical.

Fig. 1: Screenshot of **DRAW_shell**.



Under the ‘File’ menu pulldown are entries that allow the user to select a data file for processing, configure the programs by entering the locations of the DRAWxtl, VRML and POV binaries, set the command-line or POV options, and exit the shell. The ‘Edit Data’ item opens the structure description file in a text editor window where it may be changed. The ‘Run DRAWxtl’ menu item executes the crystal-structure drawing and displays the result in an openGL window. The ‘View’ menu item has entries to render the POV file, display the VRML view, or display the DRAWxtl program listing. The ‘Help’ menu item provides access to the command-line, program input, and graphics screen commands.

Fig. 3. The openGL view



Another new feature in V4.2 and later is the graphics cursor shown in the lower, left rear of the diagram above. Its position in fractional coordinates is shown on the line at the top. The various bond distances and angles and the torsion angle are shown for the last 4 selected atoms. To aid in atom selection, the following keyboard shortcuts have been implemented:

- ‘C’ or ‘c’ - turn the cursor on. Each succesive press reduces the size of the steps. When the size is reduced below 0.01 A, the cursor is turned off.
- ‘x’, ‘y’, ‘z’ – move the cursor in the positive direction parallel to the x-, y- or z-axis.
- ‘X’, ‘Y’, ‘Z’ – move the cursor in the negative direction parallel to the x-, y- or z-axis.
- ‘P’ or ‘p’ - place the graphics cursor on the atom nearest the mouse position.
- ‘A’ or ‘a’ - place the cursor at the position of the atom nearest the cursor.
- ‘M’ or ‘m’ – move the cursor to the min(M) or max(m) in the electron-density.
- ‘L’ or ‘l’ - label the atom at the cursor position.
- ‘B’ or ‘b’ – Place the bond distance between atoms 1 and 2 in the list.
- ‘D’ or ‘d’ – dump the structure as an EPS file.
- ‘H’ or ‘h’ – toggle display of on-screen graphics help.
- ‘O’ or ‘o’ – toggle between orthographic and perspective projection.
- Hold left mouse and drag – rotate the graphics object using a virtual trackball.
- Hold right mouse and drag – zoom in/out on the object. On the Mac, hold down the command (Apple) key and drag the mouse.
- Up/Down/Left/Right arrows – pan motion
- HOME key – remove all zoom and pan motions.
- Hold shift, left mouse and drag – move labels or the triple vector to a desired position. If the triple vector is dragged, all 3 of its labels will be moved with it.
- Ctrl/left click – omit the object at the mouse position.
- ‘U’ or ‘u’ – Undo the previous omit operation. N.B. Does not work for labels.
- ‘Q’, ‘q’, or <ESC> - exit the program.

Command Line Switches for the program are as follows:

-a

sets the program to autolabel atoms from SHELX or CIF files.

-b xlim ylim zlim

defines X-, Y- and Z-limits of the view box in angstroms (defaults to -10..10 Angstrom around center)

-c

expands the VRML output by adding comments. This option should not be used for any VRML files placed in a Web site.

-C

switches the VRML camera to orthographic, rather than perspective mode.

-h, -?

displays usage information

-m M

changes the magnification factor for POV and VRML diagrams in case the initial size is not correct (defaults to 1.0)

-n

suppresses plotting of unit cell 'box'

-o xcenter ycenter zcenter

defines center of view box in crystal coordinates (defaults to 0.5 0.5 0.5)

-O is the same as -C

-p xmin xmax ymin ymax zmin zmax

defines a-,b-,c- plotting range in fractions of the axes, similar to the Pluto (Motherwell & Clegg 1978) PACK RANGE command (this is especially useful for hexagonal cells, where the orthorhombic view box does not always give satisfactory results)

-v xangle yangle zangle

defines view rotation angles around the axes of the Cartesian view coordinate system (equivalent to the view keyword) (defaults to 0 0 0)

-u

display a orientation vector triple at the edge of the diagram. If this switch is not used, the axes are labeled.

If command-line and input-file information for the same quantities are provided, the command-line values will control the resulting drawing.

Input Instructions for DRAWxtl:
In this program, color is represented in symbolic form, and must be one of the color names from POV's "colors.inc" file, which may be modified to define custom colors. The standard color names and associated RGB values are listed in Table I, and sorted by RGB values in Table II. Any of these colors may be made transparent by appending the phrase 'filter xx' after it, where 'xx' is a number from 0.0 to 1.0. The larger the value of 'xx', the more transparent will be the entity. Each line of the input file is preceded by a character sequence that describes the type of information, as follows:

arrow 'xp' 'yp' 'zp' 'xc' 'yc' 'zc' 'length' 'diameter' 'color'
defines the position in fractional coordinates (xp, yp, xz) of the nuclear cell, the components (xc, yc, zc) of the spin vector, and the length, diameter, and color of the arrows The reference direction for xc is parallel to direct space a, yc is parallel to $(a \times b) \times a$, and the reference direction for zc is perpendicular to $xc \times yc$. The only symmetry elements used in placing arrows are the translations described by the mag_trans command below.

atom 'name' 'number' 'x' 'y' 'z'
defines the atoms. The 1- to 4-character name will be used on the commands that describe the objects to be created, the number is to identify which atom of this type is referenced, and 'x' 'y' 'z' are the fractional coordinates in the unit cell. If the atom is located on a special position where the coordinate is given by a fraction such as ¼, the string '1/4' may be entered rather than 0.25.

average
causes the program to draw the average structure of an incommensurately modulated crystal even if information about positional or occupancy modulation is available in the CIF.

axislines 'width' 'color'
defines the width and color of the lines that depict the principal axes of ellipsoids. The color defaults to dark gray (Gray20). If this command is not given, any ellipsoids will be drawn with principal axes of 0.00015 times the overall scale factor, which should normally be appropriate**.**

background 'color'
sets the color of the background of the graphical views. The default color is white.

bestplane 'number' 'name1' 'name2' ... 'nameN' 'width' 'height' 'color'
causes the program to calculate the best fitting plane through a subset of atoms, where number is the number of unique atom names (name and number, e.g. C8) name1 to nameN that follow. The plane is drawn as a rectangle of dimensions width x height in the given color.

betaij 'name' 'number' 'β_{11}' 'β_{22}' 'β_{33}' 'β_{12}' 'β_{13}' 'β_{23}' 'color'
defines the anisotropic thermal coefficients for an atom and 'color' of the ellipsoid. The 'name' and 'number' should correspond to the atom input described above. For this form, the temperature factor is given by $T = \exp\{-\Sigma_j\Sigma_k h_j h_k \beta_{jk}\}$. In the POV version of the program, the principal ellipses are drawn in black. The ellipses will, of course, be invisible if the ellipsoid is also black.

bij or **Bij** 'name' 'B_{11}' 'B_{22}' 'B_{33}' 'B_{12}' 'B_{13}' 'B_{23}' 'color'
defines the anisotropic thermal coefficients for an atom and the color of the ellipsoid to be drawn. The 'name' and 'number' should correspond to the atom input described above. For this form, the temperature factor is given by $T = \exp\{-0.25\Sigma_j\Sigma_k h_j h_k B_{jk} a_j^* a_k^*\}$, where a_j^* and a_k^* are reciprocal lattice constants.

bond ‘name1’ ‘name2’ ‘radius’ ‘min length’ ‘max length’ ‘color’,
where ‘name1’ and ‘name2’ indicate the types of atoms to be connected by a bond, ‘radius’ is the radius of the resulting cylinder, and the minimum and maximum lengths are given in the same units as the unit cell.

box ‘radius’ ‘color’
defines the radius and color of the cylinders that form the unit cell boundary. If ‘radius’ is 0.0, plotting of the unit cell is suppressed. The radius of the cylinders will be scaled with the size of the drawing. The default size is 0.02.

cell ‘a-length’ ‘b-length’ ‘c-length’ ‘alpha’ ‘beta’ ‘gamma’
unit-cell lengths and angles. If no angles are listed, they are assumed to be the fixed values for the symmetry class.

clip ‘xmin’ ‘xmax’ ‘ymin’ ‘ymax’ ‘zmin’ ‘zmax’

defines a-,b-,c- clipping range in fractions of the axes. Any bonds extending beyond these limits will be cut off at half-length. This command is to be used in conjunction with the pack keyword to produce 'dangling' bonds in the display of framework structures.

cutout ‘color’ (used only for POV and openGL)
sets the POV generation of thermal ellipsoids to have one octant removed, as in the program ORTEP. If this command is not given, all ellipsoids will be complete. The color is for the planes that describe the edges of the cutout

dash **‘**name1’ ‘name2’ ‘radius’ ‘min’ ‘max’ ‘color’
where ‘name1’ and ‘name2’ indicate the types of atoms to be connected by a dashed bond, ‘radius’ is the radius of the resulting cylinder, and the minimum and maximum lengths are given in the same units as the unit cell.

depthcue ‘depth’ (used only for POV)
defines the extent to which the size of polyhedral edges is increased as the edge is closer to the viewer.

edges ‘radius’ ‘color’
defines the thickness and color of cylinders along the edges of polyhedra that may be used to emphasize the faces. The radius of these cylinders will also be scaled with the size of the drawing. By default, black edges of size 0.02 will be drawn.

ellipcolor ‘name’ ‘number’ ‘color’
defines the color for ellipsoids when the thermal ellipsoid information has been read from a CIF, CSD, GSAS, SCHAKAL or SHELX import or inline file. The ‘name’ and ‘number’ must match the identification information in the input file. The parameter ‘number’ may be an asterisk (*) to indicate all atoms with that name. In addition, these input lines must be after the **import** or **inline** command.

ellipsoids ‘probability’
sets the size of the ellipsoid such that that fraction of the electron density is contained within the bounding surface. Use either 0.50 or 50 to get the standard (default) 50% ellipsoids.

finish ‘ambient’ ‘diffuse’ ‘specular’ ‘roughness’
defines parameters for the POV lighting functions that are applied to all surfaces. Suggested values are 0.7 0.3 0.08 0.01 to reduce the harsh contrasts that can result from the default material properties in POV.

frame ‘comment’
similar to ‘end', marks the division between two sets of input that are to be superimposed in a single output file. Each frame must have a complete set of ‘atom’ lines, and lines

describing the objects such as bonds, polyhedra, spheres, etc. to be created in this frame. Distinct packing ranges and space grpups are optional. A prime example of this command would be to draw structures with adsorbed molecules with symmetry lower than the cage in which it resides. Another option is to draw ball-and-stick and polyhedral pictures in side-by-side unit cells. All other parameters are global and apply to all frames.

import ‘file type’ ‘filename’ ‘phase number(for GSAS only)’
causes the program to read structural information from an external file. ‘file type’ defines the format of that file, and ‘filename’ is the name of the file. Import filters have been written for the CIF, DISCUS, FDAT (Cambridge Structure Database), GSAS, PCR, SCHAKAL, SHELX, and WIEN2K formats. For GSAS format, the number of the phase should also be given. From these files, the atomic coordinates, thermal parameters, unit cell, and space group (CIF, FDAT, GSAS and SCHAKAL formats only) will be read. To set colors for the ellipsoids, use the **ellipcolor** command. You must also use the **ellipsoids** command to get ellipsoids displayed. In files that are imported, atom names of the form Si3A will have atom names of SiA with a number of 3.

inline ‘file type’
is similar to **import**, except that the ‘foreign’ input information is included in the DRAWxtl input file. This form works for CIF, FDAT, SCHAKAL and SHELX data. To set colors for the ellipsoids, use the **ellipcolor** command. You must also use the **ellipsoids** command to get ellipsoids displayed.

labelscale ‘size’

changes the relative size of the **labeltext** entries. The default value is 1.0.

labeltext ‘x’ ‘y’ ‘z’ ‘text string’

plots the given string at the specified position given in fractional coordinates.

list ‘maxdist’
causes the program to list bond distances up to ‘maxdist’ in the preliminary scan. If this command is not given, ‘maxdist’ defaults to 3.5 Å

lonepair ‘name’ ‘number’ ‘distance’ ‘radius1’ ‘radius2’ ‘color’
creates the specified number (either 1 or 2) of cones representing free electron pairs extending from atom 'name', where 'distance' is the length of the cone, ‘radius1’ is the size of the tip, and ‘radius2’ is the size of the spherical end cap.

lookat ‘u1’ ‘u2’ ‘u3’ ‘v1’ ‘v2’ ‘v3’
causes the program to select an orientation such that vector **u** is towards the viewer, and the projection of vector **v** is vertical. This command overrides any **view** command, or the **-v** switch on the command line.

mag_trans ‘Aa’ ‘Ab’ ‘Ac’ ‘Ba’ ‘Bb’ ‘Bc’ ‘Ca’ ‘Cb’ ‘Cc’
describes the relationship between the magnetic and nuclear unit cells. In this notation, the upper-case letter states which of the magnetic axes is being described, and the lower-case letter corresponds to the nuclear cell axis. This matrix defaults to the identity.

magnification ‘factor’
sets the factor to modify the overall scaling is case the automatic value is not correct. This command matches the -m command line switch.

mapcalclimits ‘xmin’ ‘xmax’ ‘ymin’ ‘ymax’ ‘zmin’ ‘zmax’
describes the region of direct space (in fractional coordinates) for which the map has been calculated. Map types that are self documenting such as FullProf and JANA2000 do not need this line. For other types, 0 to 1 in all three directions will be assumed.

mapcontour ‘level’ ‘style’ ‘color’

defines a new contour at 'level'. The style can be either 'mesh' or 'solid', and the color is set by 'color'

mapread ‘maptype’ ‘filename’ ‘calctype’

reads a Fourier map of type 'maptype' from the file named 'filename'. At present, GSAS-style (maptype = grd), JANA2000-style (maptype = stf), WIEN2k (maptype = w2k), VASP (maptype = vsp), FullProf (GFOURIER output, maptype = flp), and O Format (maptype = dn6) electron density maps are read, as are electron density and ELF files from the FP-LAPW program EXCITING (maptype=exc). If a Shelx/Cif-style Fo/Fc file (maptype = fcf) or a JANA-style M80 file (maptype = m80) is given, the electron density is calculated during the initial read, which may take a few seconds. Both A/B and Fo/phi data formats (Shelx commands LIST 3 and LIST 6) are supported. The calctype may be 'Fo', 'Fc', 'Fo-Fc', '2Fo-Fc', or Fo2 (i.e. Patterson) to indicate the type of map to calculate. If this parameter is not given, an 'Fo' map is calculated.

mapregion ‘xmin’ ‘xmax’ ‘ymin’ ‘ymax’ ‘zmin’ ‘zmax’

describes the region of direct space (in fractional coordinates) that the map is to be displayed in the output. If not entered, these values default to the values given under 'mapcalclimits'.

molcomp ‘dist’

causes any incomplete molecules in the display box to be completed. The value of ‘dist’ defines the maximum intramolecular distance. Caution: If this distance is greater than any intermolecular distance, or if the material is not molecular, the display list will overflow.

nolabels

removes all axis labels from the output diagrams.

noshadow

causes objects in the POV file not to cast shadows.

occupancy 'name' 'average' 'minimum'

defines the occupancy of the named site in the average structure of a modulated system, and the occupancy threshold for including individual copies in a plot of the modulated structure. Use a negative value for the sphere radius to scale atom sizes by their individual site occupancies.

origin ‘xcenter’ ‘ycenter’ ‘zcenter’

defines center of view box in crystal coordinates (defaults to 0.5 0.5 0.5).

orthographic

causes the camera to be changed from the normal perspective view to an orthographic view for VRML and removes all perspective from POV diagrams.

pack ‘xmin’ ‘xmax’ ‘ymin’ ‘ymax’ ‘zmin’ ‘zmax’

defines a-,b-,c- plotting range in fractions of the axes, similar to the Pluto (Motherwell & Clegg 1978) PACK RANGE command (this is especially useful for highly oblique cells, where the orthorhombic view box does not always give satisfactory results).

phaseshift 'value1' 'value2' 'value3'

defines the initial phases t_n of the n'th modulation wave in a modulated structure.

phong ‘value’ ‘size’ (used only for POV)

defines the amount of Phong highlighting on spheres and ellipsoids. The ‘value’ ranges between 0.0 and 1.0, where 0.0 gives no highlight, and 1.0 causes complete saturation at the center of the highlight. The ‘size’ ranges from 1.0 (very dull) to 250 (highly polished). The

default quantities are 0.1 and 1.0, which gives a large, dull highlight. If ‘value’ is 0.0, the image can be rendered much more quickly.

plane ‘name’ ‘length’ ‘color’
defines the center of a plane group, such as CO_3, that is to be drawn in a structure, where ‘name’ is the name of the atom at the center and ‘length’ is the maximum distance to coordinating anions.

polyedge ‘name’ ‘radius’ ‘color’
defines the thickness and color of cylinders used to emphasize the faces along the edges of polyhedra for atom ‘name’. The radius of these cylinders will also be scaled with the size of the drawing.

polysz ‘name’ ‘length’ ‘color’

defines a polyhedron, where ‘name’ is the name of an atom at the center of a polyhedron and ‘length’ is the maximum length of distances to atoms that are to be considered as the vertices of the polyhedron. The polyhedra can be of any desired complexity - more than tetrahedra or octahedra can be drawn. For polyhedra with both upper and lower limits (which might be desirable for intermetallic compounds), use the 'shell' command.

polytolerance ‘length’
polylimit ‘length’
polyfudge ‘length’
modifies the internal limit for the deviation of vertices from the common plane. While the default value (0.1) will always generate correct drawings, it may sometimes be desirable to increase it to create idealized views of nearly symmetrical polyhedra that would otherwise show creased surfaces.

polyvert ‘name1’ ‘name2’ ‘length’ ‘color’
defines polyhedra in the manner of the ‘polysz’ command, except that the polyhedron around the ‘name1’ atom will only include atoms of type ‘name2’.

rem, REM
Any line preceded by this command is ignored.

shell ‘name’ ‘length1’ ‘length2’ ‘color’
defines a polyhedral shell, where name is the name of an atom at the center of a polyhedron and length1 and length2 are the mimimum and maximum lengths of distances to atoms that are to be considered as the vertices of the polyhedron. The polyhedra can be of any desired complexity, and can be stacked as desired.

slab ‘a’ ‘b’ ‘c’ ‘alpha’ ‘beta’ ‘gamma’ ‘xoff’ ‘yoff’ ‘zoff’ ‘xrot’ ‘yrot’ ‘zrot’ ‘flag’
defines a (possibly oblique) cutout box of the specified axis lengths and angles that is offset by xoff,yoff zoff from the origin of the structure and rotated at angles xrot yrot zrot relative to it. If flag is set to 1, any part of the structure outside the box is deleted. If flag is 2, the outline of the box is overlaid on the unchanged image to allow accurate placement of the cutout box.

spgp, sgrp, or **spgr** ‘symbol’
Space Group name consisting of the Bravais lattice symbol (must be upper case) followed by a space, the elements parallel to the first axis followed by a space, etc. Examples are I 41/a m d, P 21/n, I a 3 d, P b n m, etc. The generators will always select the origin choice with a center of symmetry at the origin. Furthermore, all monoclinic cells will have the unique axis parallel to the *b* axis, unless the full symbol is used, i.e. P 1 1 21/n describes a monoclinic cell with *c* as the unique axis. N.B. Rhombohedral space groups must be represented in the hexagonal form.

sphere ‘name’ ‘radius’ ‘color’

where ‘name’ is a one- or two-character symbol of the atom type, ‘radius’ is the radius of the sphere in Angstrom, and ‘color’ is the color of the sphere to be drawn.

title, titl ‘descriptive material’

General description of the structure - this line may appear anywhere in the file, but is generally first.

uij, Uij ‘name’ ‘number’ ‘u_{11}’ ‘u_{22}’ ‘u_{33}’ ‘u_{12}’ ‘u_{13}’ ‘u_{23}’ ‘color’

defines the anisotropic thermal coefficients for an atom and color of the ellipsoid. The ‘name’ and ‘number’ should correspond to the atom input described above. For this form, the temperature factor is given by $T = \exp\{-2\pi^2\Sigma_j\Sigma_k h_j h_k b_{jk} a_j^* a_k^*\}$, where . a_j^* and a_k^* are reciprocal lattice constants.

vectors

turns on the orientation vector triple at a corner of the diagram.

view ‘xrot’ ‘yrot’ ‘zrot’)

where ‘xrot’, ‘yrot’ and ‘zrot’ are view rotation angles in Cartesian space. These values correspond to a rotation of ‘xrot’ about the *x* axis, followed by a rotation of ‘yrot’ about the **new** *y* axis, and, a rotation of ‘zrot’ about the **new** *z* axis.

vrml1 (used only for VRML)

causes the output VRML file to have the older, VRML1 syntax for compatibility with some older viewers that are not VRML97 compliant.

vrml2 or **vrml97** (used only for VRML)

These commands do nothing, but are kept for compatibility with old data files. VRML97 format is now the default. A description of some of the VRML97 features is given in Appendix B. Do NOT use this form unless your VRML viewer is VMRL97 compliant!

xyzoff ‘u1’ ‘u2’ ‘u3’

causes all atom coordinates to be shifted by **-u**. This command is used whenever the origin defined for a structure does not conform to the standard origin selected by the space-group generator.

end, END

is the last line of a file that is processed. Any information past this point will be ignored.

Appendix A: On-line sources for information and software.

- POV - Source code for the Persistence of Vision Ray Tracer (POV-Ray) software, user documentation and precompiled executables for several platforms are available through http://www.povray.org. Code is also available through ftp.povray.org, and a number of mirror sites listed on the web site. At the time of this writing, version 3.6.0 of POV is available.
- VRML - Documentation and viewers for the Virtual Reality Modeling Language are available through http://www.web3d.org. These files are particularly suitable for viewing across the Internet, therefore, many of the viewers are in the form of plug-ins for popular web browsers. In addition, stand-alone viewers are available. The files produced by DRAWxtl adhere to the VRML V1.0 format, or the VRML97 (2.0) format. For browser plug-ins, we recommend the free product available at http://www.parallelgraphics.com/products/cortona/. It supports Internet Explorer, Netscape, Mozilla, and FireFox browsers.
- Source code and Windows executables for ‘DRAWxtl’ are available at:
  http:// www.lwfinger.net/drawxtl/

UNIX users should download DRAWxtl43_tar.gz, a gzip-compressed tar-format file. Windows users should download DRAWxtl43.zip. These files contain the current version of the source (in the C++ language), and a collection of sample input files that give examples of all of the features of

the program. The program will compile and build under Microsoft Visual C++6.0 and a number of C++ compilers for UNIX systems, normally ‘g++’, as long as the necessary graphics libraries are available. Windows users only interested in the executables should download DRAWV43.EXE. If any users experience difficulties in compiling the program, please contact LWF at Larry.Finger@lwfinger.net. Windows users do not need a C++ compiler. Also included in the Web archives are links to the current versions of POVray and VRML viewers.

Appendix B: Adding life to VRML files

One of the major differences between VRML-1 and VRML97 (VRML-2) is the availability of functions that can be used to create dynamic effects such as motion and color changes. While their purpose is more obvious in macroscopic environments like rooms and buildings, these can also be a useful addition to the crystal structures generated with DRAWxtl, especially in an educational context.

As there does not appear to be a universally applicable set of functions, they are not automatically included in the files generated by DRAWxtl. This section provides only a short overview of the features we have found useful in our work.

A detailed description can be found in the VRML97 standard document (http://vag.vrml.org/VRML2.0/FINAL) and in monographs like Carey & Bells 'The Annotated VRML 2.0 Reference Manual', which is also available online at http://www.aw.com/devpress/titles/41974.html.

1. Naming objects

Dynamic effects can be added to a static scene by adding a unique name to one of the positional or material declarations (Transform or Material nodes). Its subfields like translation, rotation or color can then be set by a function that is driven by an internal timer, the viewer clicking on an object, or a combination of both.

Thus, a fragment like

```
Transform {
  translation -0.87100 -1.50862 -2.88
  Shape {
    geometry Sphere {radius .5}
  }
}
```

which would place an atom of diameter 0.5 at -0.871 -1.50862 -2.88, could be turned into

```
DEF movingSphere Transform {
  Shape {
    geometry Sphere {radius .5}
  }
}
```

where some function would generate translations as desired.

2. Using the internal timer

All standard-conforming VRML97 browsers are expected to provide an internal clock called a TimeSensor. Its output - wallclock time in seconds - can be used as a trigger, and the fractional seconds form a sawtooth-like signal in the range from 0 to 1 that can drive periodic events.

This takes the form:

```
DEF Wallclock TimeSensor {
  loop TRUE
  stopTime -1
  cycleInterval 10
}
```

(where the speed can be regulated via the cycleInterval value). To continue our example, we could now use the output of this function to move our sphere. However, its 'heartbeat', the

fractionChanged signal, is always between 0 and 1, so we need an auxiliary function to turn this into realistic coordinates.

3. Generating new coordinates or color codes

VRML97 provides a series of 'Interpolator' functions to turn a periodic signal into a smooth succession of coordinate or color values.

These take the form:

```
DEF Parabola PositionInterpolator {
  key [ 0 .25 .5 .75 1 ]
  keyValue [ -0.87100 -1.50862 -2.88,
             -0.87100  0.50862 -2.88,
             -0.87100  1.50862 -2.88,
             -0.87100  0.50862 -2.88,
             -0.87100 -1.50862 -2.88]
}
```

where the 'keyValues' provided for the set of 'keys' are used to compute the function that links input and output. The example above would make our atom oscillate along the y axis around its initial position.

Similar functions exist for rotations:

```
DEF turnZ OrientationInterpolator {
  key [ 0 .333 .666 1 ]
  keyValue [ 0 0 1 0,
             0 0 1  2.09,
             0 0 1  4.18,
             0 0 1 0.]
}
```

and for colors:

```
DEF atomcolor ColorInterpolator {
  key [ 0 .1 .2 .3 .4 .5 .6 .7 .8 .9 1. ]
  keyValue  [1.    0.6   0.,
             1.    0.65  0.,
             1.    0.7  0.,
             1.    0.8  0.,
             1.    0.9  0.,
             1.    1.  0.
             1.    0.9  0.,
             1.    0.8 0.,
             1.    0.7  0.,
             1.    0.65  0.,
             1.    0.6   0.     ]
}
```

(which would make our atom turn orange, yellow, and orange again (which may even be useful if it belongs to different coordination polyhedra or unit cells at various stages of its movement).

4. Connecting effects and objects

The values that are produced by Sensor or Interpolator functions have to be sent to the desired objects (or other functions) via specific routes. In our example, we must feed the output of our TimeSensor to our OrientationInterpolator, and the new coordinates that it generates must be routed to the Transform node around our atom.

Thus,

```
ROUTE Wallclock.fraction_changed TO Parabola.set_fraction
ROUTE Parabola.value_changed TO movingSphere.set_translation
```

and

```
ROUTE Wallclock.fraction_changed TO turnZ.set_fraction
ROUTE turnZ.value_changed TO movingSphere.set_rotation
```

or

```
ROUTE Wallclock.fraction_changed TO atomcolor.set_fraction
ROUTE atomcolor.value_changed  TO spherematerial.set_color
```

(for the other examples) is needed to put our atom in motion.

5. More general functions

Further (and more complex) dependencies between events and actions can be defined via small functions or programs written in JavaScript.

If, for instance, we wanted to turn the wallclock time itself into a signal that fits our purpose better than the sawtooth-like 'fraction_changed', we could pass it through this function:

```
DEF mytimer Script {
  eventIn SFTime division
  eventOut SFFloat output
    url "vrmlscript:
      function division (input) {
      output = input / 3 - Math.floor (input/3);
      }
    "
}
```

and then

```
ROUTE Wallclock.time TO mytimer.division
ROUTE mytimer.output TO Parabola.set_fraction
ROUTE Parabola.value_changed TO movingSphere.set_translation
```

to move our atom.

6. Switching between alternate representations

It is sometimes useful to depict a structure in different representations, for instance as an ORTEP-style drawing of the individual atoms and reduced to the coordination polyhedra around the cation or the anion. With VRML97, these can be combined in a single file, where clicking on any part of the structure switches to another representation.

To achieve this, the different parts form the choices of a 'Switch' statement that is coupled to a TouchSensor:

```
DEF selectOne Switch {
  choice [
  View1{}
  View2{}
  View3{}
  ]
  whichChoice 0
}
DEF ClickMe TouchSensor{}
```

(where the different Views are either complete definitions or references to prototypes as explained below). The TouchSensor envelopes the object that is contained in the same Transform node (which is the complete structure in this case), it is triggered whenever the viewer moves the pointer over it and presses a button. Normally, only the first view (choice 0 of the switch) is visible.

If the output of the TouchSensor is connected to a Script that changes the choice counter:

```
DEF Trigger Script {
eventIn SFBool doit
eventOut SFInt32 set_hide
        url "javascript:
        function doit(touched) {
            if (touched) {
              set_hide = set_hide +1 ;
              if (set_hide > 2) set_hide = 0;
            }
        }
        "
}

ROUTE Trigger.set_hide TO selectOne.set_whichChoice
ROUTE ClickMe.isActive TO Trigger.doit
```

the representation can be switched at any time by a simple mouse click, while the orientation is preserved.

7. Animating larger frameworks

Many dynamic effects can be used in the same scene file. Note, however, that the destinations of a ROUTE must have unique names and there must be individual ROUTEs leading from the

transforming function to them. So, if you want several moving spheres in the example above, you must name them

```
DEF movingSphere1 {...}
DEF movingSphere2 {...}
```

and route the new coordinates with

```
ROUTE Parabola.value_changed TO movingSphere1.set_translation
ROUTE Parabola.value_changed TO movingSphere2.set_translation
```

If you want to modify groups of objects (several atoms or polyhedra) in the same way, it is probably best to add a Transform node that encompasses all of them, and then modify its properties. If you want to animate larger entities (polyhedral building blocks or layers) within a crystal structure, it may be easier to have DRAWxtl generate the desired portions separately and then copy them into the final file than to identify and group them in the output of a single DRAWxtl run. To improve readability, you can also group these sections as PROTOtypes that can be referenced through their names. Thus, a combination of a ball-and-stick model and a polyhedral component could be grouped as:

```
PROTO BallAndStick [] {
  Transform {
   children [
       ... lots of Spheres and Cylinders here ...
   ]
  }
}
PROTO Polyhedra[] {
  Transform {
   children [
       ... lots of IndexedFaceSets ...
   ]
  ]
}
```

and finally called as:

```
BallAndStick{}
Polyhedra{}
```

which is especially attractive if this is part of a Switch node (as described above) or a layered structure where the layers are animated to visualize phase transitions.

File ‘vrml2.wrl’ in the distribution set demonstrates a self-rotating fragment of the faujasite structure.

Table I. Color Names and Associated RGB Values.

Color Name	R	G	B	Color Name	R	G	B
Aquamarine	0.439216	0.858824	0.576471	IndianRed	0.309804	0.184314	0.184314
BakersChoc	0.36	0.2	0.09	Khaki	0.623529	0.623529	0.372549
Black	0	0	0	LightBlue	0.74902	0.847059	0.847059
Blue	0	0	1	LightGray	0.658824	0.658824	0.658824
BlueViolet	0.62352	0.372549	0.623529	LightGrey	0.658824	0.658824	0.658824
Brass	0.71	0.65	0.26	LightSteelBlue	0.560784	0.560784	0.737255
BrightGold	0.85	0.85	0.1	LightWood	0.91	0.76	0.65
Bronze	0.55	0.47	0.14	LimeGreen	0.196078	0.8	0.196078
Bronze2	0.65	0.49	0.24	Magenta	1	0	1
Brown	0.647059	0.164706	0.164706	MandarinOrange	0.89	0.47	0.2
CadetBlue	0.372549	0.623529	0.623529	Maroon	0.556863	0.137255	0.419608
Clear	1	1	1	MediumAquamarine	0.196078	0.8	0.6
CoolCopper	0.85	0.53	0.1	MediumBlue	0.196078	0.196078	0.8
Copper	0.72	0.45	0.2	MediumForestGreen	0.419608	0.556863	0.137255
Coral	1	0.498039	0	MediumGoldenrod	0.917647	0.917647	0.678431
CornflowerBlue	0.258824	0.258824	0.435294	MediumOrchid	0.576471	0.439216	0.858824
Cyan	0	1	1	MediumSeaGreen	0.258824	0.435294	0.258824
DarkBrown	0.36	0.25	0.2	MediumSlateBlue	0.498039	1	
DarkGreen	0.184314	0.309804	0.184314	MediumSpringGreen	0.498039	1	
DarkOliveGreen	0.309804	0.309804	0.184314	MediumTurquoise	0.439216	0.858824	0.858824
DarkOrchid	0.6	0.196078	0.8	MediumVioletRed	0.858824	0.439216	0.576471
DarkPurple	0.53	0.12	0.47	MediumWood	0.65	0.5	0.39
DarkSlateBlue	0.419608	0.137255	0.556863	Mica	0	0	0
DarkSlateGray	0.184314	0.309804	0.309804	MidnightBlue	0.184314	0.184314	0.309804
DarkSlateGrey	0.184314	0.309804	0.309804	Navy	0.137255	0.137255	0.556863
DarkTan	0.59	0.41	0.31	NavyBlue	0.137255	0.137255	0.556863
DarkTurquoise	0.439216	0.576471	0.858824	NeonBlue	0.3	0.3	1
DarkWood	0.52	0.37	0.26	NeonPink	1	0.43	0.78
DimGray	0.329412	0.329412	0.329412	NewMidnightBlue	0	0	0.61
DimGrey	0.329412	0.329412	0.329412	NewTan	0.92	0.78	0.62
DkGreenCopper	0.29	0.46	0.43	OldGold	0.81	0.71	0.23
DustyRose	0.52	0.39	0.39	Orange	1	0.5	0
Feldspar	0.82	0.57	0.46	OrangeRed	1	0.498039	0
Firebrick	0.556863	0.137255	0.137255	Orchid	0.858824	0.439216	0.858824
Flesh	0.96	0.8	0.69	PaleGreen	0.560784	0.737255	0.560784
ForestGreen	0.137255	0.556863	0.137255	Pink	0.737255	0.560784	0.560784
Gold	0.8	0.498039	0.196078	Plum	0.917647	0.678431	0.917647
Goldenrod	0.858824	0.858824	0.439216	Quartz	0.85	0.85	0.95
Gray	0.752941	0.752941	0.752941	Red	1	0	0
Gray05	0.05	0.05	0.05	RichBlue	0.35	0.35	0.67
Gray10	0.1	0.1	0.1	Salmon	0.435294	0.258824	0.258824
Gray15	0.15	0.15	0.15	Scarlet	0.55	0.09	0.09
Gray20	0.2	0.2	0.2	SeaGreen	0.137255	0.556863	0.419608
Gray25	0.25	0.25	0.25	SemiSweetChoc	0.42	0.26	0.15
Gray30	0.3	0.3	0.3	Sienna	0.556863	0.419608	0.137255
Gray35	0.35	0.35	0.35	Silver	0.9	0.91	0.98
Gray40	0.4	0.4	0.4	SkyBlue	0.196078	0.6	0.8
Gray45	0.45	0.45	0.45	SlateBlue	0	0.498039	1
Gray50	0.5	0.5	0.5	SpicyPink	1	0.11	0.68
Gray55	0.55	0.55	0.55	SpringGreen	0	1	0.498039
Gray60	0.6	0.6	0.6	SteelBlue	0.137255	0.419608	0.556863
Gray65	0.65	0.65	0.65	SummerSky	0.22	0.69	0.87
Gray70	0.7	0.7	0.7	Tan	0.858824	0.576471	0.439216
Gray75	0.75	0.75	0.75	Thistle	0.847059	0.74902	0.847059
Gray80	0.8	0.8	0.8	Turquoise	0.678431	0.917647	0.917647
Gray85	0.85	0.85	0.85	VeryDarkBrown	0.35	0.16	0.14
Gray90	0.9	0.9	0.9	Violet	0.309804	0.184314	0.309804
Gray95	0.95	0.95	0.95	VioletRed	0.8	0.196078	0.6
Green	0	1	0	VLightGrey	0.8	0.8	0.8
GreenCopper	0.32	0.49	0.46	Wheat	0.847059	0.847059	0.74902
GreenYellow	0.576471	0.858824	0.439216	White	1	1	1
Grey	0.752941	0.752941	0.752941	Yellow	1	1	0
HuntersGreen	0.13	0.37	0.31	YellowGreen	0.6	0.8	0.196078

Table II. Color Names Sorted by RGB Values.

Color Name	R	G	B	Color Name	R	G	B
Black	0	0	0	Firebrick	0.556863	0.137255	0.137255
Mica	0	0	0	Maroon	0.556863	0.137255	0.419608
NewMidnightBlue	0	0	0.61	Sienna	0.556863	0.419608	0.137255
Blue	0	0	1	LightSteelBlue	0.560784	0.560784	0.737255
SlateBlue	0	0.498039	1	PaleGreen	0.560784	0.737255	0.560784
Green	0	1	0	MediumOrchid	0.576471	0.439216	0.858824
SpringGreen	0	1	0.498039	GreenYellow	0.576471	0.858824	0.439216
Cyan	0	1	1	DarkTan	0.59	0.41	0.31
Gray05	0.05	0.05	0.05	DarkOrchid	0.6	0.196078	0.8
Gray10	0.1	0.1	0.1	Gray60	0.6	0.6	0.6
HuntersGreen	0.13	0.37	0.31	YellowGreen	0.6	0.8	0.196078
Navy	0.137255	0.137255	0.556863	BlueViolet	0.62352	0.372549	0.623529
NavyBlue	0.137255	0.137255	0.556863	Khaki	0.623529	0.623529	0.372549
SteelBlue	0.137255	0.419608	0.556863	Brown	0.647059	0.164706	0.164706
ForestGreen	0.137255	0.556863	0.137255	Bronze2	0.65	0.49	0.24
SeaGreen	0.137255	0.556863	0.419608	MediumWood	0.65	0.5	0.39
Gray15	0.15	0.15	0.15	Gray65	0.65	0.65	0.65
MidnightBlue	0.184314	0.184314	0.309804	LightGray	0.658824	0.658824	0.658824
DarkGreen	0.184314	0.309804	0.184314	LightGrey	0.658824	0.658824	0.658824
DarkSlateGray	0.184314	0.309804	0.309804	Turquoise	0.678431	0.917647	0.917647
DarkSlateGrey	0.184314	0.309804	0.309804	Gray70	0.7	0.7	0.7
MediumBlue	0.196078	0.196078	0.8	Brass	0.71	0.65	0.26
SkyBlue	0.196078	0.6	0.8	Copper	0.72	0.45	0.2
LimeGreen	0.196078	0.8	0.196078	Pink	0.737255	0.560784	0.560784
MediumAquamarine	0.196078	0.8	0.6	LightBlue	0.74902	0.847059	0.847059
Gray20	0.2	0.2	0.2	Gray75	0.75	0.75	0.75
SummerSky	0.22	0.69	0.87	Gray	0.752941	0.752941	0.752941
Gray25	0.25	0.25	0.25	Grey	0.752941	0.752941	0.752941
CornflowerBlue	0.258824	0.258824	0.435294	VioletRed	0.8	0.196078	0.6
MediumSeaGreen	0.258824	0.435294	0.258824	Gold	0.8	0.498039	0.196078
DkGreenCopper	0.29	0.46	0.43	Gray80	0.8	0.8	0.8
Gray30	0.3	0.3	0.3	VLightGrey	0.8	0.8	0.8
NeonBlue	0.3	0.3	1	OldGold	0.81	0.71	0.23
IndianRed	0.309804	0.184314	0.184314	Feldspar	0.82	0.57	0.46
Violet	0.309804	0.184314	0.309804	Thistle	0.847059	0.74902	0.847059
DarkOliveGreen	0.309804	0.309804	0.184314	Wheat	0.847059	0.847059	0.74902
GreenCopper	0.32	0.49	0.46	CoolCopper	0.85	0.53	0.1
DimGray	0.329412	0.329412	0.329412	BrightGold	0.85	0.85	0.1
DimGrey	0.329412	0.329412	0.329412	Gray85	0.85	0.85	0.85
VeryDarkBrown	0.35	0.16	0.14	Quartz	0.85	0.85	0.95
Gray35	0.35	0.35	0.35	MediumVioletRed	0.858824	0.439216	0.576471
RichBlue	0.35	0.35	0.67	Orchid	0.858824	0.439216	0.858824
BakersChoc	0.36	0.2	0.09	Tan	0.858824	0.576471	0.439216
DarkBrown	0.36	0.25	0.2	Goldenrod	0.858824	0.858824	0.439216
CadetBlue	0.372549	0.623529	0.623529	MandarinOrange	0.89	0.47	0.2
Gray40	0.4	0.4	0.4	Gray90	0.9	0.9	0.9
DarkSlateBlue	0.419608	0.137255	0.556863	Silver	0.9	0.91	0.98
MediumForestGreen	0.419608	0.556863	0.137255	LightWood	0.91	0.76	0.65
SemiSweetChoc	0.42	0.26	0.15	Plum	0.917647	0.678431	0.917647
Salmon	0.435294	0.258824	0.258824	MediumGoldenrod	0.917647	0.917647	0.678431
DarkTurquoise	0.439216	0.576471	0.858824	NewTan	0.92	0.78	0.62
Aquamarine	0.439216	0.858824	0.576471	Gray95	0.95	0.95	0.95
MediumTurquoise	0.439216	0.858824	0.858824	Flesh	0.96	0.8	0.69
Gray45	0.45	0.45	0.45	Red	1	0	0
MediumSlateBlue	0.498039	1		Magenta	1	0	1
MediumSpringGreen	0.498039	1		SpicyPink	1	0.11	0.68
Gray50	0.5	0.5	0.5	NeonPink	1	0.43	0.78
DarkWood	0.52	0.37	0.26	Coral	1	0.498039	0
DustyRose	0.52	0.39	0.39	OrangeRed	1	0.498039	0
DarkPurple	0.53	0.12	0.47	Orange	1	0.5	0
Scarlet	0.55	0.09	0.09	Yellow	1	1	0
Bronze	0.55	0.47	0.14	Clear	1	1	1
Gray55	0.55	0.55	0.55	White	1	1	1