

# MacCubeView

## v1.1.0

©Daniel W. Rickey  
1993

London, Ontario  
CANADA

drickey@irus.rri.uwo.ca  
FAX 519-663-3789

### Notice

This software may be used by not-for-profit organisations, including universities and colleges. It may not be used or distributed by any for-profit organisations. If the MacCubeView software is redistributed to another not-for-profit organisation, then this document must accompany it. If you obtain a copy of this software and are using it, then I request that you make a donation to myself at the address below. The amount of the donation is up to you. In exchange, I will supply you with a serial number that will enable the "save as..." options of MacCubeView.

Daniel W. Rickey  
#22-550 Platt's Lane  
London, Ontario  
CANADA N6G 3A8

### History

First release 1.0.0β	1993-05-19	
Second release 1.0.1β	1993-08-06	checked for the presence of colour QuickDraw as well as a math co-processor before running.
Third release 1.0.2β	1993-08-20	Changed constraint on maximum image size from 256 to 1024 pixels. Removed some bugs that caused the programme to hang. Removed the "Accurate Draw" option.

Third release 1.0.3 $\alpha$	1993-09-01	Added scroll bars to the image display window. Now saves the colour palette with a saved screen TIFF image file.
Fourth release 1.1.0	1993-12-05	added: -user registration & serial number dialogue - ability to save 3D data as a vff file - ability to load an HDF format colour palette - a pseudo-colour LUT & colour phase control

## Overview

### Description

MacCubeView is designed to display a texture map image of three-dimensional (3-D) data. The data in mind is typically generated by medical imaging techniques such as CT, MRI, and nuclear medicine. Some geophysics techniques also produce suitable 3-D image data.

### Hardware Requirements

MacCubeView will probably run on any Colour Macintosh Computer that is running System 7 or newer. The software will be at its best when used with a large eight-bit colour monitor. The Macintosh should have at least eight megabytes of memory installed. It will not run on a Mac Plus (sigh).

### Image Data

The 3D data set must be in the form of a X-Y-Z cube. All voxels must be eight bits deep. The maximum size for any one dimension is 1024 voxels, and the minimum size is four voxels. Up to eight 3-D data sets may be loaded at one time, however, only one image will be displayed with the correct colour map.

### Demonstration MRI Image Data

The image included with MacCubeView is a 3-D MRI scan of the author's head, and is presumed to be normal. The scan was performed with a General Electric Signa 0.5 Tesla magnetic resonance imager using a 3D spoiled gradient recalled echo sequence. The following parameters were used; flip angle = 45°, echo time = 8 ms, repetition time = 33 ms, received bandwidth = 16 kHz, scan time = 9' 10", frequency direction = A/P. The resultant 256 x 256 x 60 by sixteen-bit data set has been chopped down to a 128 x 128 x 57 by

eight-bit data-set in order to save network bandwidth. The bright white vial shown in the image indicates the right side of the head. The field of view is 28 cm x 28 cm, and the slice thickness is 0.5 cm. Thus, the aspect ratio of the image is 1.0 x 1.0 x 2.6.

### **Programme Windows**

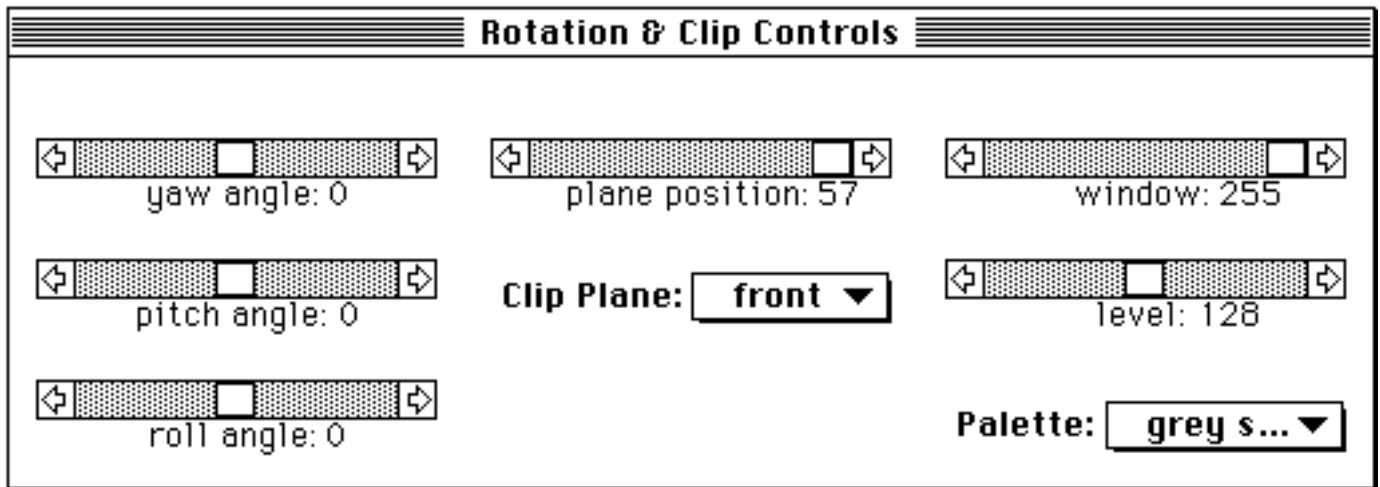
MacCubeView uses one window for controls, and up to eight windows for displaying the images of the 3D data sets. The control window is always visible, however, it is only active when at least one 3D data set is loaded. If there is more than one data set loaded, then the control window will correspond to what ever image window is active. An image window is made active by selecting it with the mouse.

#### **Image Window(s)**

An image window is used to display an image of a 3D data set. A linear pixel intensity ramp can be drawn on the left side of the window in order to illustrate the palette being used. If more than one data set is loaded, then only one image window can be active at a time. The 3D data set displayed in the active image window will be affected by the settings of the controls located in the control window. The active image window is also affected by the cube parameters option located under the special menu. The maximum size of a window is determined by the size of the data set and the scale factors.

#### **Control Window**


The control window contains the slide controls, which affect the 3D data set. This window is always active when data set is loaded. The yaw, pitch and roll controls set the relative orientation of the 3D data set. Each 3D data set has six sides: top, bottom, front, back, left, and right. Thus, there are six corresponding clip planes for slicing into the data set. The desired clip plane can be selected via the clip plane pop-up menu. The position of the selected clip plane is set with the clip position slide control.



The control window also contains a pop-up menu for selecting the desired colour palette. When a grey-scale palette is selected, window and level slide controls will be displayed. The user can also select a pseudo-colour or HDF-format palette. In these cases, a phase slider and invert palette check box will be displayed. The phase control sets the origin of the palette, and the invert check box changes the direction of the palette.

## Dialogue Boxes

### Cube Parameters

The cube parameters dialogue is located under the special menu (-j). This dialogue contains three entries for setting the width (X), height (Y), and depth (Z) scale factors. These scale factors set the size of the displayed 3-D data set, and can be used to change the aspect ration of the data set. Note that a smaller scale factors, and thus, a smaller image will result in a shorter display time. The maximum scaled dimension of a data set side is 1024 voxels. both the scaled and non-scaled data set dimensions are displayed for reference.

**DansHead.vff**

**Data Set Title:** Daniel W. Rickey's Head

---

**3D Image Size (H•Y•Z):** 128 • 128 • 57

**Scaled 3D Image Size (H•Y•Z):** 128 • 128 • 148

---

<p style="text-align: center;"><b>Cube Scale Factors</b></p> <ul style="list-style-type: none"> <li>• width <input style="width: 100px;" type="text" value="1.0"/></li> <li>• height <input style="width: 100px;" type="text" value="1.0"/></li> <li>• depth <input style="width: 100px;" type="text" value="2.6"/></li> </ul>	<p style="text-align: center;"><b>Display Technique</b></p> <p><input checked="" type="radio"/> texture map</p> <p><input type="radio"/> ray cast</p>	<p style="text-align: center;"><b>Display Options</b></p> <p><input checked="" type="checkbox"/> Show Colour Bar</p> <p><input checked="" type="checkbox"/> Show Cube Outline</p>
--	---	---

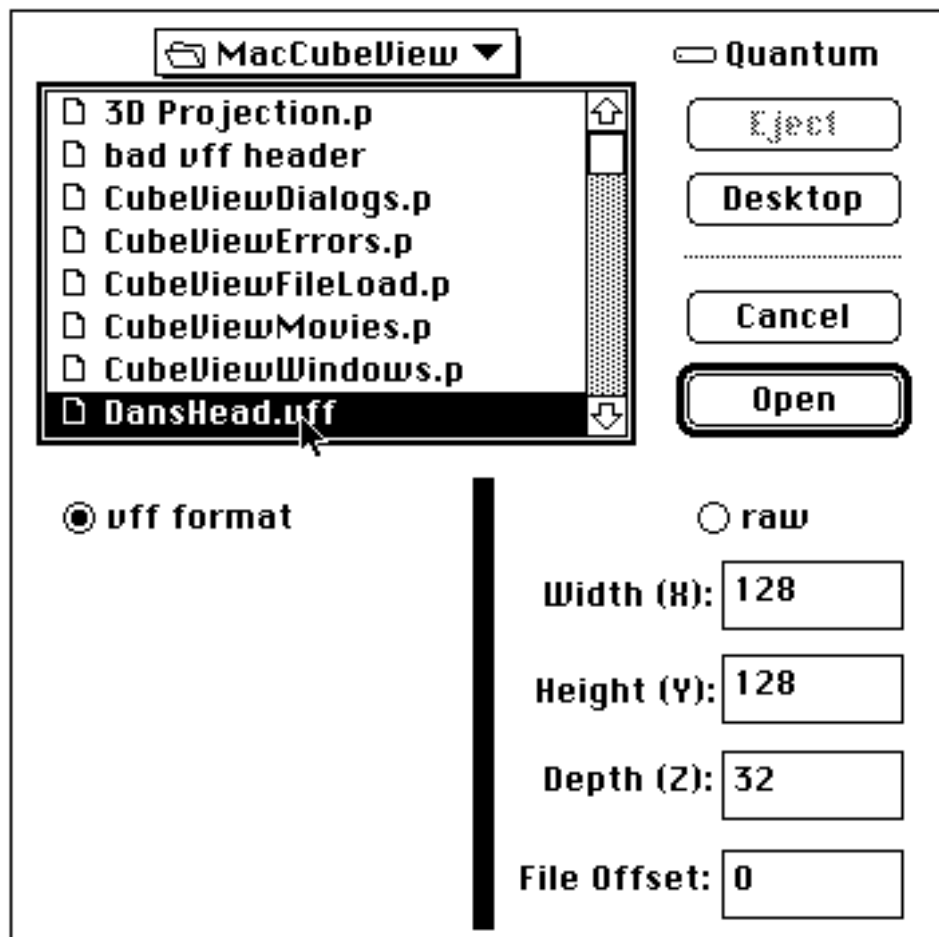
  

There is an option for choosing the display technique. At present, the raycast option is NOT functional. Check boxes allow the grey-scale ramp and the cube outline to be drawn or not. The title of the data set can also be changed. The title is independent of the file name.

### Open 3-D Data Set...

The open file dialogue (-o) allows the user to load a 3-D data set into MacCubeView. Only vff format and raw eight-bit data sets can be loaded. When the vff format radio button is set, the width, height, depth, scale factors, and title of the data set are given by fields in the vff header.

When the raw radio button is set, the width, height, and depth (X,Y,Z) of the data set is determined by entries in the three text boxes. The user can also specify the initial offset of the file in order to remove a file header. If there is no header, then the offset should be set to zero.



### Open HDF Colour Palette...

This option allows the user to load a HDF format colour palette. A loaded palette is made active by selecting the HDF file option from the pop-up palette menu located in the control window. HDF format palette can be generated easily with a public domain programme called NCSA PalEdit 2.0. NCSA PalEdit is available via anonymous FTP from [zaphod.ncsa.uiuc.edu](http://zaphod.ncsa.uiuc.edu).

### Save Screen Image as...

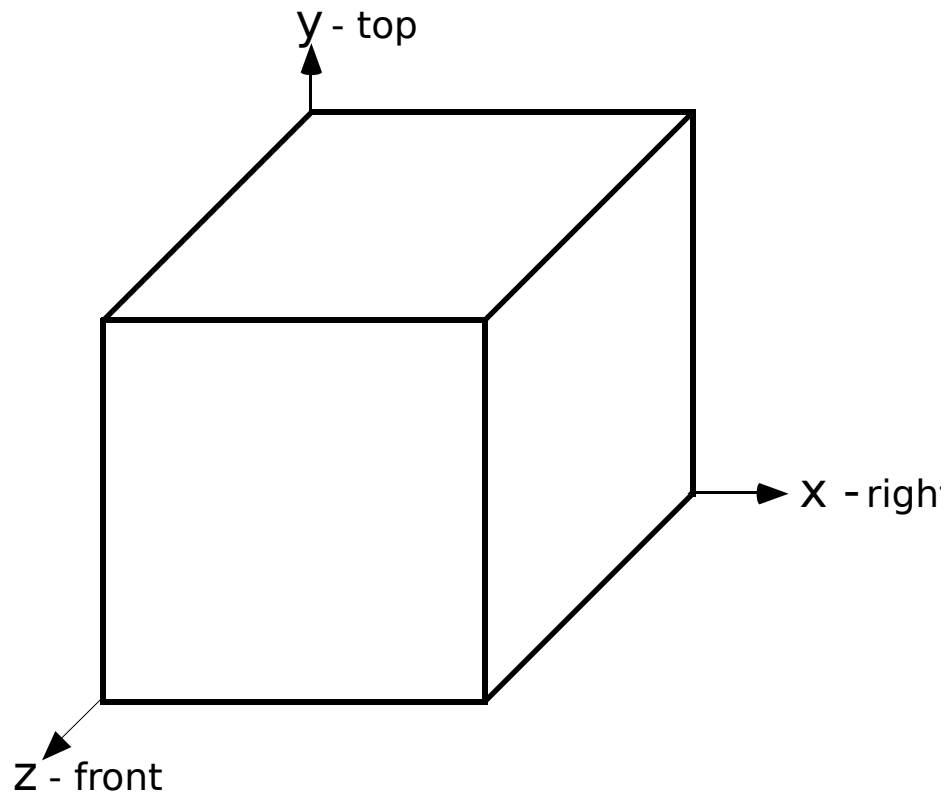
The save screen image as dialogue (⌘-s) allows the user to save the image that is displayed in the active image window. The image is saved as a TIFF file, which contains the colour palette. The saved image will also contain the displayed grey ramp, unless the ramp has been turned off via the check box contained in the cube parameters dialogue box. Note that this feature is available only when a correct registration number has been entered during initial programme start-up.

**Save 3-D Data Set as...**

The save 3-D data set as dialogue allows the user to save the data set that is currently active as a vff format file. The saved data set will contain the scale factors and title determined by the setting contained in the cube parameters dialogue box. The saved file is also given a MacCubeView icon, and thus, will load from the finder when double clicked upon. This option is useful when a data set was loaded as raw, as saving it places the dimensions and scale factors in the file's header. Note that this feature is available only when a correct registration number has been entered during initial programme start-up.

## Coordinate Transformations

This section is intended for those geeks who really want to know how MacCubeView does its coordinate transformations. Consider an orthogonal data set, which we will refer to as a cube although the actual dimensions are arbitrary.



In order to display the outside surfaces of the cube a number of transformations must be performed. The following six matrixes are the ones that MacCubeView uses to do its coordinate transformations. Remember that matrix multiplication is not commutative, so that the order in which that matrices are presented is the order that the multiplication's are performed.

First we want to translate the data cube's centre to the origin of the cube's coordinate system. Here  $x_c, y_c,$  and  $z_c$  are the coordinates of the cubes centre.

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ -x_c & -y_c & -z_c & 1 \end{bmatrix} \quad (1)$$



Next we want to scale the cube in order to obtain the correct aspect ratio, as well as change the size of the displayed cube. Here  $s_x, s_y,$  and  $s_z$  are the scaling factors along the x,y, and z directions.

$$\begin{bmatrix} s_x & 0 & 0 & 0 \\ 0 & s_y & 0 & 0 \\ 0 & 0 & s_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (2)$$

Next we rotate the cube about the cube's x-axis.

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos(\theta_x) & \sin(\theta_x) & 0 \\ 0 & -\sin(\theta_x) & \cos(\theta_x) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (3)$$

Then rotate the cube about the cube's y-axis.

$$\begin{bmatrix} \cos(\theta_y) & 0 & -\sin(\theta_y) & 0 \\ 0 & 1 & 0 & 0 \\ \sin(\theta_y) & 0 & \cos(\theta_y) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (4)$$

Last rotate the cube about the cube's z-axis.

$$\begin{bmatrix} \cos(\theta_z) & \sin(\theta_z) & 0 & 0 \\ -\sin(\theta_z) & \cos(\theta_z) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (5)$$

The last thing that is needed is to locate the cube's centre at the screen's centre. Here  $v_x$  and  $v_y$  are the screen's centre coordinates, in terms of the screen's coordinate system.

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ v_x & v_y & 0 & 1 \end{bmatrix} \quad (6)$$

Multiplying transformation matrices eq(1) through eq(6) together results in the expression for the transformation from the cubes x,y, and z coordinates to the screen's x', y', and z' coordinates.

$$[x' \ y' \ z' \ 1] = [x \ y \ z \ 1] \cdot \begin{bmatrix} A & D & G & 0 \\ B & E & H & 0 \\ C & F & I & 0 \\ Q_1 & Q_2 & Q_3 & 1 \end{bmatrix} \quad (7)$$

The elements of the transformation matrix eq(7) are as follows:

$$A = S_x \cos(\theta_y) \cos(\theta_z) \quad (7a)$$

$$B = S_y \sin(\theta_x) \sin(\theta_y) \cos(\theta_z) - S_y \cos(\theta_x) \sin(\theta_y) \quad (7b)$$

$$C = S_z \cos(\theta_x) \sin(\theta_y) \cos(\theta_z) + S_z \sin(\theta_x) \sin(\theta_y) \quad (7c)$$

$$D = S_x \cos(\theta_y) \sin(\theta_z) \quad (7d)$$

$$E = S_y \sin(\theta_x) \sin(\theta_y) \sin(\theta_z) + S_y \cos(\theta_x) \cos(\theta_z) \quad (7e)$$

$$F = S_z \cos(\theta_x) \sin(\theta_y) \sin(\theta_z) - S_z \sin(\theta_x) \cos(\theta_z) \quad (7f)$$

$$G = -S_x \sin(\theta_y) \quad (7g)$$

$$H = S_y \sin(\theta_x) \cos(\theta_y) \quad (7h)$$

$$I = S_z \cos(\theta_x) \cos(\theta_y) \quad (7i)$$

$$Q_1 = -x_c A - y_c C - z_c C + v_x \quad (7j)$$

$$Q_2 = -x_c D - y_c E - z_c F + v_y \quad (7k)$$

$$Q_3 = -x_c G - y_c H - z_c I \quad (7l)$$

Using eq(7), one can easily write a program to display the surfaces of a data set on a monitor. In this case, the cube data will be projected onto the display screen. However, a problem arises. In general, not all of the screen pixels will have a data value written/displayed in them. The result is an image that has a multitude of little black holes in it. To overcome this problem, we must be a little bit more sophisticated in how we transform the cube data.

MacCubeView functions by transforming the screen pixels onto the cube. This technique ensures that no screen pixel will be unintentionally left black. In the interest of computational efficiency ,i.e., speed, we want to transform only those screen pixels that are located on the surface of the cube. MacCubeView does this in the following way.

**1)** The one, two or three cube sides that face the screen are determined by using eq(7) to transform a vector, which is normal to the cube side. The sign of the transformed vector indicates whether the side faces the screen or not. If the transformed vector is equal to zero, the cube side is perpendicular to the screen and is ignored.

**2)** The four vertices corresponding to each cube face are transformed into screen coordinates using eq(7). Lines are then drawn between the vertices so that the outline of the cube face, as projected onto the screen , is formed and stored. These values form the boundary of the face.

**3)** All of the screen pixels  $x'$  and  $y'$  contained within a boundary are then transformed back onto the cube's face. This transformation makes use of the fact that for each face, we now know  $x'$ ,  $y'$ , as well as one of  $x,y,z$ . This inverse transformation is different for each face of the cube. The inverse transformations are as follow:

for the front and back x-y planes

$$x = \frac{1}{AE - BD} \cdot [E \cdot x' - B \cdot y' + (FB - EC) \cdot z + BQ_2 - EQ_1]$$

$$y = \frac{1}{AE - BD} \cdot [A \cdot y' - D \cdot x' + (CD - AF) \cdot z - AQ_2 + DQ_1]$$

for the left and right y-z planes

$$z = \frac{1}{CE - BF} \cdot [E \cdot x' - B \cdot y' + (BD - AE) \cdot x + BQ_2 - EQ_1]$$

$$y = \frac{1}{CE - BF} \cdot [C \cdot y' - F \cdot x' + (AF - CD) \cdot x - CQ_2 + FQ_1]$$

and for the top and bottom z-x planes

$$z = \frac{1}{CD - AF} \cdot [D \cdot x' - A \cdot y' + (AE - BD) \cdot y + AQ_2 - DQ_1]$$

$$x = \frac{1}{CD - AF} \cdot [C \cdot y' - F \cdot x' + (BF - CE) \cdot y - CQ_2 + FQ_1]$$